The Estimated Interest Amount should be filled out when they pay the TD item. The Probability of being a future problem (i.e. the interest probability) in this case was *low*. In this case, they also added in the card the Date when they paid the TD item, *05/23/2013* and how long it took them to pay off the item, also *twenty minutes*.

These boards represent some of the boards used in the team's informative workspace. Some teams used a specific board to manage TD, as shown in Figs. 1 and 2, other teams used the Kanban board and put the TD items together with the tasks of the sprint. Furthermore, some teams also placed the list of TD items in the tool used to manage the project.

3.2.2 Tools

The teams had two code quality analysis tools:

- Code Climate: a tool for quality analysis of code repositories (https://codeclimate.com/).
- Sonar Qube: a code quality analysis platform that has a plugin that identifies TD (http://www.sonarqube.org/).

4 Research Methods

Data was collected and analyzed for this study through interviews and questionnaires. Before data collection, the teams spent some time identifying TD items in their projects. Below, we first describe how TD items were identified, and then we describe our data collection and analysis methods.

4.1 Technical Debt Identification

In the two offerings of the XP Lab, we followed slightly different steps to help the teams to identify TD.

4.1.1 XP Lab 2013

Four weeks after the students started working on their projects, we gave a presentation about TD, as some students were familiar with the term, but others were not. After the presentation, we had a discussion and we encouraged the students to talk about their views of TD. We also talked about some concrete examples they had in their projects. The discussion lasted for about one hour and after that, each team had to prepare a TD board, where they would have to document their TD items (Fig. 3 shows an example of a TD item). Each TD item was written on a card, and in this card they had to fill out a list of topics, after that it was then pinned to the board. The card structure was based on the template developed in [12]. Each team had a board with a set of cards representing TD items. Every team that identified or incurred a debt put the information on the board.

4.1.2 XP Lab 2014

In the XP Lab 2014 offering, we made a presentation about TD for all the students together for 30 min, about two months after the course began. After that, we discussed it for another 20 min, during this time the students could clarify their doubts about TD. The students already had the code quality analysis tools available, Sonar Qube and Code Climate, since the beginning of the course. We did not impose the use of either the boards or the tools to identify TD. We showed them the meanings of TD items and some examples in each project. We also presented examples of a TD board and of the categories they could use to identify TD items. Then each team decided whether they would monitor TD on their project or not.

4.2 Second Step – Interviews and Questionnaires

Data collection was done differently in the two XP Lab offerings. In both cases, similar data was collected both at the beginning and at the end of the course.

4.2.1 XP Lab 2013

Eight weeks after the teams started to identify TD items and fill the boards with cards, we carried out a face-to-face interview with the pairs in each team. The interview motivation was to verify the influence on the team of the TD visibility. The interview was composed of twenty questions, with open-ended and multiple-choice questions, separated into the following topics[2]:

- The concept of TD.
- Were there any changes in the software development process?
- Negotiation with clients.
- About the experience of identifying TD.
- What is the relevance of identifying TD?
- What is the impact on software quality?
- Do the teams pay off TD?
- Will the teams pay off some TD?

Four weeks after the first interview, at the end of the course, we did the last interview with an open format and we performed it for each team. In the last interview, each team was invited to talk about the experience of making TD explicit. Each interview took about twenty minutes.

4.2.2 XP Lab 2014

In this edition of the course, we decided to apply a questionnaire on what each team member thinks about TD (the questionnaire was answered by the students individually; this approach was taken to try decrease a possible bias). This questionnaire was applied

[2] It is possible to access all the questions in the following link https://www.dropbox.com/sh/gen3dr97xxofs21/AACo11oqbBsaCprOCQtSYv5Ja?dl=0.

one week before the class received a talk about TD. We sent a link to the questionnaire by email and then the students had one week to answer it.

The questionnaire was composed of seventeen open-ended and multiple-choice questions, separated into the following topics[3]:

- Software quality
  - What does the team do about quality?
- Familiarity with TD.
  - Do you know about and use the TD concept?
- How TD is used in the project.
  - Are you using the Sonar Qube or Code Climate report?
  - Are you using a TD list?
  - Have you paid any TD item?
  - Is there any evidence that having the TD items visible has an influence on the team?
- Did you identify any TD item that was not identified by the tools?
- Are you going to consider TD in future projects?
- Do you think the TD concept is relevant?

The same questionnaire was applied a second time at the end of the course. The aim was to see if there was any change in the team members' behavior.

4.3 Third Step – Data Analysis

For the data analysis, we used coding techniques from the grounded theory approach [13]. Grounded theory methods are aimed at building or discovering a theory. In this approach, the data analysis proceeds in three interdependent steps: open coding, axial coding, and selective coding. In the open coding step, the researcher interprets the data to identify patterns and define codes, "…event/action/interaction, are compared against others for similarities and differences; they are also conceptually labeled […] conceptually similar ones are grouped together to form categories and their subcategories…" [13]. In axial coding "…categories are related to their subcategories, and these relationships tested against the data…" [13]. Then in selective coding "…all categories are unified around a central 'core' category and categories that need further explanation are filled-in with descriptive details…" [13].

For data analysis, we used the NVivo[4] tool, which is widely used for analysis of qualitative data. In this case, the goal was not to use grounded theory to develop a new theory but only use its coding steps to answer our research questions.

We also analyzed the source code of the projects with the Sonar Qube and Code Climate tools, to try to identify relationships between team's beliefs and the reports from these tools.

[3] It is possible to access all the questions in the following link https://www.dropbox.com/sh/gen3dr97xxofs21/AACo11oqbBsaCprOCQtSYv5Ja?dl=0.

[4] http://www.qsrinternational.com/support/downloads/nvivo-9.

5 Results

In this section, we describe our findings organized by the coding steps. As the main question in the both editions of XP Lab was the same and the obtained results were similar, we analyzed the results of both editions together.

5.1 Open Coding

In this step, the data analysis was conducted by reading the transcripts of the interviews and also the answers from the questionnaires. We applied the coding process to this material, line by line. In this phase, we discovered the open codes. In Table 2, we list three code samples. It is possible to access the list of the open codes that emerged from this first codification in an appendix[5].

Table 2. Example of codes resulting from open coding

Open codes	What they talked about
Changed attitude of the teams	The team discussed more the tasks they have to do before incurring a TD and they thought more before taking the decisions.
Communication	After the identification of TD items, the team had more discussions.
Maintainability	The identification of TD items helps the teams to know that there will be some changes in the software in the future.

5.2 Axial Coding

The open codes were reassembled in new ways during axial coding to form categories. The goal was to create a higher abstraction level. Thus, codes were grouped to form subcategories, and in turn, they were organized into categories. This process was highly iterative, with codes and categories forming and re-forming as more data were incorporated into the evolving understanding [13].

In Fig. 4, it is possible to observe the list of categories and subcategories resulting from axial coding analysis. The first level is the main category resulting, this category emerged from the subcategories of the second level, the subcategories are resulting from the codes emerged in the third and fourth level.

One of the most important influences when we make a list of TD items is the *attitude of the team (team behavior)*, "…registers by not forgetting, there was a change in the attitude of the team…" and "…increased people's concern regarding the Technical Debt…". The team had less 'untouchable' expert professionals and behaved more as a whole team. They *talked more about the TDs* "…we discussed these debts. Otherwise, the project would not have advanced…" and thought about the necessity of incurring it. *It helped* them to have the same understanding of the concept of TD because they discuss it (*TD concept*). In addition, if the team members were not sure whether to incur

[5] It is possible to access all the codes in the following link https://www.dropbox.com/sh/gen3dr97xxofs21/AACo11oqbBsaCprOCQtSYv5Ja?dl=0.

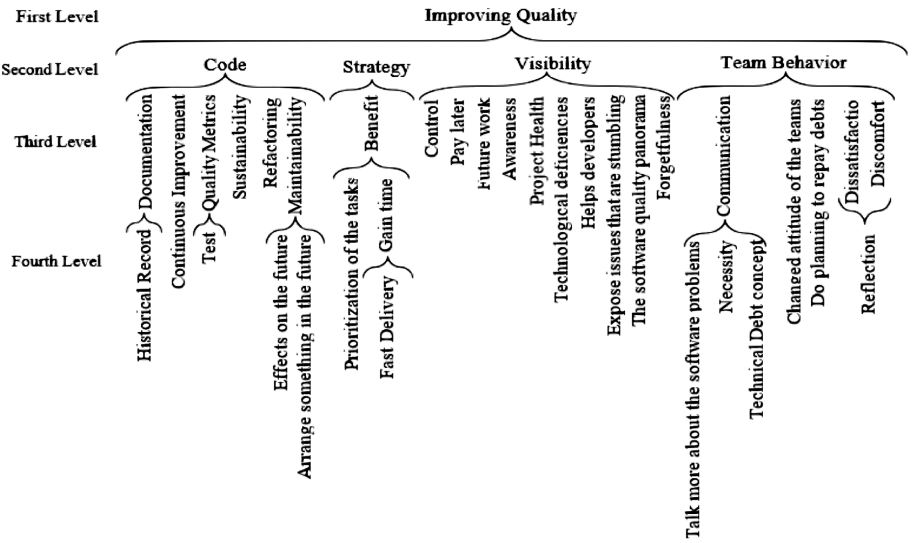

Fig. 4. Categories and subcategories of the XP Lab 2013 and 2014 edition

a TD item or not, the team member debated with another team member to help him to take this decision, thus improving the team's *communication*. Team members started *talking more* with each other and because of it, they knew what part of the project was being modified and the problems of the software, "…It was easier to remember that we have to fix things, debts…". Furthermore, if team *communication* was good, they were more comfortable to share with each other their difficulties. After the team began to identify TD, developers discussed their decisions rather than just doing something and moving on, now that all the software problems were more clear to the team members, "…usually only the person thought or knew about it (…), now with the TD it becomes clearer as well…". They began to argue among themselves, before incurring a debt, "… As evidenced here we even got everyone talking about the debts, instead of just looking to give a quick solution and move along…".

They started to *think more about* if it was *necessary* or not to incur some TDs. Several times they concluded that it was not required, a team member says: "[…] From the moment I started to think about that item, which was debt, I asked myself about what is the current cost, compared to the future cost. Because if the cost of doing now is less, then it's better to fix it now…", other student says: "…we think twice before making a TD…".

When TD items were visible, the team had more *control*, over whether they will pay off the TD item, whether they will incur more debts or not, or whether they will incur and *pay later*. Moreover, this facilitates planning to repay the TD items, "**…** we analyzed some of the debts, now we will plan, what we might kill, to kill some of those debts, then it becomes easier to make this analysis…". Therefore, if they incur a TD item, they would make it visible, would monitor it, and sometimes they would look back at this TD item. This way, they always thought about *continuous improvement*. Thus, the team could be defining a strategy to pay off or not some TD items to improve the *code* quality.

However, if the team member incurred a TD item, and never paid, the project would probably lose *quality.* Nevertheless, they might incur TD to prioritize other tasks or as a business decision to *deliver fast* and then used it *strategically*.

The TD item list presented indications on whether the code quality was improving or not, and helped them to understand the current development state It also indicated if they would have a lot of future work and future *refactoring* to do in that code. Indeed, if the software had TD items, the team would probably need to perform some *refactoring*, and it directly affected the *sustainability* of the project. Furthermore, the team could do an analysis of the TD items and they could be defining a software's *quality metrics* to help them monitor the software quality, for instance, *test*. They also could identify *technological deficiencies* and define possible directions to improve their technical skills and not repeat the same mistakes, in order to mitigate the occurrence of those TD items that were recurrent.

