Agile Practices: The Impact on Trust in Software Project Teams

Orla McHugh, National University of Ireland, Galway

Kieran Conboy, Australian School of Business, Sydney

Michael Lang, National University of Ireland, Galway

Three case studies of agile teams highlight how agile practices can enhance trust among agile team members while also generating new challenges for such teams.

People are core to any software development effort, but they're particularly important in an agile team. The Agile Manifesto places great emphasis on the team, encouraging autonomy and giving individuals the environment and support they need to get the job done.[1] Leadership is shared, and the agile team has substantially more control, which dramatically changes the project manager's role.[2] Managers must have greater trust that their team will make the right decisions and complete tasks in a timely manner. An environment where stakeholders trust and respect each other is both a prerequisite for and a consequence of using agile methods. For example, practices such as collective code ownership and pair programming require developers to trust each other,[2] while other agile practices such as iteration planning, daily stand-ups, and retrospectives help foster that trust.

Agile methods have been the subject of much research, as has trust, but the impact of trust on agile teams has not.[3] To address this gap, we explore how to develop and nurture trust among team members through agile practices.

Trust in Teams

Roger Mayer and his colleagues define trust as the "willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party."[4] Individuals with different personality types, experiences, and cultural backgrounds vary in their propensity to trust others, with the trust level evolving or diminishing over time as they interact with and observe each other.[4–6] Distributed teams face additional challenges such as difficulties controlling processes and quality and missing opportunities to strengthen team morale and trust through face-to-face personal bonds. The limited opportunity to communicate orally can increase miscommunication and delays.[7]

The emergence of agile teams has increased the importance of trust because teams are relatively free to develop as they choose and set targets they consider appropriate.[5] But a cohesive team of members who collaborate and trust each other can be difficult for developers who are accustomed to working predominantly on their own.[2]

Trust requires team members to believe that their colleagues possess the knowledge, competence, and integrity to complete their assigned tasks. It's enhanced when team members help each other.[4] Conversely, trust declines when team members see others as not fulfilling their obligations or see obligations as incongruent, depending also on how vigilantly members monitor each other and the magnitude of perceived discrepancy perceived between obligations and behavior.[8]

Agile Practice Case Studies

Although there are many different agile practices, we focused on the three listed in Table 1—namely, sprint/iteration planning, daily stand-up, and sprint/iteration retrospective—and how they contributed to trust among agile team members. We selected these practices, first because they're among the most commonly used,[9] and second because each one requires the collective participation of all team members with a focus on people, communication, interaction, and teamwork.

Table 1. Agile practices studied.

Agile practice	*Description*
Sprint/iteration planning	Meeting at the start of each sprint/iteration, during which the team collectively defines and plans tasks for next sprint/iteration
Daily stand-up	Short team meeting (10–15 minutes), during which members briefly outline personal accomplishments, plans, and potential impediments
Sprint/iteration retrospective	Meeting held after each sprint/iteration, during which a team reflects on what went well, what didn't, and what could be improved in future sprints/iterations

To help understand how these agile practices contributed to trust, we conducted case studies across three teams. We selected the cases for their diversity in terms of how they were distributed and their industry setting (see Table 2). Each team had implemented an agile methodology for at least six months, was actively using the three agile practices studied, and had all team members available to participate in the study.

Table 2. Case study profiles.

Case	*Agile methodology implemented*	*Team size*	*Time since agile implementation*	*Interviews conducted*	*Average time of interviewees' employment with organization*	*Number of agile practices observed*
Case A	Hybrid of XP and scrum	10 persons	2 years	8 interviewees:* 1 project manager (PM) 1 business analyst 1 technical architect 5 developers	4 years	Sprint/iteration planning (2) Daily stand-up (2) Sprint/iteration retrospective (2)
Case B	Scrum	9 persons	9 months	9 interviewees: 1 scrum master (SM) 1 product owner (PO)†	15 years	Sprint/iteration planning (1) Daily stand-up (3)

				7 developers		
Case C	Scrum	8 persons	11 months	8 interviewees: 2 SMs 1 PO 3 developers 1 technical architect 1 quality assurance (QA) person	5 years	Sprint/iteration planning (1) Daily stand-up (2) Sprint/iteration retrospective (3)

*Two quality assurance team members in Case A, both based in India, departed the team prior to interview. Both had participated in the agile practices observed.

