

Motivation and Cohesion in Agile Teams

Elizabeth Whitworth and Robert Biddle

Human-Oriented Technology Laboratory
Carleton University, Ottawa, Canada
elizabethwhitworth@gmail.com, robert.biddle@carleton.ca

Abstract. This research explored aspects of agile teamwork initiatives associated with positive socio-psychological phenomena, with a focus on phenomena outside the scope of traditional management, organizational, and software engineering research. Agile teams were viewed as complex adaptive socio-technical systems. Qualitative grounded theory was used to explore the socio-psychological characteristics of agile teams under the umbrella research question: What is the experience of being in an agile software development team? Results included a deeper understanding of the link between agile practices and positive team outcomes such as motivation and cohesion.

1 Introduction

A growing body of evidence suggests that participants in agile team environments find the experience particularly rewarding; more so than most other software development environments. A survey by Cockburn and Highsmith [1], for example, found that agile methodologies were rated higher than other methodologies in terms of morale. Although the ‘hype’ surrounding agile methods is likely a strong contributor to such enthusiasm, there seems to be a solid basis for the association of agile practices with the idea of ‘project chemistry’ or positive ‘team climate’ that can contribute to high performance.

Pockets of literature surrounding software development methodologies contain strong references to socio-psychological issues, such as ego, well-being, control, and team conflict [2]. Even so, there is a lack of basic research into the socio-psychological experience of individuals in agile software development teams — or any other type of software development team, for that matter. Agile and software engineering literature was found to be overwhelmingly based on management and engineering perspectives, concerning the practicalities of software construction, software development processes management, and the hurdles of making it all work within a business context.

The motivation for this research, therefore, was to better explore the animation and excitement observed in practitioners of agile software development. We hoped that examination of positive experiences in agile teams would yield a deeper understanding of the aspects of agile methods that support cohesive team activity. It should be noted that the view of agile in this study is based on, but not limited to, Extreme Programming (XP) practices [3].

2 Theoretical Framework

The framework for this study defines agile software development teams as complex adaptive socio-technical systems. Systems theory [4], on which the framework is based, is a determining influence in small group interaction theory [5, 6], and already used in some instantiations of agile software development. Exploration of system properties, such as feedback and feedforward loops, was valuable in that it supported an understanding of invariant relationships that remained constant despite complex and evolving systems, and would be difficult to obtain through use of simple cause and effect paradigms. Advances in systems theory related to human agency further outlined the importance of considering the team itself as a holistic entity that has an impact on individuals. We found the socio-technical perspective valuable in that it highlights the importance of considering agile teams as systems comprised of both social and technological components. The use of physical artifacts in agile, for example, such as interactive wall charts and automated testing tools, is integral to team coordination and motivation. Practices such as daily stand-ups and pair programming were also viewed as technology, in that they too structure and mediate team interaction. Socio-technical thinking addresses that fact that management and engineering perspectives tend to focus on either the social or the technical aspects of team activity respectively, and thus fail to account for the fact that human and technical subsystems interact and mutually adjust, often with dramatic effect.

3 Method — Grounded Theory

Grounded theory [7–9] is a research methodology that provides a set of procedures for the systematic collection and analysis of qualitative data. Grounded theory is characterized by use of the constant comparative method of analysis. Data and abstract concepts are constantly compared to each other, ensuring the development of an integrative theory that is firmly and empirically grounded in raw data. Twenty-two participants were recruited through networking with members of the agile software development community. Participants included a variety of roles, including developers, interaction designers, project managers, coaches, and quality assurance specialists. All but two of the participants had previously worked in non-agile teams. There were sixteen male participants, and six female participants. Participant interviews investigated the subjective experiences of individuals in agile software development teams. Semi-structured interviews were chosen in order to maintain focus on the theoretical framework, while still leaving room for phenomena significant to participants to emerge. Each interview was audio recorded and transcribed. The transcriptions were broken down into discrete parts and incidents were identified, conceptualized, and named in the process of open coding. Open coding was conducted line by line to ensure thorough grounding and critical thinking about the data. Axial coding was then used to examine the relationships between data. Open and axial coding were performed in parallel as data were gathered, analyzed, and reanalyzed in light of the emerging theory or concepts.

4 Results

Participants in this study, when asked about software development teams characterized by strong feelings of excitement, discussed well functioning teams that ‘clicked,’ ‘gelled,’ or ‘really worked together’ to successfully develop software. Such ‘cohesive’ teams can be held in comparison to non-cohesive teams, which were not associated with feelings of excitement. The distinction between cohesive and non-cohesive teams in this study was separate from the distinction between agile and non-agile teams. Examination of results therefore involved the question: what characteristics of agile teams are related to team cohesion? The following sections outline key aspects of the answer. Interview segments are referred to by three radices, for example (L.4.35), identifying the interview batch, the number of the quoted participant within each batch, and the interview paragraph.