The teams used the TD item list as *documentation* providing a *historical record* of the immature parts of the project. Therefore, each team knew that some parts of the project should be improved; it was possible to see if there was a TD item to pay off. This *documentation* helped the teams to maintain the code and it reflected on the *health of the project.* In addition, it affected the teams by sometimes causing *dissatisfaction* with the *quality* of the software. Many team members saw that it was very *uncomfortable* to arrive at work and see that the *health of the project* was not so good. If the source code health of other teams was good, it was even more *uncomfortable*. Therefore, this process of the team having discussions, and having *dissatisfaction* impacted in the following *reflection:* when a team member decided to incur or not TD, thought and discussed it, he better understood the problem that he had to solve. This often resulted in the non-insertion of a TD item in the code, since it only lacked the understanding of what should be done.

Considering TD implied generating a culture focused on *quality*. It affected factors related to project continuity. It has an influence on the cost and the viability of *maintainability* and evolution of the project. A developer said that, if they did not have the TD items *visible* it was so difficult to identify *the software quality landscape,* "…it was difficult for you to identify the whole landscape…". Therefore, if in the *future* the team needed to make some changes in the legacy code, they already knew what they were and where the problems were located. It provided a general *awareness* to the team about the problems of the software. It also might help a new team member that did not know about the code to have a notion on the quality of the code and the location of the code problems.

5.3 Selective Coding

Selective coding constituted the third stage of data analysis, with the objective to refine and integrate categories, unveiling a category deemed as central, encompassing all the others. The full potential of abstraction was employed to incorporate the full scope of the data investigated and coded [13].

In our case, the objective was not to generate a theory, but rather to identify the main categories. The aim was to describe the impact of TD awareness.

The categories, *Strategy, Team Behavior, Code*, and *Visibility* represent the main influences on teams due to making TD items explicit. Each of these categories captures part of our results, although none of them describes the phenomenon entirely. For this reason, another abstract category is required, a conceptual idea on which all categories are included. As such, we concluded that the resulting core category might be a perceived notion on ''*Improving Quality*".

All teams progressed towards creating a culture of *Quality* of the *code*, *team,* and *project*. This arose primarily because each team member started to think more about the need to incur TD items. Many times they decided that incurring TD was not necessary in a given situation. When a team member was not sure about the necessity or not to incur the debt, they spoke with other members to make a decision. Therefore, the team improved their communication and then it was clearer what each member was doing. So, it was easier to understand the objective of the project. Then, making TD explicit has a direct implication on the *Team Behavior*. Most of the team members said that after they started to identify TD, they talked more with each other, thought more about the real need to incur debt or not, discussed more about code quality, refactored more frequently some parts of the code, knew where the code problems were, and where each team member was working at any time on the project. In addition, when communication among the team was good, people felt more comfortable to expose and discuss their problems with the team. The team then became more a group that works together, rather than a group of experts on different parts of the system.

In some situations incurring some TD items was a *Strategy* to gain some time, due to the time to market. Moreover, if there was a list of TD items it might be possible in the future to correct them, by refactoring. The list of TD items was used as documentation, this enabled the historical record of the TD items list that the code contains. When the need to change a particular part of the system appeared, it was possible to verify if it had some TD item and if this debt would affect such functionality. Therefore, the team members had the option of paying it off or not. The documentation helped in the *Visibility* of the project's health. If the software had any TD, probably it would have more deficiencies.

Finally, when a team had the TD items list they automatically became *Aware* of the TDs of the project, consequently about the software quality. So, the team could think about it, before possibly incurring in another TD item. The team could decide when and where they would improve the software quality.

5.4 The Code Analysis with the Tools

In XP Lab 2013 edition, we analyzed one project with Code Climate tool (supported the language used in the project, in this case, Ruby). Further, we analyzed two projects with the Sonar Qube tool. In the XP Lab 2014 edition, we also analyzed two projects with Code Climate tool and four projects with the Sonar Qube. We considered the Code Climate metric, grade point average[6] (GPA) and in the Sonar Qube the following metrics: code smells, security, reliability, maintainability, duplications, documentation,

[6] https://docs.codeclimate.com/docs/code-climate-glossary#gpa.

issues, technical debt rating, complexity, size, duplicated blocks, bugs & vulnerability and duplications. The GPA of the Mezuro's code and the Monitoria project increase after the teams considered TD. Then the quality of project increased, it means that the remediation (the amount of effort required to improve a software issue), was 0 to 2 M (too short). In the analysis with Sonar Qube of the projects: Game VidaGeek, Tiktak, Arquigrafia, Family Tree, Social Networking Startups and Specialist in Sport we did not identify large variations in the measured metrics comparing the two different versions of each project (Before and after they consider TD). However, in 5 of the 6 projects, the rate of duplication and the duplicated blocks decreased after the team started to consider TD, indicating an improvement in code quality.

It is important to highlight that the students said that most of TD items were not identified by the tools, which explains the modest size of the changes in these metrics. For instance, one team was using Handsontable[7] for data entry, and they had a problem with the validation of the data. The presence of this type of TD item, nor the impact of paying it off, could not be identified with Sonar Qube or Code Climate reports. In general, many types of TD items (e.g. those related business rules) cannot be identified through static metrics.

6 Discussion of the Findings

The teams had some similar views on the importance and benefits of making TD explicit. A significant finding is that the teams considered it very helpful because they could see the whole landscape of the software quality (they knew which part of the software had immature code). They also emphasized that it was very useful to have a board where every day they could see the health of the code. Before becoming aware of TD, the team members reported that they sometimes incurred TD but never remembered to go back and correct it. But after considering TD, they thought about the necessity of incurring TD and often decided against it. Also, they could see the TD list and so they did not forget the TD items that needed to be addressed. They discussed more about how to implement the tasks, also they talked more about the problems of the software because they had the list of the TDs visible. This process of thinking about incurring or not TD, discussing about it and reviewing the TD during the project can create a culture focused on improving the software quality.

In addition, in this study we explored some ways of identifying and monitoring TD. Our subjects found some form of a TD board very useful for documenting TD, making it visible, and adjusting both the TD board and their behavior accordingly. By using the TD board, they always know the list of software deficiencies so have a constant reminder of how to organize their work and improve the software. As a complementary aid, they may use tools to help them to identify and monitor TD occurrence. However, it is important to highlight that tool reports provide a static analysis of the software quality and some TD item could not be identified using static metrics.

[7] https://handsontable.com/.

The results of this study could motivate teams to consider TD further, to help developers convince leaders and directors, the decision makers, to start considering TD. These approaches used by the XP Lab teams, such as boards, cards and tools can help teams in companies to deal with TD. In addition, they could define the list of TD items that are crucial to the project but hard to identify with the tools. As a result they can define a strategy to deal with the TD over time.

6.1 Threats to the Validity

In this study, we took some actions to mitigate possible biases, we describe these in the following points:

- *Construct validity (credibility):* We used multiple data collection approaches with the aim to reduce possible bias. When planning the interviews and the questionnaire we discussed the best way to formulate the questions. We did a first interview and questionnaire with one member of the group as a pilot test. Based on this test we reviewed the questions. We did not include these data in the final analysis. One thing that it is important to highlight is that the students might not have understood the main meaning of the questions correctly, in the interviews and questionnaires. Because of this, in the interviews, if the student did not understand the question the interviewer explained the question for them. The researcher was available throughout both studies if the students had any questions.
- *External Validity:* This study can be replicated in other academic courses, also in companies. In both cases, the study can be separated into two parts and can be applied in these situations: one in teams that do not consider technical debt yet, to verify if awareness of TD influences something in the team, such as communication. Furthermore, this study can be analyzed with teams that already consider technical debt, by identifying, monitoring and managing if it is possible to identify some changes in the team behavior and in the software quality. If they have a historical record, we could also measure the software quality with tools. Finally, to carry out this study is not necessary to make significant changes in the team's environment, which makes feasible to replicate in companies.
- *Internal Validity:* We analyzed the data separately, first the interview transcriptions, then the questionnaire responses, and then compared and merged the findings that were relevant and had a lot of evidence in the results of both studies. In the case of any doubt about a specific point, we went back to the data and re-analyzed them. After that, the advisor and co-advisor read the results and if they indicated some points to be re-analyzed, the researcher re-analyzed the data. We did analysis and re-analysis many times until we were sure of the conclusions.
- *Reliability:* To interpret the data we followed the coding techniques from the grounded theory steps. Also, the data analysis was made by a single researcher, however, the results of the analysis were discussed by the two researchers and with the advisor and co-advisor, every time a doubt arose the data were re-analyzed. Also, this paper is a result of an analysis of the data that lasted two years, where the researcher compared the data many times. Furthermore, the preliminary results were

presented and discussed at seminar[8] attended by top researchers in this field. It is important to observe that when we infer that awareness of TD could impact software quality, we are describing the perceived quality by the team.

- *Objectivity:* The results show the information derived from the data, the codes and categories emerged were related with the data quotes.

7 Final Considerations and Future Works

This work describes results about the influences of making TD explicit in an academic setting. Our results show the importance of making TD visible and how that influences teams. It is important to point out that no negative influences were identified. The team members were always very excited about the results of making TD items visible. As communication in the team was improved, all team members thought more about quality, not just specific members. The "agile" culture of the teams improved and in addition, the team believed that it was easier to show the impact of the TD level to clients, showing that it is possible to invest some time to improve the quality. The main results emphasize the Extreme Programming values and helped the teams to support values such as communication at all levels, courage to change and feedback to continuously improve the software.

In future work it is important to verify the influence on the team in the long term, especially concerning speed and code quality. It is also important to create ways to compare the perceptions of the developers with the results of the tool reports. For instance, in these studies the students believed that when they started considering TD, the project quality improved, but when we analyzed the code with the tools, we saw that the reports did not indicate significant changes in source code quality. Then, it is interesting to investigate why this happened, and possible future solutions.

Acknowledgments. We are grateful to all the students and TAs of the XP Lab course (2013 and 2014 offerings) for providing valuable data for this research. We would like to thank IBM, CNPQ, CAPES and FAPESP, too, for funding this work.

References

1. Santos, V. et al.: Uncovering steady advances for an extreme programming course. CLEI Electron. J. **15**(1) (2012). paper 1
2. Edith, T., Aybuke, A., Richard, V.: An exploration of technical debt. J. Syst. Softw. **86**, 1498–1516 (2013)
3. Kent, B.: Extreme Programming Explained: Embrace Change. Person Education Inc, United States (2005)
4. Ward, C.: The WyCash portfolio management system. In: Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications, pp. 29–30 (1992)
5. Extreme Programming Projects – CCSL. (http://www.ccsl.org.br/oldwiki/index.php)
6. Arquigrafia. http://www.arquigrafia.org.br/. Accessed May 2016

[8] http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=16162.