†The product owner represents the customer and manages and prioritizes the customer requirements.

- Case A is a distributed team in a financial services organization that develops software for internal business units.
- Case B is a collocated team in an engineering organization that manufactures and supplies industrial robots.
- Case C is a collocated team in an organization that develops insurance industry software.

Data collection took place over a six-month period and consisted of 25 interviews across a range of roles as noted in Table 2: project managers (PMs), business analysts, technical architects, developers, scrum masters (SMs), product owners (POs), and quality assurance (QA) personnel. We also observed and documented the three agile practices in field notes. During our data analysis, themes emerged across all three teams on how agile practices can increase trust. We elicited opinions about the challenges relating to each theme and reported the results in our findings solely on the basis of participants' views. More details on the research methodology, the case study's broader context, and the interview protocol are available elsewhere.[10]

Results and Analysis

Participants of all three teams reported that trust existed among their members. We found that agile methods increased trust by increasing transparency, accountability, communication, and knowledge sharing and feedback. In the following discussions, we quote interviewees directly.

Status Transparency and Visibility

All three agile teams agreed that using the agile practices increased their projects' transparency and visibility both within the team and in the organization. Our finding is consistent with those of Jan Chong.[11]

For example, the iteration/sprint planning meeting gives team members visibility on requirements, individual task assignments, and estimates agreed for each task: "Everybody here hears the information at the same time, not through others" [QA, Case C]. The daily stand-up provides transparency and visibility on the day-to-day progression of tasks. It's "obvious if someone is not completing tasks" [SM, Case B], and "potential delays are immediately addressed" [PM, Case A] by the team. The retrospectives also provide transparency and visibility regarding achievement of sprint goals. Team members could quickly seek clarifications from each other when delays occurred and immediately improve work processes to avoid recurrence.

Both the QA personnel and the POs noted that they place greater trust in the development team as a result of agile practices. They're aware of what the team intends to deliver in each iteration, "so new features are not a surprise to anyone" [PO, Case C]. The QA and PO have the opportunity to influence task prioritization, raise problems easily, and receive a timely response. From a QA perspective, the development team and POs were aware of testing requirements and set their expectations accordingly. "The

planning meeting is very useful as it is crucial to hear what product management wants. At that point I can question them directly as to what they mean and want, and suggest things that they haven't thought of" [QA, Case C].

Transparency and visibility enhance congruence and vigilance, factors that have been seen to increase trust within software development teams.[8]

Accountability and Collective Responsibility

All members across the three teams openly provide estimates for tasks assigned to them, whether it's a development task, QA task, or requirement for the PO to obtain information from the customer. Team members feel accountable to each other. "Now it's a team goal, so if someone is not completing their task then it means that the team has a harder time" [developer, Case B]. Team members also feel personal responsibility and pressure themselves to deliver what they promised. "If you take the task, you feel responsible for it" [developer, Case C]. By delivering tasks within the timeframe promised, team members demonstrate their competence and trustworthiness to complete similar tasks in the future.

The iteration/sprint planning practice provides an opportunity for team members to determine and agree on estimates. The cases we studied implemented this practice in two different ways. In case A, each developer detailed his or her own estimates. Several team members were considered experts in a particular area, and their estimates "are not really questioned" [developer, Case A]. This trust might reflect the experts' high competency level because "we tend not to [exceed estimates] as a team" [developer, Case A]. In cases B and C, estimating was more collaborative and "decisions are taken collectively" [developer, Case B]. Where estimates varied widely, individuals discussed them until they agreed on a final estimate. This built trust by helping team members develop a better understanding of each other and their competencies to complete a task.

In case C in particular, the developers and QA felt a collective sense of ownership for the software and its quality, which didn't exist prior to the scrum implementation. "I think the sense of shared responsibility and ownership has been strengthened by the idea that you have to stand up and talk about what you are doing and you can't hide. And that everybody has a say or can have a say, if they wish, in what the end product is" [QA, Case C].