Ease of Interaction One of the main factors associated with enjoyment and excitement in agile software development teams was the ease and speed by which team members could get things done; questions were answered, problems were resolved, and collaborative opportunities were quickly grasped.

(T.3.33/44) And so once we started adopting some of the agile techniques in the previous product I was working on, they were very welcomed. It was instantly recognizable as a pleasant way to work for the people, for the developers, for the managers, for everybody involved; because there is less crisis, it’s easier to manage, you have a better idea of what it really takes to deliver what people are asking for. It’s so much more manageable... I mean it takes out uncertainty, it reduces the risk, crisis management, it’s easy to schedule and plan — everyone’s happier.

A Clear Objective What was found to be one of the most valuable aspects of agile software development methodologies in supporting cohesive teamwork was that they provide a clear team goal — to deliver the most business value to the customer in a certain amount of time. Cohesive agile teams were also seen to maintain a strong focus on developing quality software code. The value of such goals from a team perspective is that they are objectives that most everyone in the team will agree to and happily work towards, without the reservations or divisiveness commonly associated with specifications-driven objectives.

(L.1.82) “Have you been on a really dysfunctional team?” In some ways that spec driven team was a bit dysfunctional, but that might have just been my perceptions. Um — it’s bits like there’s more push to meet the spec than to meet the customer needs. I mean, in my view that’s dysfunctional, but in terms of most software engineering teams, it’s not.

The Planning Game In addition, the team goal is instantiated in a clearly visible and rigorous process as outlined by the agile practices such as the “planning game”. Participants often noted the agile prioritization and planning procedure as a point of pride in their team process, and expressed feelings of excitement in that it allowed them to negotiate to create a plausible plan to develop software:

(T.1.11) There is a lot of tension is just when you develop a feature; there is only so much you can do . . . the product specialist will often come in with a list of a 1000 items on it, [and] there is this dance that takes place and the developer estimates, they say ‘well I can do 50 of those’ . . . And so we will have discussions about that a lot earlier and that also causes a little bit of tension because you know, now the product specialists realize the implications of all they are asking for. And then likewise the developers realize that they actually do need to put some thought into how they are going to do something so that they can provide reasonable feedback and reasonable estimates.

Agile planning was noted as especially valuable as a means of generating group agreement, and was seen to greatly reduce the tension and conflict traditionally surrounding requirements specification and planning. Collective participation and negotiation, in particular, rather than top-down mandates, was seen to strongly support individual involvement, engagement, and buy-in to project planning and activity.

The ability of the agile plan to adjust and allow for specific project and team needs was particularly related to cohesive team activity. The agile practices of allowing developers to estimate their own stories, for example, and the fact that the project plan would then be constructed around these estimates, seemed associated with the feel of ‘rhythm’ or ‘flow’ often discussed with regard to agile teams. Planning in this manner was seen to allow individual pace and team pace to be highly synchronized, where tasks to be completed were neither too difficult nor too easy for individuals in the time allotted. Such team momentum and responsiveness was highly related to excitement and motivation in the agile team environment.

(T.5.5) What I have done in the past has been different, like at other companies where it was more waterfall where you created an obnoxious UI stack from now until next year; and then stuff changed, and I am not a big fan of that. You know there is a lot of investment in terms of writing stuff out and then once you have done all that, the commitment is really high to keep all of that in, even though it may not be the best solution.

Long term planning was also associated with pressure for individuals to ensure that their specific needs were met in a plan that they would have to live by for the course of development. In comparison, iteration-based plans spanning a week, two weeks, or a month, seemed to increase the perception of the current situation as temporary or non-fatal. Flexible and short term planning was therefore seen to allow for more relaxed team relations in planning and implementation, seemingly because there is less at stake. Customization of the agile plan was seen as reducing preoccupation with personal tasks and goals, and allowing a focus on generating agreement and succeeding in team-based software development around a small number of tasks. Interestingly, while the agile plan was seen to highly regulate individual behavior over the course of an iteration, the relatively constrained nature of the agile environment was related to feelings of liberation:

(O.2.15) It's invigorating, because for the first time in your career you know exactly what is expected of you.

Regular Iterative Delivery Iterative delivery was seen to increase the sense of immediacy in the team environment; particularly through the prioritization process, which meant that the majority of the team would always be working on the most important thing. The fact that team members had to deliver a working product at the end of the week or month was further seen to increase the sense of urgency in team interactions, and the capacity for team members to resolve or put aside personal differences in order to work together to deliver. The ability to deliver regularly was several times noted as the main motivator related to agile software development:

(O.2.8) You are working on something and at the end of the week it goes out to the customer and you get feedback right away. And that's great, because your work matters; every day matters. You notice when we have a new product and you are working on it for 6 months and then it is really tough going; because you are not delivering.