7. VidaGeek. http://aprenda.vidageek.net/. Accessed May 2016
8. Mezuro. http://mezuro.org/pt. Accessed May 2016
9. CoGroo. http://ccsl.ime.usp.br/cogroo/. Accessed May 2016
10. Oliveira, R.M., Goldman, A., Mello, C.: Designing and managing agile informative workspaces: discovering and exploring patterns. In: Proceedings of the 46th Hawaii International Conference on System Sciences (2013)
11. Oliveira, R., Goldman, A.: How to build an Informative workspace? an experience using data collection and feedback. In: Agile Conference (2011)
12. Seaman, C.: Technical Deb Minicourse. At the University of São Paulo (2013)
13. Corbin, J., Strauss, A.: Grounded theory research: procedures, canons and evaluative criteria. Zeitschrift fur Soziologie **19**(6), 418–427 (1990)
14. Monitoria. www.monitoria.ime.usp.br. Accessed May 2016
15. System Specialist in Sport. http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0145733. Accessed May 2016
16. Li, Z.G., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. J. Syst. Softw. **101**, 193–220 (2014)
17. Poliakov, D.: A systematic mapping study on technical debt definition. Lappeenranta University of Technology School of Industrial Engineering and Management Degree Program in Computer Science (2015)
18. Lim, E., Taksande, N., Seaman, C.: A balancing act: what software practitioners have to say about technical debt. IEEE Comput. Soc. Softw. **29**(6), 22–27 (2012)
19. VersionOne.: State of Agile Report (2015). http://info.versionone.com/state-of-agile-report-thank-you.html
20. Guo, Y., Seaman, C.: A portfolio approach to technical debt management. In: Proceeding of the 2nd Workshop on Managing Technical Debt (2011)
21. Bavani, R.: Distributed agile, agile testing, and technical debt. IEEE Softw. **29**, 28–33 (2012)
22. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: from metaphor to theory and practice. IEEE Softw. **29**, 18–21 (2012)
23. Martini, A., Bosch, J., Chaudron, M.: Architecture technical debt: understanding causes and a qualitative model. In: Proceedings of the 40th Euromicro Conference on Software Engineering and Advanced Applications (2014)
24. Sterling, C.: Managing Software Debt: Building for Inevitable Change. Addison-Wesley Professional, Boston (2010)
25. Curtis, B., Sappidi, J., Szynkarsky, A.: Estimating the size, cost, and types of technical debt. In: Proceedings of the IEEE 3rd International Workshop on Managing Technical Debt (MTD 2012) (2012)
26. McConnel, S.: Managing Technical Debt. http://www.construx.com/File.ashx?cid=2797. Accessed April 2008
27. Buschmann, F.: To pay or not to pay technical debt. IEEE Softw. **28**(6), 29–31 (2011)
28. Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C.: Managing technical debt in software engineering. Dagstuhl Rep. **6**, 110–138 (2016)

Agile in Organizations

Don't Forget to Breathe: A Controlled Trial of Mindfulness Practices in Agile Project Teams

Peter den Heijer[2(✉)], Wibo Koole[3], and Christoph J. Stettina[1,2]

[1] Centre for Innovation The Hague, Leiden University,
Schouwburgstraat 2, 2511 VA The Hague, The Netherlands
c.j.stettina@fgga.leidenuniv.nl
[2] Leiden Institute of Advanced Computer Science, Leiden University,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
pdheijer@gmail.com
[3] Centrum voor Mindfulness,
Raadhuisstraat 15, 1016 DB Amsterdam, The Netherlands
wibokoole@cvm.nl

Abstract. While the effects of mindfulness are increasingly explored across different fields, little is known about the application of these practices in agile project teams. In this paper we report on a rigorous controlled trial executed to understand the impact of the three minute breathing exercise on the perceived effectiveness of stand-up meetings. We compare (1) an active group using a three minute breathing exercise, to (2) a placebo, and (3) a control group in 3 organizations and 8 teams with over 152 measurements. Our findings indicate an immediate positive impact on perceived effectiveness, decision-making and improved listening in the active groups compared to the placebo and natural history groups. We provide a preliminary agenda for future research based on our findings and previous evidence from other fields.

Keywords: Empirical study · Mindfulness · Scrum · Teamwork · Resilience · Agile software development

1 Introduction

In a world led by 'volatility, uncertainty, complexity and ambiguity', depending solely on automatic pilots can have disastrous effects on human life [1]. Present-day organizations are facing the same problem as they are operating in a highly unpredictable and stressful environment to which they daily need to respond adequately. It is difficult for organizations to adapt to changing circumstances and demands in a highly volatile world. Carefully crafted plans, that should work like business or project auto-pilots, are met by a stubborn reality that does not fit the envisioned strategy. Such an increase in speed and uncertainty leads to an increase in stress for teams and management [2,3]. As a consequence people fall back on autopilot behavior with suboptimal results.

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 103–118, 2017.
DOI: 10.1007/978-3-319-57633-6_7

This is a problem because organizations that do not possess the agility to reply to the present and its changed demands, run a great risk of becoming obsolete or at least lose some of their striking power within the market that they operate. Big corporations like Atari, Kodak, DeLorean, Polaroid, Pan Am and Compaq, once cutting edge businesses, have failed to meet these changing demands and showed no signs of agility, which eventually led to their demise. Their business auto-pilot was focused on a fixed point and failed to prevent them from crashing into new competitors, new technologies, new demands and waning public interest at the next junction. Companies that cannot alter their course because they cannot recognize the changes in the market, will fail or decline. Their employees will likely have to deal with stress levels that keep on building up in their system with a great chance of burnout and demoralization.

Mindfulness, a concept increasingly popular in practice, promises relief to some of those symptoms. Mindfulness deals with a certain attitude towards reality in which the practitioner approaches the here-and-now in 'the fullest attention to whatever the moment presents' [4]. Mindfulness provides tools to increase attention and aims to create habits of mind that lead to stress reduction [5]. While there is a firm evidence base of mindfulness in clinical psychology, research on the application of these practices in the context of professional organizations such as agile teams, is still in its infancy.

In this report we present the first empirical perspective on the application of a very concrete mindfulness practice in agile teams: the three minute breathing exercise. While previous studies predominantly conducted in the field of clinical psychology only revealed results after several weeks, our findings point at an immediate effect of the exercise in a subsequent meeting. Based on our experiences we draw out an agenda for further research. Our findings provide a strong base for further exploration relevant for both research and practice.

2 Background and Related Work

While the debate on the definition of mindfulness is ongoing, it's roots can be traced to Buddhist psychology where it has been practiced for several millennia [6]. The concept has then been introduced in the field of contemporary psychology by Jon Kabat-Zinn in the mid-1980s, as 'paying attention in a particular way, on purpose, in the present moment, and nonjudgmentally' [7,8]. Since then mindfulness has been applied in many fields such as education [9], law [10], "prison programs" [11], "IT" [12], and "business" [13] to stimulate more positive responses and better-decision making. While there is a growing base of evidence that specific mindfulness practices can have a positive effect on human behaviour, little is known on its impact in professional organizations such as agile teams.

In the following subsections we will discuss existing evidence of mindfulness practices applied in clinical psychology, organizational psychology, information systems, management research, and lastly in agile teams.

Mindfulness in Clinical Psychology. Several therapies and trainings have been developed to execute mindfulness based interventions. Kabat-Zinn for example introduced Mindfulness-Based Stress Reduction (MBSR). This treatment was originally designed to "treat patients with chronic pain" [8]. Eighteen known studies have been undertaken toward fathoming the consequences of MBSR on different groups of participants [14]. All of the research indicates that there is a positive correlation between MBSR and psychological well-being. Shapiro, Schwarz and Bonner for example have conducted a study among medical students, wanting to find out if the students would be able to cope better with stress after they had gone through an official MBSR program [5]. The results indicate that participation in a mindfulness-based stress reduction intervention can effectively (1) reduce self-reports of overall psychological distress including depression, (2) reduce self-reported state and trait anxiety and (3) increase scores on overall empathy levels [5]. Studies have shown that mindfulness has a general positive impact on one's psychological health [7]. Mindfulness has been correlated to a myriad of positive effects on people with psychological issues. Good results have been shown in the areas of "self-esteem" [15], "self-efficacy" [16], "clarity" [17], "self-compassion and empathy" [18]. The correlation of mindfulness has also been associated with the reduction of "depression" [19] and "stress" [20].

Mindfulness in Organizational Psychology, Information Systems and Management Research. Several randomized controlled trials have been undertaken to prove the effectiveness of mindfulness in a business setting. In an integrative review Good et al. [21] integrate the impact of mindfulness into five areas of basic functioning (attention, cognition, emotion, behavior, and physiology) and into three clusters of workplace outcomes (performance, relationships, and well-being).

Reb et al. [22] for example examined the effect of "leader's mindfulness on employee well-being and performance". The study showed that the higher the supervisor's mindfulness: (1) the higher the employees' psychological need satisfaction, (2) the higher the job satisfaction of the employee, (3) the more favorable overall job performance ratings, (4) the higher the in-role performance, and (5) the higher the engagement with organizational citizenship behaviors. Other randomized controlled trials in this area have also shown a positive correlation between the trait mindfulness and psychological well-being, better decision-making and better handling of stress [23].

Hafenbrack et al. [23] discuss the association with a mindfulness condition towards (1) positive emotions, (2) focus on the present, and (3) better decision-making. Mindfulness practices are associated with an augmentation of a positive emotional state of being, since mindfulness "increases the willingness to tolerate uncomfortable emotions and sensations" [24] which indirectly increases the quality of decision making [25]. There is a significant direct correlation between the mindfulness state and decision making [23]. Lastly mindfulness has a focus on the present [8,23] which indirectly increases the value of decision making.

Mindfulness, Agility and Agile Teams: Initial work on agile teams and well-being indicates that teams that feel more empowered experience less stress [26]. However, while the popularity of agile methods is continuously rising, establishing the right team atmosphere and leadership approach remains a challenge [27,28]. Especially in situations of increased speed and competition, agile teams are experimenting with practices to counter the loss of focus [29].

Mindfulness, while promising relief to some of the aforementioned symptoms, has so far received little attention in the context of agile methods. In existing literature the concept has been explored in two main directions in relation to agility: (1) Mindfulness as an organizational condition and a theoretical concept that supports agility through attention to detail and reliability of systems (compare [30]), and (2) Mindfulness practices as a set of tools to achieve it.

Mindfulness as a theoretical concept to support agility in organizations has been explored by McAvoy et al. [30] to compare *'Doing' Agile* vs. *'Being' Agile* - thus understanding the effectiveness of agile practices in organizational contexts. Nagle et al. [31] utilize a mindfulness measure to understand how an organization can achieve flexibility and reliability in the context of Global Software Development (GSD).

The interaction of concrete mindfulness practices and agile practices is far less well understood. Agile practices such as stand-up meetings for team coordination [32], Iteration Reviews for continuous customer feedback, or Retrospectives for teams to reflect and improve their ways of working, are concrete routines that help teams to deliver their products and improve. Mindfulness practices, similarly to agile practices, provide very specific patterns of action and reproducible protocols, routines that can help build mindful behaviour in organizations [33]. For example, Bernárdez et al. [12] conducted an experiment comparing groups of students conducting a mindfulness exercise to a control group practicing public speaking, with the former being more efficient in developing conceptual models.

Following evidence across various fields we know that mindfulness exercise can have a positive impact on decision-making, the ability to focus and psychological well-being. However, until now little is known on the impact of those exercises in business settings, especially in agile project teams. The three minute breathing space exercise [34], for example, is a concise mindfulness exercise that can be applied relatively easy in teams with little investment. The participant approaches the short exercise with an attitude of alertness and curiosity throughout its three stages of 'becoming aware', 'focusing attention on breathing' and 'extending the attention' [34]. Similarly to meeting routines in agile teams, such as stand-up meetings, it provides a concrete and convenient protocol. As such, the two practices, stand-up meeting and the breathing exercise, can be combined into an experiment.