The accountability and collective responsibility in agile methods nurture trust by facilitating vigilance, aligning members' perceptions realistically with individual competences and abilities, fostering a sense of benevolence through team solidarity, and enhancing perceived integrity through a demonstrated shared work ethic.[8]

Open and Frequent Team Communication

All teams indicated that prior to implementing agile methods, communication occurred only when it was required—for example, at status meetings. Agile practices require constant interaction and frequent communication, which all team members acknowledged to be a huge benefit and a key factor in developing trust. Conversation take place "that wouldn't happen if these practices weren't being used" [PM, Case A], and the practices build common awareness of tasks and how they affect each other.

Team members weren't afraid to ask for and offer help to each other. "I think it is a major improvement that we have a product owner, and we can ask him anytime and get instant feedback on decisions" [developer, Case C]. Regular communication was very important for Case A, whose team was distributed across three continents. The daily stand-up in particular made it easier to establish team cohesion. "Because of the continuous communication between the team, it helped me feel part of the team" [Technical Architect, Case A].

In all three cases, agile practices integrated new team members integrate much more rapidly. New team members were expected to participate immediately in all three practices, which builds trust and relationships quickly with existing team members. This result is consistent with previous studies that found daily stand-ups around "information radiators" **(**a large, visible display used by software development teams to track progress of tasks**)** to raise awareness and build trust through greater cohesion.[12,13]

Using agile practices has also helped alleviate the distrust that distributed team members from different cultures can experience,[7] which was a concern to Case A. Building good relationships with distributed team members can be difficult, especially when "you haven't met face-to-face" [developer, Case A]. The daily participation of distributed team members helped build trust, especially for the project manager, who had some trust concerns with the distributed team members but found the "stand-up is a great way to keep on

top of [progress]" [PM, Case A].

Thus, open and frequent communication in agile development facilitates trust by promoting a sense of good will and belonging. It also lessens the perceived magnitude of discrepancies between expected and actual output by clarifying the reasons for any variances.[8,13]

Knowledge Sharing and Feedback

All three practices provide an open forum for sharing knowledge and obtaining feedback: "It's good to know you can throw your ideas out there" [business analyst, Case A]. The practices help build trust among team members because "they are having [meetings] on a more regular basis" [PM, Case A] and "we speak a bit more about how we actually work" [developer, Case B]. Individuals are encouraged to voice their opinions in all three practices without fear of repercussions, and no one had ever been "reproached for expressing an opinion" [Developer, Case A]. If a task takes longer than estimated, an individual isn't reprimanded or viewed negatively but instead is helped to complete the task. The practices give team members an "opportunity to question, and once you get a valid answer back…, then that does help [with trust]" [business analyst, Case A]. If the environment was not as supportive, there might be a tendency for individuals to "become more conservative when [they] plan" [developer, Case A] so they can avoid blame, which could be "very detrimental" [developer, Case A] to the project.

The practices also provide opportunities for feedback "and that builds trust, for sure" [SM, Case B]. In cases B and C, the team holds a demo at the end of each iteration or sprint. The customer or PO is expected to attend **the** demos, which gives them confidence that the development team can deliver on what they've promised and, to a certain extent, satisfaction in seeing their requirements as part of a working piece of software. For example, "two weeks ago, that was something I wrote [as a requirement] and now there it is in the screen" [PO, Case C]. The PO can quickly ascertain whether team members are knowledgeable and competent to deliver on what they promise, which increases trust through good will and more realistic perceptions of the team members' individual abilities.

Challenges

Trust was strong in all three teams and reinforced by agile practices, particularly from the QA and PO perspectives. We identified four factors that increased trust but also created challenges for teams, as reported by participants of the study (Table 3).

Table 3. Challenges facing agile teams in relation to trust.

Agile practices	***Factors that have an impact on trust***	***Challenges***
Sprint/iteration planning	Transparency and visibility of project status to all stakeholders	* Organisational personnel external to the team have visibility on task status but perhaps not on actual causes of delays, if any
Daily stand-up	Accountability and collective responsibility	* Team members tend to inflict pressure on themselves to deliver * Development of tension between the product owner and the project team
Sprint/iteration retrospective	Open and frequent communication within the team	* Developers too accessible to the product owner
	Sharing knowledge and obtaining feedback	* Teams tend to underestimate tasks * Teams see little value in the sprint/iteration retrospective

External Visibility on Project Tasks

In two cases, developers identified a challenge arising from broader organizational access and visibility to the status of all project tasks. This visibility triggered comments on the progress of particular tasks by individuals who were not part of the team,. Some team members considered the comments inappropriate because these individuals might be unaware of reasons for a task's delay.