Agile practitioners, in delivering working software on a regular basis, were increasingly able to see the purpose and value of their efforts outside the context of development tasks, as well as outside of the context of project activity. Holistic understanding in the larger context of development was seen to provide meaning to low-level tasks, and to support the ability and willingness of team members to create software that would be of value to customers and users:

(T.2.48) You know I am trying to think back in the old way when I was on [another larger project]. I think those user issues were very much filtered. So by the time it got to me [there was] not a lot of background as to what the original task was and what the original situation was, [which] makes it harder to question . . . Whereas I think definitely on this product there is a lot more interaction that way and we can go back and forth and say 'You know I know a customer said he wanted this but I think really it would work better if we gave him this,' which to me makes a better product definitely. I really got the sense of that being a problem in [on the old team]; where you work, you work and work and work on a feature, in the end it isn't exactly what they want but at that point it's too late; and I think in agile you at least modify it on the way and get, end up with a better solution. Does that motivate me better? Yeah I think so.

Thus splitting the planning and development activity into small chunks was related to increased motivation and enjoyment surrounding project activity as a whole:

(T.3.18) On other projects or teams that I was on, it was very much you define a bunch of features and then you worked on them for months and then you bug fix for twice as long and that was the release. So you really didn't know what the big picture was, ever. You just knew that one of these or a couple of these features were what you were supposed to work on for however long and then you are just fixing bugs for ten months or — well that's an exaggeration, but that's pretty much what it felt like.

Splitting the development process into iterations was seen to increase energy in the team environment and allow the team to work together more cohesively. Finally, consistent iterative delivery was seen as a source of pride, and highly related to the motivation to maintain team standards. It was additionally associated with high levels of trust and security, in that agile team members were well practiced and assured in their ability to work together as a team to produce results:

(X.4.11) I don't think that we would have been so consistent in our releases if we had done it otherwise . . . The main point is that I feel more secure about what I'm doing. Very strongly. It's also what they tell you the basic reasons behind unit testing, continuous integration. And it's basically that. And I really feel it like that. You really prove after two weeks that this thing is really working, in a more or less stable way.

(O.2.13) We will do it. We always do it.

5 Interpretation

The main value of agile methods in supporting team cohesion, as well as motivation and excitement, was found to be their ability to support collective team culture. Collective team functioning, where each individual is aware of and invested in the activity of the team as a whole was seen to support feelings of personal security and control; feelings that seem to be absent in many instances of software development in teams.

The stability provided by agile planning and iterative development as a team was seen to offset increasingly unpredictable development environments. Team-based software development is an unstructured, complex, creative, and social, problem-solving and design activity (see Warr & O'Neill [10]). Software development outcomes are further dependent on the resolution of interdependences between team members, the synergy resulting from team-wide discussion and collaboration, and the ability of all team members to share a common vision for the software to be developed. Such activity and coordination must be differentiated from 'knowledge work'. Cohesive software development teams in this study were set apart, not by their ability to 'leverage specialist knowledge' or 'utilize human resources,' but by their ability to get all members of the development team to work closely together towards a common goal. Agile methodologies were seen to allow, support, and even *require* the development of a collective culture over time.

The instantiation of such a culture seems highly dependent on feedback and feedforward mechanisms in the agile socio-technical system, where agile practices support heightened team member awareness of collective tasks, goals, and progress. A study by Eby and Dobbins [11], for example, found that personal preference towards collective group activity is related to positive past experience working in teams and self-efficacy for teamwork, as well as the need for social approval. Positive experience can be seen to result from agile planning and iterative delivery. Positive experience results from agile planning and iterative

delivery, which allows teams to develop a history of success, and thus increase the likelihood of team member involvement in collective activity.

Self-efficacy for teamwork was also seen to be particularly well maintained by agile methodologies. Eby and Dobbins discuss team self-efficacy with regards to efficacy expectations and locus of control [12]. Efficacy expectations involve perceived self-efficacy regarding effort or skill expended in effecting change in a specific context. The related but distinct concept of locus of causality or control, on the other hand, involves the amount of controllability or modifiability of one's environment. Agile practices such as daily team meetings and continuous integration and testing, which focus daily activity on the collective goal of working software, and which provide constant feedback on the validity of individual actions in relation to team goals, were seen to greatly increase perceptions of self-efficacy and control in the team environment.