Based on the literature reviewed above we thus pose the following question: *What is the effect of the three minute breathing space exercise on the quality of meetings in an agile project team?*

Table 1. Stand-up meeting protocol for the three trial groups

Step	Duration	Activity			Actor(s)
		Active	Placebo	Control	
1	5 min	Execute the three minute breathing space	Listen to Tango by Igor Stravinsky	-	Facilitator & Teams
2	15 min	Participate in Stand-up meeting			Teams
3	5 min	Fill out questionnaire			Teams
4	1 min	Collect questionnaires			Facilitator

3 Research Method and Conduct

Following the research question this paper aims to help understand the impact of a specific mindfulness practice, the three minute breathing space, applied in agile project teams. As the three minute breathing exercise as well as the Scrum stand-up meetings provides reproducible and comparable routines, we embedded our research question in an experiment following the design of a controlled trial as common in clinical settings [35,36]. As the trial is executed in a social context with many interconnected factors such as teamwork, process, culture and the perceptions of individuals, we applied a mixed methods approach using quantitative and qualitative sources to analyse the data [37].

Protocol: The trial was divided into three phases, a (1) preparatory phase from April until May 2016, three organizations were asked to join and facilitators were instructed, (2) collection of a baseline measurement in the beginning of June 2016, and (3) the actual trial period lasting from mid-June until mid-July 2016.

In order to reduce bias, we designed a controlled trial including a placebo as well as a natural history control group to compare the effect of the mindfulness exercise. To do so, we created a trial protocol including three groups, (1) an active group with teams executing the breathing exercise before their meetings, (2) a placebo group, which would listen to classical music by composer Igor Stravinsky, and (3) a control group. In order to distract attention from the actual mindfulness exercise, the study was strictly framed as an *"experiment to increase effectiveness in Scrum Meetings"* across participants and supporting facilitators. The placebo[1] group was added to compare the impact to a non-meditative form of relaxation, which could have an impact on the team, and

[1] We are aware that similarly to trials in social therapy, there is no placebo for an intervention in a social environment, as even a trivial interaction across individuals does have an impact [36]. For the sake of simplicity we still call the second trial group as "placebo" although it is technically not the case.

to further remove attention from the mindfulness exercise. All data collection was kept strictly anonymous and we repeatedly asked the teams to give honest opinions.

We chose stand-up meetings as the agile practice the trial was aligned to, also referred to as "Daily Scrum". We chose that specific meeting type due to frequency, commonly accepted format and contribution to decision-making within the team [32]. The meetings are short in nature and strictly time limited. The team members address the three questions *"What have I done? What will be done? What obstacles are in my way?"* and make operational decisions [32].

The interventions for the three trial groups were designed as depicted in Table 1. For the active and placebo groups a guided 5-minute exercise was given just before the start of the stand-up meeting, the natural history control group had no exercise whatsoever. The mindfulness breathing exercises (for a protocol compare [34]) as well as the Stravinsky[2] placebo exercise were both guided by experienced mindfulness instructors to give the best results. The breathing exercise was chosen due to its short nature, accessibility and prior exploration in the context of software teams [12].

The instructors were present 5 min before the meeting started and conducted the exercise type that was assigned to the team. After the exercise had taken place the team would start with its meeting. Shortly thereafter the team would fill out the forms. The natural history control group (nh) was not guided at all, but needed to fill out the forms at exactly the same moments as the other teams to follow their heartbeat. The procedure was repeated for the active and placebo groups four times until the end of the trial. Due to different iteration lengths, and to have sufficient time between the measurements to ensure that the interventions themselves would not influence each other because of too short an interval between exercises, the measurements took place once per week.

Organizations, Teams and Participants: Between April 2016 and May 2016 we reached out to organizations with software development departments in the Netherlands. The selection criteria was to find organizations with at least three software development teams applying Scrum for a period of at least three years.

Table 2. Distribution of the three trial groups (active, placebo, control) across the three participating organizations and involved teams

Organisation	Alpha			Beta			Gamma	
Team	T1	T2	T3	T4	T5	T6	T7	T8
Trial group	Active	Placebo	Control	Active	Placebo	Control	Active	Control
Team size	5	8	7	8	10	10	5	8
Measurements	4	4	1	4	4	4	4	2
Total responses	32	13	7	24	24	19	19	14

[2] "Igor Stravinsky - Tango (audio + sheet music)", URL to the video: https://www.youtube.com/watch?v=VcXTFRXenwI.

This ensures that these teams are working with short cyclical iterations in which working software is completed after each sprint, and applying stand-up meetings. Out of the 10 inquired organizations, three organizations and a total of 8 teams agreed to participate. After gaining the commitment of the teams, we assigned them to one of the three trial groups as depicted in Table 2. Organisation Gamma originally included a placebo team as well, however, the team dropped out due to internal deadlines before the trail execution. At last there were 8 teams included in the trial and respective analysis. Furthermore, seven facilitators were instructed to conduct the respective exercises and collect the data on-site.

Questionnaire Design and Data Collection: Following our literature study we compiled a questionnaire based on two dimensions: mindfulness and effectiveness. The questions can be found in Table 3. The questions addressing mindfulness (Q03, Q05, Q07, Q08, Q09, Q10) have been selected based on the dimensions mindfulness has been reported to have an impact on, such as: improved decision-making, better emotional responses, focus on the present [23]. In addition to that we added questions on effectiveness of the meeting (Q01, Q02, Q04, Q06). The questions have been administered with a 7-point Likert scale: *1 = Never, 2 = Rarely, 3 = Sometimes but infrequently, 4 = Neutral, 5 = Sometimes, 6 = Usually, 7 = Always*

Before the actual trial we conducted a baseline measurement which would later serve as a base for comparison. The baseline measurement was collected at the beginning of June 2016, the actual trial followed in mid-June 2016. During the trial team members were asked to fill out the questionnaire directly after the meeting and respective intervention (compare Table 1). In order to have sufficient time between the measurements, the exercises were conducted and data was collected once per week across the participating teams. After each allocated meeting, being three stand-up meetings per team, the stated items were graded by each team participant and were handed over to the facilitators. The time frame in which these measurements took place is from May 30th until July 25th of the year 2016. The facilitators made sure that the forms were then forwarded to the researcher.

At the end of the trial we asked the participants that took part in both the active and the placebo group to answer a number of open questions to get a more qualitative view on their perceptions. The questions were: *How valuable did you find this exercise? Would you continue this exercise without the trainers? What are the challenges you had? What worked well?*. For this qualitative view we used a deductive and exploratory approach in order to understand whether the personal perceptions of the participants would confirm or refute the quantitative analysis.

Data Analysis: The data generated was analyzed by question and by preparation type, i.e. the baseline of each question of each preparation type was aggregated and compared to the figures that were the result of the actual

Table 3. Questions Q01–Q10

• Q01 - Everyone is involved in the decision-making process.
• Q02 - The team vision was well defined.
• Q03 - The meeting atmosphere was constructive, calm and open
• Q04 - The meeting was effective
• Q05 - All meeting participants listened well to each other
• Q06 - The meeting objectives were met
• Q07 - The level of disagreement during the meeting was acceptable
• Q08 - The tension during the meeting was tolerable
• Q09 - The interaction in the meeting was good
• Q10 - The emotional responses within the meeting were healthy

measurements that were taken after the experiments had been conducted. With that aggregation level a t-test was executed on the difference between the baseline and the experiment per preparation type, finding the difference in average scores on all questions and the significance value (the p.value) of all these differences indicating if the difference could be explained through the intervention itself. The significance value we sought was a p-value $< 0{,}05$.

Besides the differences in average per question given the preparation type, we also took an average on the aggregated sum of the questions per team and tried to identify the maturity of the team. Furthermore the variance of all questions per team was measured to ensure the homogeneity of the given answers per team. To control for any unexpected influences T-values were measured.

4 Results

This section presents the results of the experiment. Table 2 depicts the participating teams, the respective team size, the number of measurement points, as well as the number of completed questionnaires. Every team consisted of approximately eight members. Each team, with the exception of the control groups, had four measurement moments. Those consisted of one baseline to measure the effectiveness and culture of the team before any intervention was provided and three guided measuring moments.

Table 4 summarizes the results for the ten questions (Q01–Q10) for the three trial groups. As depicted in the table teams that submitted themselves to the mindfulness exercise showed a slight but statistically significant ($p < 0.05$) increase in some key elements of effectiveness and cultural aspects of the team. Specifically our data indicates an improvement on the perception of (1) listening, (2) decision-making, (3) effectiveness of the meeting, (4) good interaction and (5) healthiness of emotional responses. Neither the placebo nor the natural history control groups showed statistically significant differences.

Table 4. Difference to baseline measurement for questions Q01–Q10 (Total n = 152)

Question/Trial group	Active (n = 75)	Placebo (n = 37)	Control (n = 40)
Q01 Decision-making	0.6659*	0.1666	0.1190
Q02 Team-vision well defined	0.2513	−0.4666	0.2857
Q03 Atmosphere constructive	0.3170	0.4	0.0238
Q04 Meeting effective	0.6139*	0.1333	0.2142
Q05 Listening	0.6299**	0.0666	0.4285
Q06 Objectives met	0.2905	0.2666	0.2857
Q07 Disagreement acceptable	0.3276	0.1000	−0.0238
Q08 Tension tolerable	0.3382	−0.0666	−0.1190
Q09 Interaction good	0.5673*	0.1333	0.0000
Q10 Emotional responses	0.4178*	0.2333	0.4333

* $p < 0.05$, ** $p < 0.01$

5 Discussion

The main query of this paper is whether a short mindfulness intervention has an impact on the effectiveness and culture in stand-up meetings of agile development teams. In the following subsections we will discuss (1) the perceptions of the teams with respect to our research question, (2) the embedding of the exercise in a broader organizational setting and barriers to its adoption, and (3) directions for future research.

5.1 Three Minute Breathing Exercise in Agile Teams, Does It Work?

The trial shows that even short mindfulness exercises, such as the here presented three minute exercise have a positive impact on the teams similarly to those reported in other domains (compare Table 4). The data indicates a self-reported improvement along five of the ten questionnaire items, particularly: (1) participants listened well to each other, (2) Everyone is involved in the decision-making process, (3) the meeting was effective, (4) the interaction in the meeting was good, and (5) the emotional responses within the meeting were healthy. The questions with the biggest difference to the baseline were *Q01 Everyone is involved in the decision-making process*, and *Q05 participants listened well to each other*. These perceptions were supported by the qualitative data (n = 14), e.g.: *"The 3 min of silence helped me rest and relax. It helped gather my senses back after a few hours of (usually) stressful work."* (Participant Team 4). We did not observe any statistically significant negative effects in our data. In the placebo group the question Q02 had a statistically not significant decrease compared to the baseline measurement. Here we could raise the question if the Stravinky song had a distracting effect on the team and its vision during the meeting. Looking back at our research question we will now lead the discussion in

two ways: (1) how the exercise can support building emotional intelligence and leadership skills in the individual, and (2) how the exercise can help building mindful teams.

Taking the perspective of the individual our findings indicate that the breathing exercise could help agile team members and team leaders to build up their emotional leadership skills. As pointed out by Porthouse and Dulewicz [38] emotional leadership competencies (e.g., emotional resilience, sensitivity, self-awareness, conscientiousness) are of greater importance for leaders in agile projects compared to traditional projects. As the leadership skills and style of individual managers have a big impact on the culture of an organization, emotional leadership skills are important for the success of agile methods. A meta-analysis conducted by Giluk [39] on the relationship between mindfulness and the Big Five personality traits shows relationships with neuroticism, negative affect, and conscientiousness, but also with agreeableness.