One case dealt with this to a degree by letting any employee attend and observe the daily stand-up or the demo and retrospective but restricting their comments or queries to the demo part of the retrospective. This could help build trust between the development team members and non-team members and keep nonteam members informed about what's happening.

Team Pressure

Many team members said they felt extra pressure to complete a task within a specific timeframe once they had committed to the team that they would do so. They felt they were letting the team down if they weren't able to complete the task on time. This pressure was entirely self-inflicted, but it might be consequent to the increased visibility of tasks and personal accountability to the team.

A failure to deliver tasks might also demonstrate a lack of competence and thereby diminish trust between team members.

Tension between the PO and the Team

Interviewees differed about whether POs were part of the team. Some team members considered them to be a team member and others did not. In two cases, the POs themselves didn't feel they were part of the project team and indicated that a certain amount of tension exists between them and the rest of the group. Nevertheless, the PO must trust the development team to do what it says it will do, and the development team must trust the PO not to overburden it with work.

The POs reported a constant struggle with their loyalties. They saw themselves situated between the business management and the development team, but they occasionally side with the development team to justify decisions to the business management. The difficulties in meeting the demands of different stakeholders can make the PO role unattractive for employees. Individuals in this role must be able to handle conflict and manage the expectations of all stakeholders.

Developers Accessibility to the PO

The three agile practices increased communication and helped build relationships and trust between the PO and the development team. In one case, this made the PO feel easier about approaching developers to make requests or ask their opinion at any time. From a developer's perspective, this access must be managed and controlled so that ad hoc requests aren't suddenly added to the requirements list without agreement from the team.

At the same time, when a team doesn't agree to work on an ad hoc request, the PO can feel that the team is inflexible and not agile.

Tendency to Underestimate Tasks

Two of the three cases showed a tendency to underestimate task durations. Team members were mostly experienced and quite familiar with the development environment, but they found it difficult to accurately estimate unknown tasks—even with the planning meeting as a forum for sharing information.

The projects openly acknowledged this fact, but it didn't appear to concern the team members or affect trust levels between them. If a task wasn't completed, the prevailing view was that it could always move to the next iteration or sprint.

Value of the Retrospective

Although the teams considered the concept of a retrospective to be important, most interviewees placed little value in it. They reported that it was routine and raised few issues. When an issue did come up, it might be discussed but seldom generated action points or follow-up. As implemented, the retrospective limited the communication, knowledge-sharing, and feedback mechanisms for building trust.

There was also some concern expressed over the retrospective participants. In one case, the PO was regularly absent from the retrospective, which affected the relationship with the team. The team members

reported little trust between them and the PO.

Despite the challenges they raise, the three agile practices we examined helped the groups we studied function more cohesively as a team, rather than just a group of individuals working together. Mandatory participation in the practices builds trust among agile team members. The practices can also quickly highlight any trust issues that exist in a team.

There are some limitations and avenues for future research arising from this study. First, other factors can also enhance trust, such as the organizational culture and the experience and personalities of individual team members. Furthermore, the teams we studied were relatively small, which is typically the case when agile approaches are adopted. Consequently, trust might be due to the teams' size and proximity as opposed to agile practices. Although we made every effort to link the agile practices we studied to trust, studying the practices in larger teams would address this limitation. In addition, our study focused only on trust among team members. Future studies could look at other aspects of trust and at how agile and other practices affect trust between the team and other stakeholders.

Accountability seemed to be very high in this study, which might reduce the importance of trust. This doesn't render the findings any less valid, but a study of agile teams that exhibit less accountability might reveal lower trust levels and more negative effects. Finally, the challenges we report reflect team members' trust not only in each other but also in the agile process. We focused on the former, but future research could examine trust and confidence in the agile process being employed.

As is typical with case study research, there are several limitations regarding its generalizability and scope. Future research could use more quantitative, explanatory methods to determine the extent to which the trust factors raised in this study are representative of the field. Trust is a sensitive and often a subtle concept, so more generalizable studies that identify the extent of the issues would make a significant contribution.