The negotiation of a flexible plan well suited to team capabilities was further seen to increase individual perceptions of team-wide self-efficacy and control. Other factors such as positive feedback in the agile social and development environment (see Bandura [13]), and detailed and holistic awareness and involvement in project activity, particularly by way of information radiators and the evolutionary development of a working software product, also supported such feelings of self-efficacy. In contrast, software development environments where those conducting development tasks were relatively uninvolved in the development of a detailed project plan, or where team members were not clued-in to the day-to-day activities of others, were related to high levels of discomfort, dissatisfaction, and the absence of perceived team self-efficacy and control, seemingly regardless of the actual state of the project.

This study was not focused on measures of performance, but it should be noted that the presence of self-efficacy in the agile team environment indicates a socio-psychological environment of high-performance. It is generally accepted, for example, that performance in both physical and academic tasks is enhanced by feelings of self-efficacy [14], with effects ranging from the physiological [15] to the socio-psychological. The Collective Effort Model of Karau and Williams [16] adds to such findings. According to this model, individuals will work hard on a given task only to the extent that a) they believe that their hard work will lead to better performance, b) they believe that this performance will be recognized and rewarded, and c) the rewards are ones that they value and desire: "individuals working alone will exert effort only to the extent that they perceive direct links between hard work and the outcomes they want" [14].

Aspects of agile teams, such as transparency, highly focused iterative delivery, and noticeable measures of progress can therefore be seen to increase perceived linkages between day-to-day effort and valued collective goals, such as delivery. Agile environments thus provide motivation for individuals to work harder towards team goals when compared to environments where team members are less aware of the activity of others, or where it is less clear how the team is working together to produce results.

6 Conclusions

This paper has described our study of positive socio-psychological phenomena in agile teamwork. Our theoretical framework was complex adaptive socio-technical systems, and we used the qualitative approach of grounded theory based on interviews with members of agile teams. Our results lead to a deeper understanding of the link between agile practices and positive team outcomes such as motivation and cohesion. In particular, it appears that these positive outcomes are strongly linked to operation and effect of agile practices.

References

1. Cockburn, A., Highsmith, J.: Agile software development: The people factor. *IEEE Computer* **34** (2001)
2. DeMarco, T., Lister, T.: *Peopleware: Productive projects and teams*. Dorset House Publishing Co., Inc. (1999)
3. Beck, K., Andres, C.: *Extreme programming explained: Embrace change* (2nd ed.). Addison-Wesley Professional, Reading, MA, USA (2004)
4. von Bertalanffy, L.: An outline of general system theory. *British Journal of Philosophie of Science* (1950)
5. Arrow, H., McGrath, J.E., Berdahl, J.L.: *Small groups as complex systems: Formation, coordination, development and adaptation*. Sage (2000)
6. Ilgen, D.R., Hollenbeck, J.R., Johnson, M., Jundt, D.: Teams in organizations: From input-process-output models to IMOI models. *Annual Review of Psychology* **56** (2005) 517–543
7. Charmaz, K.: *Grounded theory*. In J. A. Smith, R. Harr and L. Van Langenhove (Eds.), *Rethinking methods in psychology*. Sage Publications (1995)
8. Glaser, B.G., Strauss, A.: *The Discovery of Grounded Theory*. Aldine, Chicago, IL, USA (1967)
9. Strauss, A., Corbin, J.: *Basics of Qualitative Research*. Sage (1990)
10. Warr, A., O’Neill, E.: Understanding design as a social creative process. In: *Proceedings of the 5th Conference on Creativity and Cognition(C&C ’05)*, London, UK (2005)
11. Eby, L., Dobbins, G.: Collectivistic orientation in teams: An individual and group-level analysis. *Journal of Organizational Behaviour* **18** (1997)
12. Bandura, A.: Self-regulation of motivation and action through anticipatory and self-reactive mechanisms. In Dienstbier, R., ed.: *Nebraska Symposium on Motivation 1990*. Volume 38., Lincoln, NB, USA, University of Nebraska Press (1991)
13. Bandura, A.: The explanatory and predictive scope of self-efficacy theory. *Journal of Social and Clinical Psychology* **4** (1986)
14. Baron, R.A., Byrne, D.: *Social Psychology* (9th ed.). Allyn and Bacon, Needham Heights, MA, USA (2000)
15. Bandura, A., Cioffi, D., Taylor, C.B., Brouillard, M.E.: Perceived self-efficacy in coping with cognitive stressors and opioid activation. *Journal of Personality and Social Psychology* **55** (1988)
16. Karau, S.J., Williams, K.: Social loafing: A meta-analytic review and theoretical integration. *Journal of Personality and Social Psychology* **65** (1993)