Taking the perspective of the team our findings indicate that the practice could help building agile teams. Self-managing teams are considered to be one of the corner stones of agility, yet they are difficult to establish [28]. The five dimensions of agile teamwork, such as shared leadership, team orientation, redundancy, learning and autonomy [28,40] require shared decision making and the ability to listen to each other and understand each others opinions, as supported by the breathing exercise. Further, similarly to what McAvoy et al. [30] call *'Doing' Agile* vs. *'Being' Agile*, our experiences with the trial indicate that the exercise could help build up mindful behaviour, which helps the team understand agility and agile practices in context rather than blindly following them. The lack of focus can be an issue for agile and entrepreneurial teams [29]. Hafenbrack et al. [23] researched the positive influence of mindfulness on decision making and the sunk-cost bias, the tendency to continue investing in a project once time, money or effort was invested, although that project might not be a viable initiative after all. Stettina and Smit [29] researched agile teams working in entrepreneurial settings. The results reveal that when trying to handle many project requests due to customer pressure, mindfulness could help making better decisions on what projects to follow.

5.2 Mindfulness in Our Case Organisations: Barriers to Adoption

Our quantitative results show that mindfulness enhances qualities of effectiveness and team cooperation in the daily working culture of an agile team. The qualitative open questionnaires distributed to the teams after the trial, however, draw another perspective on our findings. While several participants saw the personal use of the exercise, none would continue it in a public setting. As a participant from Team 2 commented: *"For some members, the pause before the standup was useful, because they could focus on their activities done in the previous day. But for the rest of the team, the exercise was considered just not suitable with their own way of working."* Several participants in Team 4, for example, indicated that the fact that they conducted the exercises in an open space, they felt looked at by other teams. Others indicated, they would continue

with the breathing exercise on their own rather than in the team setting: *"Yes, I want to do those exercises more often. I have chosen to do this at home and not at work."* (Participant Team 7). So, although the results show statistically significant increases of effectiveness on several entries, the perceived usefulness does not raise to the level that the participants want to keep on using it in a public setting. The teams apparently encountered a barrier to introduction of these practices.

This raises the more general question of what conditions could support the adoption of these type of mental practices in agile teams. From the literature (cf. [21]) we know a few: support of management for these practices, voluntary participation and a safe team climate. Management support for these practices seems obvious: if leaders do not support these practices it will not happen. In that respect these mental practices do not differ from other agile team practices that help teams perform better. As a line of research, this would be interesting to look into.

Voluntary participation is a necessary corollary of these type of practices. It enhances intrinsic motivation, which is an important mediator of success of team practices (cf. [41,42]). Lastly, also a safe team climate is important. If, as the qualitative data examples showed, people feel exposed, the practices will not function very well. That is a general factor for well-functioning teams: psychological safety is a crucial characteristic of successful teams. If such a climate is absent, social defense mechanisms will come into play and diminish team performance. Safety has both an environmental side (what space is the team working of meeting in, open or closed) and a communicative side: do people feel safe to utter difficulties, ask questions, disagree, praise each other, etc. In general it means that within the team culture or the organization, it is recognized that emotions play a role and are not subdued. It is generally known from psychological research into emotional agility that if this happens, they will play out in a different but uncontrolled way with mostly negative effects on team climate and effectiveness.

5.3 Mindfulness in Agile Project Teams: A Preliminary Research Agenda

Having studied the results of a mindfulness intervention in agile teams and discussed its relation to existing literature, we now continue to discuss a potential future research agenda. The following is a thematic list of questions, not aiming to be exhaustive, but as possible entry points for an exploration of mindfulness in agile teams:

Effects on leadership competencies and team development. From Porthouse and Dulewicz [38] we know that emotional leadership competencies are more important in agile project teams compared to traditional project teams. Also shared leadership is an integral aspect of agile teams and can be difficult to acquire [28]. *How does mindfulness influence the development of leadership competencies and emotional intelligence? What role do mindfulness practices play in team development?*

Effects on decisions. From Hafenbrack et al. [23] we know that meditation practices are reducing sunk-cost bias. *What types of decisions do mindfulness practices have an impact on?*

Lengths of training and lengths of effect. In this trial we worked with a brief mindfulness exercise at the beginning of a short agile meeting. We did not, however, measure the impact of this short exercise on a longer type of meeting. It could be that the enhancing effect wears off quickly and that for longer meetings the exercise needs to be repeated several times in order to gain its lasting effect. Also, in clinical research, experiments have been more intense in nature. It would be interesting to see if a whole team that volunteers to submit to a whole intensive program will see even better results. *Do longer, more intense mindfulness exercises have greater impact on agile teams? Do short mindfulness exercises also have an impact on longer agile meetings? Do short mindfulness practices become increasingly more effective over time?*

Implementation. Although teams indicated that they benefited from the mindfulness exercise they also communicated that they did not want to continue the exercise once the experiment ended. This is an interesting observation which has a contradicting tension. It would be interesting to find out why we were confronted with this tension. *What is the best possible organizational culture in which mindfulness will thrive? What is the correlation between the effectiveness of a mindfulness exercise and the maturity of a team? If the teams are to sustain such a practice on their own, how would they teach to new team members? And if they have to teach the practice to new members, will it be as good as they have learned is form a mindfulness teacher?*

Interaction with other practices and routines. In this paper, we have only focused on stand-up meetings during this experiment. Future research can broaden the scope and could determine if there is a correlation between the trait mindfulness and the effectiveness of other types of Agile meetings like retrospectives, sprint planning, sprint review or refinements. It would be interesting to find the effect of other types of mindfulness exercises on the effectiveness of team meetings in Agile teams. *What is the effect of other expressions of mindfulness exercises on the effectiveness of meetings in agile project organisations? What is the effect of a mindfulness exercise on other type of meetings in an Agile project organisation?*

Types of teams and domains of practice. Our research has focused on software development teams, it would be interesting to expand our understanding towards other domains of practice. We have seen that the trait mindfulness helps make better decisions and is an enabler for the handling of stress. Some types of teams might benefit even more from exercises in the mindfulness spectrum. Teams that are dealing with higher stress levels than software teams or teams that have an acute need for clear and effective decisions would potentially be better candidates in this regards. Portfolio management teams, innovation teams or board room teams would be suitable candidates to consider. *What type of teams benefits most from the trait mindfulness?*

Costs vs. Benefits. Understanding the costs of a potential implementation is important for management. Hales et al. [43] discuss the costs of implementing mindfulness in a health care context. *What are the costs of implementing mindfulness in project organizations compared to their benefits?*

5.4 Threats to Validity

A controlled trial executed within eight teams in three organizations can be more of a challenge to set up in the operational phase than when designed on paper. To avoid potential sources of bias, we followed the recommendations of Pannucci et al. [44] to prevent bias in clinical trials across stages of research in the planning, data collection, analysis, and publication.

In the pre-trail phase study design and in recruitment selecting a favourable population could impact study results. We addressed selection bias by masking the study purpose. During trail execution, the facilitators educated mindfulness trainers, could have consciously or subconsciously influenced the responses of the team members which could result in higher scores for the treatment teams. We used standardized protocols for execution, data collection and carefully instructed the facilitators, reiterating that masking the study purpose is important for its outcomes. Further, participants might be prone to please the experiment leader and give him the answers he needs for his experiment to be successful. Due to masking the purpose, the participants were not aware of the actual study purpose. Another potential source of bias could be the concept of the breathing exercise, which could polarize some of the participant. Potential skepticism could influence the answers of the participants, provoking interest and random answers. We have tried to notice this within the data set but did not find statistically relevant outliers or noise in the data. In the post-trial phase, bias can occur during data analysis and publication. To address external validity, we compared our findings to existing evidence in the fields of clinical psychology [14] and in professional organizations [21]. To further improve construct validity we applied a mixed methods approach in collecting and data analysis using qualitative and quantitative sources.

6 Conclusions

The goal of this study was to explore the impact of a short mindfulness exercise on the quality and effectiveness of meetings in agile project teams. A controlled trial was designed to observe effects associated with mindfulness in the context of eight Scrum teams in three organizations.

The participants perceived the practice as useful, and statistically significant improvement was reported on some of the dimensions in the groups performing the exercise (listening, decision-making, meeting effectiveness, interaction, emotional responses). The teams in our case organizations will not continue with the exercise in their particular setting. Nonetheless, the result is quite remarkable as the trial shows an instant effect while other studies had a preparation phase of

several weeks or more. Further research needs to be done in order to understand the circumstances under which its effects are perceived more or less. If there is more collaboration and more pressure in future business settings to keep our organizations healthy, sustainable and effective, the use of mindfulness might be more essential. To do so we provide concrete ideas for a research agenda to explore the effects further.

The conclusion that we can draw is that mindfulness in the form of breathing exercises indeed enhances the quality of meetings in an agile team. Research indicates the increasing importance of emotional intelligence and empathy to be for future workforce next to analytical skills. Practices such as the here discussed exercise could help build up some of those skills in the future.

Acknowledgments. We thank all the participating organizations, teams and facilitators for generously contributing to this study.

References

1. Inge, J.: Safe data: Recognising the issue. Safety Syst. **21**(1), 4–7 (2011)
2. Bodensteiner, W.D., Gerloff, E.A., Quick, J.C.: Uncertainty and stress in an r&d project environment. R&D Manag. **19**(4), 309–322 (1989)
3. Ashford, S.J.: Individual strategies for coping with stress during organizational transitions. J. Appl. Behav. Sci. **24**(1), 19–36 (1988)
4. Brown, K.W., Ryan, R.M.: The benefits of being present: mindfulness and its role in psychological well-being. J. Pers. Soc. Psychol. **84**(4), 822 (2003)
5. Shapiro, S.L., Schwartz, G.E., Bonner, G.: Effects of mindfulness-based stress reduction on medical and premedical students. J. Behav. Med. **21**(6), 581–599 (1998)
6. Kohls, N., Sauer, S., Walach, H.: Facets of mindfulness-results of an online study investigating the freiburg mindfulness inventory. Pers. Individ. Differ. **46**(2), 224–230 (2009)
7. Kabat-Zinn, J.: An outpatient program in behavioral medicine for chronic pain patients based on the practice of mindfulness meditation: Theoretical considerations and preliminary results. Gen. Hosp. Psychiatry **4**(1), 33–47 (1982)
8. Kabat-Zinn, J.: Full catastrophe living: using the wisdom of your body and mind to face stress, pain, and illness. Dell Pub. A division of Bantam Doubleday Dell Pub. Group, New York (1991)
9. Hyland, T.: Mindfulness and the therapeutic function of education. J. Philos. Educ. **43**(1), 119–131 (2009)
10. Rogers, S.L.: Mindfulness for Law Students: Using the Power of Mindful Awareness to Achieve Balance and Success in Law School. Mindful Living Press, Miami Beach (2009)
11. Vengapally, M.: Preparing to Leave Prison: A Mindfulness-based Intervention to Reduce Recidivism (2014)
12. Bernárdez, B., Durán, A., Parejo, J.A., et al.: A controlled experiment to evaluate the effects of mindfulness in software engineering. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, p. 17. ACM (2014)