Acknowledgments

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero-the Irish Software Engineering Research Centre (www.lero.ie), and by grant aid received from Enterprise Ireland as part of the Global Agile Innovation project.

References

1. AgileAlliance, "Manifesto for Agile Software Development," 2001; www.agilemanifesto.org.
2. S. Nerur, R. Mahapatra, and G. Mangalara, "Challenges of Migrating to Agile Methodologies," *Comm. ACM*, vol. 48, no. 5, 2005, pp. 72–78.
3. E. Hasnain and T. Hall, "Investigating the Role of Trust in Agile Methods Using a Light Weight Systematic Literature Review," *Agile Processes in Software Engineering and Extreme Programming*, Springer, 2008, pp. 204–207.
4. R.C. Mayer, J.H. Davis, and F.D. Schoorman, "An Integrative Model of Organizational Trust," *Academy of Management Rev.*, vol. 20, no. 3, 1995, pp. 709–734.
5. T.K. Das and B.-S. Teng, "Trust, Control, and Risk in Strategic Alliances: An Integrated Framework," *Organization Studies*, vol. 22, no. 2, 2001, pp. 251–283.
6. J. Iivari and N. Iivari, "The Relationship between Organizational Culture and the Deployment of Agile Methods," *Information and Software Technology*, vol. 53, no. 5, 2011, pp. 509–520.
7. B. Ramesh, L. Cao, and K. Mohan, "Can Distributed Software Development Be Agile?" *Comm. ACM*, vol. 49, no. 10, 2006, pp. 41–46.
8. G. Piccoli, and B. Ives, "Trust and the Unintended Effects of Behavior Control in Virtual Teams," *MIS Q.*, vol. 27, no. 3, Sept. 2003, pp. 365–395.
9. "State of Agile Development Survey 2009," white paper, VersionOne, 2009; http://pm.versionone.com/StateofAgileSurvey.html.

10. O. McHugh, K. Conboy, and M. Lang, "Using Agile Practices to Build Trust in an Agile Team: A Case Study," *Proc. 19th Int'l Conf. Information Systems Development*, **Springer** 2010, pp. 503-516
11. J. Chong, "Social Behaviors on XP and Non-XP teams: A Comparative Study," *Proc. Agile Development Conf.*, **IEEE Computer Society,** 2005, pp. 39–48.
12. H. Sharp and H. Robinson, "Collaboration and Co-ordination in Mature eXtreme Programming Teams," *Int'l J. Human-Computer Studies*, vol. 66, no. 7, 2008, pp. 506–518.
13. E. Whitworth and R. Biddle, "Motivation and Cohesion in Agile Teams," *Proc. 8th Int'l Conf.* (XP 2007), **Springer** 2007, pp. 62–69.

Orla McHugh *is a lecturer in information systems at National University of Ireland, Galway. Her research interests lie in the area of agile software development with a specific focus on control in agile teams. McHugh has a PhD in information systems from NUI Galway. She's a member of the Association for Information Systems. Contact her at orla.mchugh@nuigalway.ie.*

Kieran Conboy *is an associate professor at the University of New South Wales and the Lero research center in Ireland. His research focuses on agile systems development approaches as well as agility across other disciplines. Conboy has* ***a PhD*** *in* ***information systems*** *from* ***the University of Limerick****. He's associate editor of the* European Journal of Information Systems*. Contact him at k.conboy@unsw.edu.au*

Michael Lang *is a lecturer in information systems at National University of Ireland, Galway. His principal research interests are systems development approaches, information systems security and information systems education. Lang received his PhD in* ***information systems*** *from the University of Limerick. He's on the editorial board of several international journals and on the executive committee of the International Conference on Information Systems Development. Contact him at michael.lang@nuigalway.ie.*

Abstract: Agile software development involves self-managing teams that are empowered and responsible for meeting project goals in whatever way they deem suitable. Managers must place more trust in such teams than they do in teams following more traditional development methodologies. The authors highlight how the use of agile practices can enhance trust amongst agile team members. They also present challenges that agile teams can face as a result of using agile practices. Their results are based on the findings from three case studies of agile software development teams.

Keywords: Agile methodology, agile practice, daily stand-up, planning, retrospective, trust, culture, distributed software engineering

DOI: http://doi.ieeecomputersociety.org/10.1109/MS.2011.118