13. Reb, J., Atkins, P.W.: Mindfulness in Organizations: Foundations, Research, and Applications. Cambridge University Press, Cambridge (2015)
14. Keng, S.L., Smoski, M.J., Robins, C.J.: Effects of mindfulness on psychological health: a review of empirical studies. Clin. Psychol. Rev. **31**(6), 1041–1056 (2011)
15. Ward, D.: Overcoming Low Self-Esteem with Mindfulness. SPCK Publishing, London (2015)
16. Shapiro, S.L.: The integration of mindfulness and psychology. J. Clin. Psychol. **65**(6), 555–560 (2009)
17. Moffitt, P.: Emotional Chaos to Clarity: Move from the Chaos of the Reactive Mind to the Clarity of the Responsive Mind. Penguin Publishing Group (2012)
18. Kingsbury, E.: The Relationship Between Empathy and Mindfulness: Understanding the Role of Self-compassion. Alliant International University, San Diego (2009)
19. Segal, Z.V., Williams, J.M.G., Teasdale, J.D.: Mindfulness-Based Cognitive Therapy for Depression. Guilford Press, New York (2012)
20. Stahl, B., Goldstein, E.: A Mindfulness-Based Stress Reduction Workbook. New Harbinger Publications, Oakland (2010)
21. Good, D.J., Lyddy, C.J., Glomb, T.M., Bono, J.E., Brown, K.W., Duffy, M.K., Baer, R.A., Brewer, J.A., Lazar, S.W.: Contemplating mindfulness at work an integrative review. J. Manag. **42**(1), 114–142 (2015). 0149206315617003
22. Reb, J., Narayanan, J., Chaturvedi, S.: Leading mindfully: two studies on the influence of supervisor trait mindfulness on employee well-being and performance. Mindfulness **5**(1), 36–45 (2014)
23. Hafenbrack, A.C., Kinias, Z., Barsade, S.G.: Debiasing the mind through meditation mindfulness and the sunk-cost bias. Psychol. Sci. **25**(2), 369–376 (2014)
24. Arch, J.J., Craske, M.G.: Mechanisms of mindfulness: emotion regulation following a focused breathing induction. Behav. Res. Ther. **44**(12), 1849–1858 (2006)
25. Loewenstein, G., Lerner, J.S.: The role of affect in decision making (2003)
26. Laanti, M.: Agile and wellbeing-stress, empowerment, and performance in scrum and kanban teams. In: 2013 46th Hawaii International Conference on System Sciences (HICSS), pp. 4761–4770. IEEE (2013)
27. Moe, N.B., Dingsøyr, T., Dybå, T.: A teamwork model for understanding an agile team: a case study of a scrum project. Inf. Softw. Technol. **52**(5), 480–491 (2010)
28. Stettina, C.J., Heijstek, W.: Five agile factors: helping self-management to self-reflect. In: O'Connor, R.V., Pries-Heje, J., Messnarz, R. (eds.) EuroSPI 2011. CCIS, vol. 172, pp. 84–96. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22206-1_8
29. Stettina, C.J., Smit, M.N.W.: Team portfolio scrum: an action research on multitasking in multi-project scrum teams. In: Sharp, H., Hall, T. (eds.) XP 2016. LNBIP, vol. 251, pp. 79–91. Springer, Cham (2016). doi:10.1007/978-3-319-33515-5_7
30. McAvoy, J., Nagle, T., Sammon, D.: Using mindfulness to examine ISD agility. Inf. Syst. J. **23**(2), 155–172 (2013)
31. Nagle, T., McAvoy, J., Sammon, D.: Utilising mindfulness to analyse agile global software development. In: ECIS (2011)
32. Stray, V.G., Moe, N.B., Aurum, A.: Investigating daily team meetings in agile software projects. In: 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, pp. 274–281. IEEE (2012)
33. Jordan, S., Messner, M., Becker, A.: Reflection and mindfulness in organizations: rationales and possibilities for integration. Manag. Learn. **40**(4), 465–473 (2009)
34. Koole, W.: Mindful Leadership: Effective Tools to Help you Focus and Succeed. Warden Press, Amsterdam (2014)

35. Pocock, S.J.: Clinical Trials: A Practical Approach. Wiley, Hoboken (2013)
36. Leff, J.: Clinical and methodological problems in interaction studies. In: Epidemiological Impact of Psychotropic Drugs. Elsevier, Amsterdam (1981)
37. Miles, M., Huberman, A.: Qualitative Data Analysis: An Expanded Sourcebook, 2nd edn. Sage, Thousand Oaks (1994)
38. Porthouse, M., Dulewicz, V.: Agile Project Managers' Leadership Competencies. Henley Management College (2007)
39. Giluk, T.L.: Mindfulness, big five personality, and affect: a meta-analysis. Person. Individ. Differ. **47**(8), 805–811 (2009)
40. Salas, E., Sims, D.E., Burke, C.S.: Is there a big "five" in teamwork? Small Group Res. **36**(5), 555–599 (2005)
41. Bain, A.: Social defenses against organizational learning. Hum. Relat. **51**(3), 413–429 (1998)
42. Goto-Jones, C.: Zombie apocalypse as mindfulness manifesto (after žižek). Postmodern Cult. **24**(1) (2013)
43. Hales, D.N., Kroes, J., Chen, Y., Kang, K.W.D.: The cost of mindfulness: A case study. J. Bus. Res. **65**(4), 570–578 (2012)
44. Pannucci, C.J., Wilkins, E.G.: Identifying and avoiding bias in research. Plast. Reconstr. Surg. **126**(2), 619 (2010)

Enhancing Agile Team Collaboration Through the Use of Large Digital Multi-touch Cardwalls

Martin Kropp[1(✉)], Craig Anslow[2], Magdalena Mateescu[1], Roger Burkhard[1], Dario Vischi[1], and Carmen Zahn[1]

[1] University of Applied Sciences Northwestern Switzerland, Windisch, Switzerland
{martin.kropp,magdalena.mateescu}@fhnw.ch
[2] School of Engineering and Computer Science,
Victoria University of Wellington, Wellington, New Zealand
craig@ecs.vuw.ac.nz

Abstract. Agile software development has become mainstream, and with it many tools have been developed to support Agile software development. Nonetheless, studies show, that most Agile software teams still also use physical cardboards for their daily work. This is error prone and causes a lot of extra effort to keep both in sync. In our research project we conducted an interview study about the reasons for this media break. Based on the findings we developed visualization and interaction concepts for an Agile cardwall using an extra-large multi-touch wall display which provides Agile teams the lightweight collaboration workspace for their Agile meetings. We implemented the concepts in the software prototype *aWall*, and evaluated the usability of aWall in a user study. The evaluation indicates that aWall enables and encourages team work due to the large size of the wall, the easy accessibility and visibility of the needed information, and the integration with existing issue tracking tools. This suggests that augmenting digital cardwalls with large interactive touch technology and integration with task tracking systems is a useful way to support effective collaborative Agile software development processes.

Keywords: Agile software development · Cardwalls · Large wall displays · Multi-touch · Tool · Software processes · Collaboration

1 Introduction

In Agile software development, physical cardwalls continue to be an essential part of the Agile processes despite the relative large number of available digital tools. Although many commercial and open source digital Agile tools like JIRA [3], CA Agile Central (formerly Rally) [16] and VersionOne [12] are available and have been adopted by a large number of Agile companies, studies show that physical cardwalls are still widely used [4,7]. Azizyan et al. conducted interviews with software practitioners and found that 31% of companies used both project management tools and physical cardwalls, where the usage of cardwalls was not restricted to co-located teams [4]. Despite their prevalence, physical cardwalls

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 119–134, 2017.
DOI: 10.1007/978-3-319-57633-6_8

still have issues as content is not digitalized and not integrated with issue tracking systems. To address the issue with physical cardwalls, we aim to bridge the gap by creating a large digital cardwall that supports elements of the physical nature, integration with existing tracking systems, while also preserving the Agile collaborative work style.

In this paper we present *aWall*, a large digital cardwall, providing a collaborative workspace for Agile teams. While the main focus of aWall is for use by co-located teams, aWall is designed to be used also by distributed teams which is one of the main driver for digital Agile tools (see Fig. 1). aWall has the size of classical physical cardwalls by using large multi-touch high resolution displays and so provides enough space for the whole team to interactively collaborate. We first give an overview of related work, followed by an evaluation of an interview study about the usage of Agile tools in software teams. We then present the design consideration for a large scale Agile cardwall and the user interface design of aWall. The following section presents the evaluation of aWall in a user study which we conducted with software practitioners to evaluate the usability and effectiveness of the chosen approach. The paper concludes with a final summary and outlook for future work.

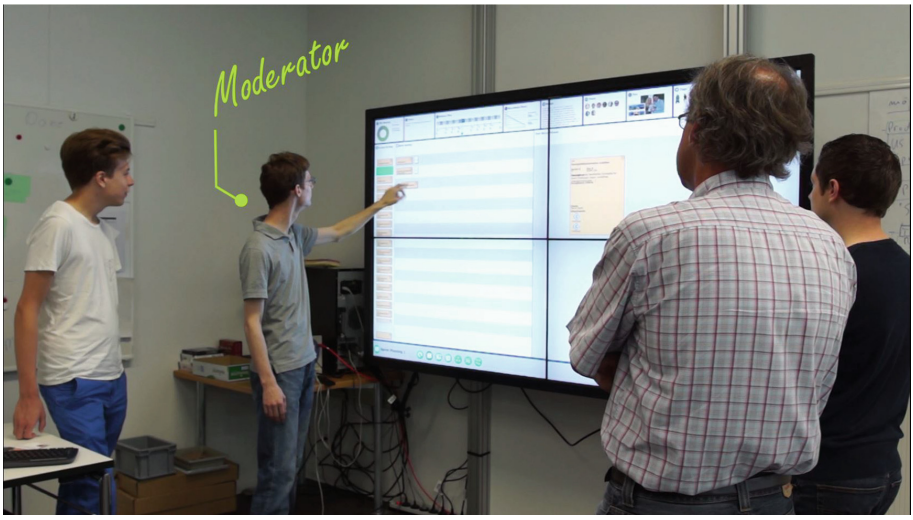

Fig. 1. aWall – digital Agile cardwall displayed on a large high resolution multi-touch wall (2×2 46 Inch 4 K displays) for planning and Agile team meetings.

2 Related Work

Sharp et al. [15] reports that physical artifacts like pin boards, sticky cards, flip charts or whiteboards are used as a means of communication and collaboration by Agile teams. However they also report that there are some disadvantages

of physical artifacts such as cards may get lost and they cannot be searched or shared easily. Any attempt to overcome these disadvantages by digitalizing cards and cardwalls should retain the advantages of the physical form while also benefiting from translation to the digital medium [15].

Gossage et al. [7] report in their study that the physical nature of artifacts is important to the collaborative process. For example being able to manipulate the cards easily (writing and posting) and their permanent availability on the cardwall helps support effective communication at least in co-located teams. Physical cardwalls are valued for their flexibility, light-weight and easy usage, providing a big picture, and permanent and instant availability of information. Physical cardwalls are not well suited for distributed environments and displaying large amounts of information is difficult. On the other side, they report that digital tools were not necessarily easy to use, hard to personalize, or to adapt to the teams' needs. They come up with suggestions on the design of digital cardwalls and with critical features: always provide an overview, offer easy zoom-and-pan, to hide details, assign annotations to any object on the board, automatic synchronization, for example.

Paredes et al. conducted a survey of existing literature on information visualization techniques used by Agile software development teams and found that information radiators and cardwalls are most frequently used for Agile teams in communication and progress tracking [13].

Azizyan et al. [4] report that digital Agile tools like JIRA and VersionOne account for less than 10% of tools used to support Agile processes, while physical walls, paper, and spreadsheets account for almost 50%. They also report that it is important to find the right balance between enough features and usability is critical.

A number of research digital tools have been developed for use on large interactive surfaces (e.g. horizontal and vertical). DAP [10] and subsequently AgilePlanner [18] were early prototypes developed to support Agile planning on horizontal tabletops for co-located teams. SmellTagger supports collaborative code reviews for co-located teams using multi-touch tabletops [11].

CodeSpace [5] does not focus on any particular Agile process but uses shared touch screens, mobile touch devices, and Kinect sensors to share information during developer meetings. They report that professional developers were positive with this approach and felt that pointing with hands or devices and forming hand postures are socially acceptable. Anslow et al. [2] evaluated large display walls for collaborative software visualization. SourceVis used large multi-touch tabletops to support code reviews using collaborative visualization techniques [1]. They show in their paper that large displays have the potential to provide a good overview about complex situations and thus can help to get a better understanding of it. Rubart developed a basic prototype for multi-touch tabletops to support Scrum meetings [14] and evaluated the prototype in a study with student groups. They report positive feedbacks with respect to team collaboration, but also difficulties with editing data. Though support of distributed teams is currently not in the focus of our work, dBoard [6] is a very interesting approach

in which a Scrum board on a vertical touch screen has been enabled with in-screen video capabilities for distributed development. This might be a possible approach for aWall to support distributed teams.

Based on our review we conclude that most digital Agile tools only partially support collaborative Agile processes and meetings. Digital Agile tools especially seems to lack support for social interaction and team cognitive activities compared with physical tools. Neither existing physical nor digital cardwall tools seem to sufficiently support the collaborative Agile process for Agile teams effectively. With aWall we present an approach to overcome these limitations. aWall tries to combine the advantages of physical and digital cardwalls, by making use of the large screen size and the touch functionality, and serve as an Agile collaborative workspace and information radiator for Agile teams.

3 Pre-study Tool Usage

3.1 Study Method

As a first step in our project we conducted a qualitative interview study among eleven IT companies that have adopted agile methods in their software development. The interviews focused on collaborative processes in agile teams and the tools they use for communication.

We conducted ten semi-structured group interviews and three individual interviews with eleven IT companies. The interviews were conducted with a total of 44 participants (7 female, mean age of all participants was 38.5 years). The participants mostly worked in multiple Agile teams and had different roles in those teams (e.g. Scrum Master, Product Owner, Team Member). The participants had at least one year experience with Agile software development, most between 2–5 years. Each group interview took about two hours and the individual interviews one hour. All interviews were conducted in German. The focus of the interviews was on the employment of agile methods and practices, team and collaborative processes, meetings and tools used. All interviews were audio recorded and transcribed. The transcribed interviews were segmented into small units of analysis and coded using MAXQDA [17]. A category system for the analysis was developed and continuously refined [9].

3.2 Findings

We found that 10 out of 11 teams still use physical cardwalls typically in combination with digital tools, like Jira/Confluence, TFS, Trello. Only one team was working exclusively with digital tools. When using physical cardwalls we found that it is a common practice to put a lot of extra information around the task board as shown in Fig. 2. The extra information includes for example, the Definition-of-Done, team members' periods of absence, burn-down charts, but also private pictures, post cards from holidays or other greeting cards from team members.

Fig. 2. A physical task board with extra information

When asking for the advantages of physical cardwalls compared to digital Agile boards the most often named reasons were ease of use, always-on visibility, flexibility, good overview, and the focused view on the information. On the disadvantage side the interviewees mainly named that the board is only locally available, the missing traceability and documentation, and the missing integration into digital tools. Table 1 lists the most often named advantages and disadvantages of digital tools by the participants.

The following statements restates opinions from some participants about various aspects about digital tools[1]:

> "Yes, it is all well integrated. If I have it electronically, I have it in the database. I have to capture it only once. That's what I like about the digital tools (I8, 305)[2]."

> "The digital board looks quite nice, is always well ordered (I11, 364)."

> "You have so many options in JIRA, so many input fields. And if you are looking for information, you always have to do a lot of navigation.(I1, 235)."

> "It takes so long to start up the tool. And after five seconds passed, I have to fill out this template. And then I just want to add a picture. I have to take a photo and somehow add it to the system. That's very cumbersome. (I10, 396)."

[1] All statements have been translated from German.
[2] The numbers refer to the interview and the line number of the transcription.

Table 1. Pros and Cons of digital agile tools compared to physical tools

Pros	Cons
Changes are stored automatically	Feeling of having no control
A lot of extra features	Too many features
Traceability	Missing visibility for others
Transparency in who did what	High effort for usage/administration
Provide some overview	Missing good overview
Adaptability of tools	High effort for customization
Access from everywhere	Not always on (have to start up)
Teams can meet in virtual rooms	Must be customized before usage
-	Have to navigate to information
-	Performance not always good
-	Displays are too small

We also asked the participants about the requirements for an ideal digital Agile cardwall. The interviewees stressed the importance of non-functional requirements. These included the need for a large size display, configurable views, instant availability of information, overview of information, at all time visible information, easy to reach context dependent information, increased readability of information, simultaneous multi-user touch interaction, direct interaction with data, and no need for navigation.

3.3 Summary

In summary the study results seem to show that available digital tools do not sufficiently well support the required flexible collaborative Agile workstyle. Users value the traceability of information in digital tools, linking possibilities of artifacts, and the flexibility to adapt the tools to the users' needs. The main disadvantages of digital cardwall tools seem to be that they are often too complicated to use, the need to navigate to extra information not shown on the main board, no direct and concurrent interaction by all team members, and small displays, missing overview, missing instant availability.

4 aWall - Digital Agile Collaboration Wall

Based on our study we developed *aWall* to support Agile teams (co-located or distributed) more effectively than existing physical and digital tools. aWall is designed for various Agile team meetings (e.g. daily stand up, sprint planning, and retrospectives) by providing information dashboards, maintaining user stories and tasks, enables customization of Agile processes, and integrates with issue tracking systems. aWall was developed by an interdisciplinary project team of

computer scientists and psychologists (from the School of Engineering, and the School of Applied Psychology). We now outline the design and user interface of aWall, followed by a user study to evaluate the prototype.

4.1 Design

Based on the requirements elicited during the interviews, we identified a number of design considerations.

Physical Size. A digital cardwall needs to satisfy not only the needs for interacting with the digital content, but also provide enough physical space to display information to effectively support team collaboration. Therefore, the size of a digital cardwall needs to be at least comparable to that of physical cardwalls. aWall consists of four 46 in. displays (2×2), for a wall size of 2.05 m width and 1.25 m height (see Fig. 1).

High Resolution. Each display in aWall is 3840×2160 pixels, for a total resolution of 15360×8640 pixels. The high resolution display wall provides enough real estate to display large amounts of information at once while still ensuring the readability of text elements, widgets, and views.

Multi-user and Multi-Touch. The display wall consists of a 12 point multi-touch infrared optical overlay (PQ Labs frame[3]) which is attached to the display wall. The multi-touch capabilities allows multiple users to work simultaneously with artifacts and provides an accurate and effective touch experience.

Integration with Issue Tracking Systems. aWall is designed to run on top of existing third party issue tracking systems such as JIRA. Therefore, infrastructure functionality can be reused and already defined Agile processes utilized.

Availability of Information. aWall can replace physical cardwalls and act as the team's external memory of the project. For that, aWall should be installed in a team's open office area, always being switched on, and have a permanent view of the task board.

Web Technologies. In order to have a ubiquitous and easily deployable design, aWall was developed as a web application based on HTML5 and JavaScript technology. For multi-touch support we used the interact.js framework[4].

4.2 User Interface

The aWall user interface contains a number of different views, widgets, and interaction techniques designed to support different types of Agile meetings.

Action and Information View. The results of the interviews showed that most interaction with the cardwall takes place during Agile meetings. Each meeting

[3] http://multitouch.com/.
[4] http://interactjs.io/.

has specific goals, operates on different data, and requires different supporting tools and information. To support these different types of information handling, we divide the display into an *action view* and an *information view*. Figure 3 shows the view for a daily standup meeting highlighting the separation into information view and action view. The action view is the main work area, which is dedicated to the core artifacts of a specific meeting. The main interactions during a meeting are performed by users on the action view. The information view provides supporting information and tools needed for the meeting. The information view represents the dynamic memory of the team and as any dynamic system they need to allow for change. For example, the information view for the daily standup meeting contains additional information, like a timer widget showing the meeting moderator and a countdown, a team widget showing the team members, a definition-of-done widget, an impediment list widget, and a burn-down chart for an iteration. When necessary, new widgets can be added and removed from the information view.

Dedicated Views. aWall provides dedicated views that are tailored to the specific needs of Agile meetings. For the sprint planning meeting shown in Fig. 4, the action view is divided into three columns. The left column shows the top priority user stories of the product backlog. The centre column shows the so far selected user stories for the next iteration. The right column shows a detailed view of the currently selected user story. This column can be used by the product owner to discuss and clarify open issues during the meeting with the development team. Relevant documents can be easily attached and opened in the application. Figure 5 shows the retrospective meeting view after team members have sent their iteration feedback where the notes have been ordered on the right side. Users can navigate between the different meeting views by means of a navigation bar displayed at the bottom of the view.

Information Widgets. The information view consists of a set of widgets (e.g. team widget, timer widget, fun widget, avatar widget – see Figs. 3, 4, 5) and can be independently configured for each Agile meeting. Each widget is designed to support distinct aspects of the collaborative Agile process. The team widget shows the team members and can be used to assign people to tasks during a daily standup meeting. The timer widget supports time boxing during the meeting and furthermore, allows to choose a meeting moderator. The moderators' names are stored in the application and future moderators can be suggested based on previous selections. The fun widget allows users to post personal or fun images to the information view to help bring emotion to the cardwall and foster team thinking. The avatar widget can be used to drag avatars to any position on the wall or attach it to tasks or user stories. Both the fun and avatar widgets are designed to help with the interpersonal process in Agile teams (emotion management, team spirit). All widgets can be detached from the information view and moved around the cardwall to facilitate user interaction.

Availability of Information. Any information needed for a meeting is visible and accessible; either on the action view or on the information view. If the team

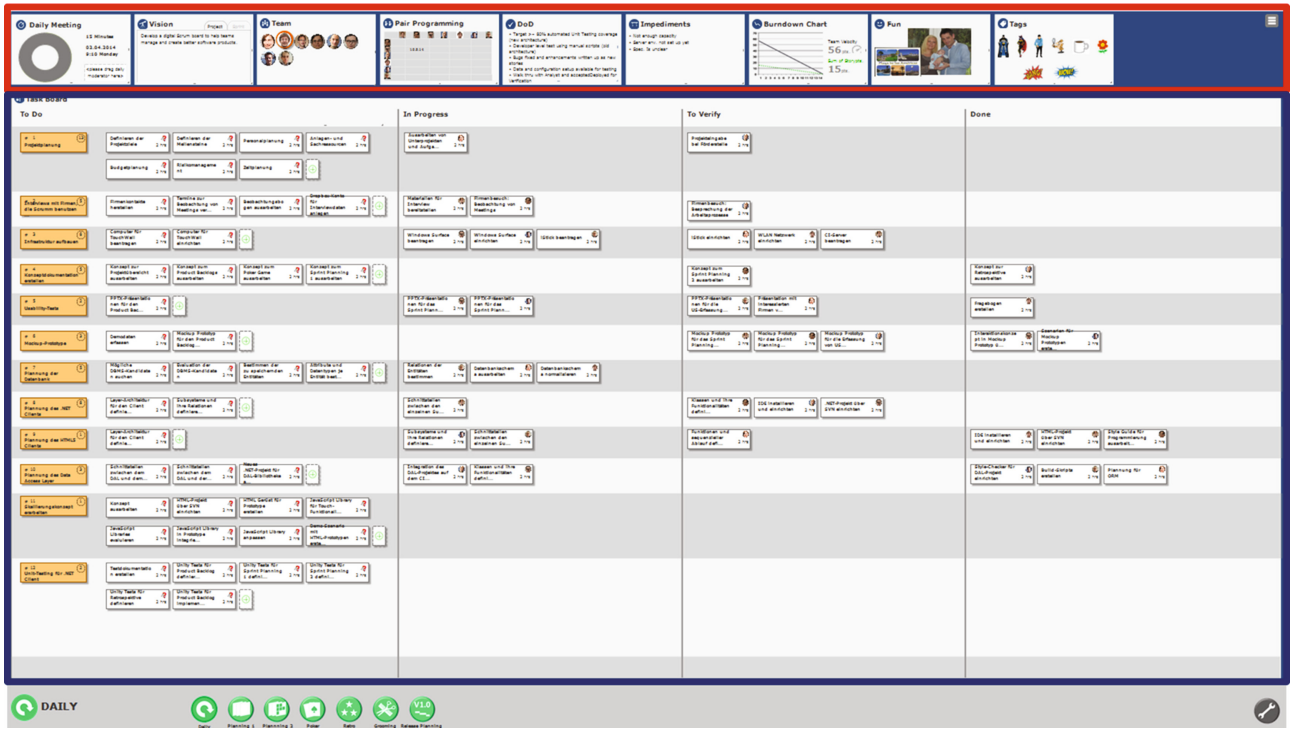

Fig. 3. Daily Standup with the following views: Information View (top section with red border) and Action View (middle section with blue border). (Color figure online)

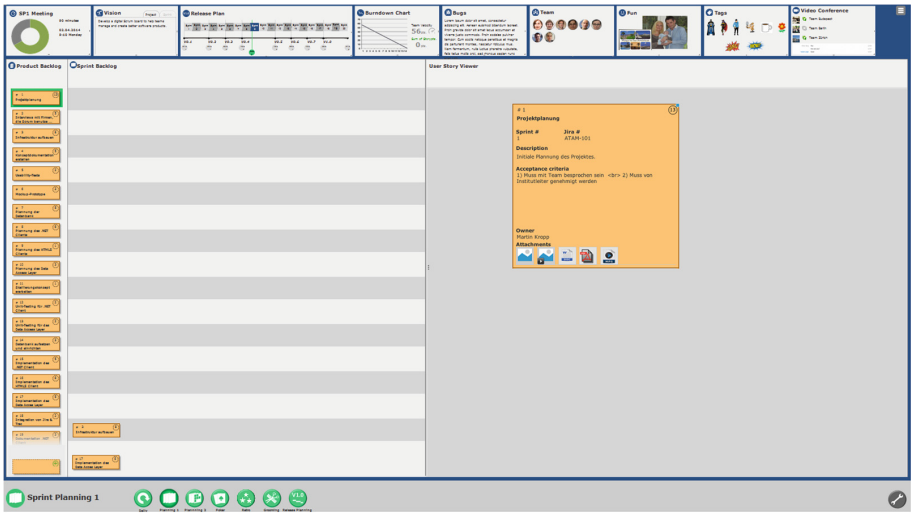

Fig. 4. Sprint planning meeting with a user story detail view.

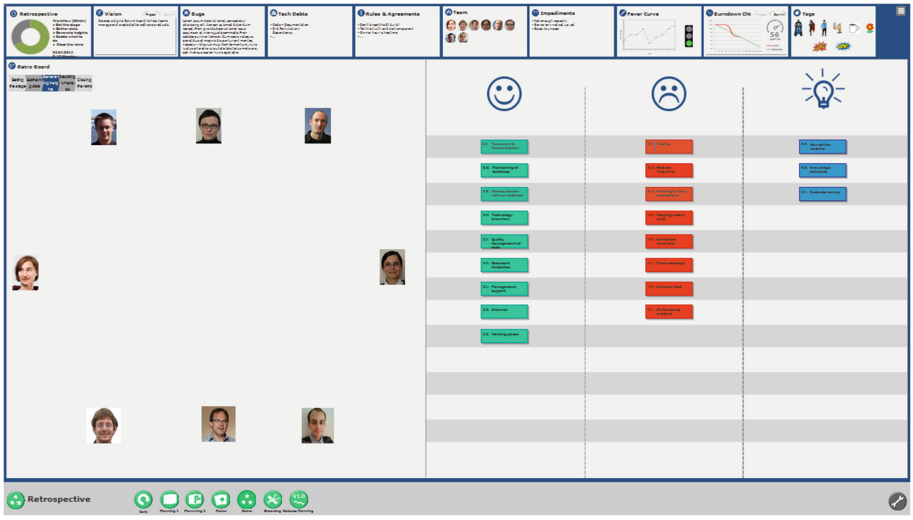

Fig. 5. Retrospective meeting view.

needs different supporting information, additional widgets can be switched on or off in the configuration button on the right side of the information view.

Interaction. aWall supports multi-touch and multi-user interaction. Fluid interaction with widgets and cards is enabled by gestures like tap, double tap, drag-and-drop, and pinch-to-zoom supporting changing task and user story cards position, moving widgets around the cardwall, and changing the size of a widget.

Data can be either entered on the cardwall with a virtual or physical keyboard or via the underlying issue tracker system and mobile devices such as tablets.

Scalability of Information. By default, user story cards and task cards show only a few details (e.g. title). By increasing the card size with a pinch-to-zoom gesture more information is displayed. The text size increases concomitantly with the widening of the cards so that information can be more easily read depending on the distance from the cardwall. When all information is shown the widget automatically switches into edit mode, so that data can be added or modified.

5 User Study

To evaluate the usability of the aWall prototype we conducted a qualitative user study with professional Agile practitioners. The goals were:

1. evaluate the availability of context specific information
2. evaluate reachability and discoverability of functionality and information
3. evaluate the support of Agile workstyle and Agile culture,
4. evaluate the applicability to real life situations in Agile teams.

The user study was conducted with an early prototype of aWall. The participants worked in teams and had to complete various tasks.

5.1 Participants

We recruited 11 employees (nine men and two women – see Table 2) from the same companies that participated in our interview study [8]. Most participants had many years of experience in IT, and several of them in Agile development. They came from different fields and covered a wide spectrum of Agile team roles. Among the participants were four Scrum Masters, two Agile coaches, two senior developers, one Agile grandmaster, one UX consultant and one head of a software development department. Two of the companies were from the assurance domain, one manufacturing, two service providers, one engineering, and one enterprise software development company. Four companies sent two employees, and three companies sent one employee each. All companies had been applying Agile processes for at least one year, and all employees to executed the given tasks in their companies before.

5.2 Procedure

We divided the eleven participants randomly into two groups by five and six people. Both groups completed the same tasks with aWall in two separate workshop sessions. Each workshop session lasted one hour.

Upon signing an informed consent statement, the participants were asked to act as a team during the workshop. Prior to the user study, the participants received a presentation on the interview study results, but did not receive any

Table 2. Demographics of workshop participants: gender, IT experience, Agile experience, job title, company (anonymized), and workshop group.

Gender	IT Exp.	Agile Exp.	Job title	Company	Group
Male	23	3	Head SW dev	D	1
Male	5	1.5	Senior dev	E	1
Male	13	2	Grandmaster	C	1
Male	10	3	Agile coach	F	1
Male	19	4	Senior dev	G	1
Male	10	3	UX consultant	B	1
Female	8	3	Agile coach	C	2
Female	15	5	Scrum master	A	2
Male	15	3	Scrum master	A	2
Male	1	1	Scrum master	E	2
Male	6	2	Scrum master	F	2

information about the aWall application. Each member of the team received three tasks to be solved together in groups using aWall. The tasks involved a daily standup meeting and a sprint planning meeting. After receiving the task, each participant read the task out aloud to the other participants and completed it with their help.

The daily standup tasks included to start the daily standup meeting (task 1), choose a moderator for the meeting (task 2), and update the task board during the meeting (task3), assign team members to a task card (task 4). For example: "*In this team you play the role of team member M. Please find a way to carry out a daily standup. The application suggests a moderator. Please ask the team member suggested by the application to play the moderator. Please act as a team accordingly to the received instructions.*". The sprint planning tasks included to show and discuss a user story during the meeting (task 1) and move the story to the sprint backlog (task 2). Other tasks were to switch on and off different widgets in the information view.

After completing the tasks for each type of meeting we asked the participants about the benefits and difficulties of aWall in an open discussion. The discussions were recorded and the results written down. Both team workshops were conducted by two moderators.

5.3 Findings

The overall feedback for the prototype was very positive, with the participants considering aWall to be usable, capable to support Agile processes in general and especially the collaborative working style in teams.

Size Aspects. The participants especially valued the large size and high resolution of aWall. The large size supports real team collaboration capabilities, similar to

physical cardwalls. Displaying a large amount of information at once was deemed positive. As one participant stated[5]:

> "With the large size you can display many user stories and tasks."

Readability of Information. Most participants considered the displayed information to be legible, especially since the card titles are relatively large. Some participants considered the actual cards to be too small. Therefore, it is very important to be able to display the whole content of a card and enlarge the font size so that the whole team can read it from a distance. One participant stated:

> "That's really a nice feature, that cards can be enlarged and font size increases to improve readability."

Availability of Information. The participants especially valued the availability of additional information and functionality for the different meetings. The separation of the display into action view and information view was easily understood. Some participants mentioned that elements placed on the upper side of the display wall might be out of reach for smaller people. Another participant liked the extra features:

> "I like the extra features around the main view and the additional information."

Discoverability of Functionality. The participants discovered most functionality of aWall by themselves and could easily interact with the display wall. There were some issues with discoverability of those functions that were not a straight-forward transfer of the pin-boards into the digital world. For example, the timer widget has no corresponding artifact in the practice of Agile teams. Whereas, direct implementations of the pin-boards functionality (e.g. the task-board shown in the daily standup meeting) were instantly understood and deemed as valuable by the participants. That was also the case for the widgets inspired from Agile practices such as the team widget which is based on the observation that Agile teams sometimes write the team members' names on the cards or even hang their pictures on the pin-boards.

Third-Party System Integration. The integration with third-party tools was positively rated. Tasks modified during the daily standup meeting, are immediately synchronized in the Agile project management tool (JIRA). There is no extra effort to update the tasks manually from the physical cardwall after the meeting. One participant stated:

> "The link to JIRA with automatic update of data is important."

Flexibility and Customization. Increased flexibility with respect to both the manner of conducting the meetings and displaying information was considered important by the participants. For example, the timer widget solicited choosing a moderator at the beginning of a meeting. The flexibility provided by aWall was also

[5] All quotes have been translated from German.

positively rated, especially with respect to conducting retrospective meetings that sometimes might prove strenuous. The participants considered that it is important to create a proper environment especially for this type of meeting as sometimes they tend to transmute into a drill. Most participants were in favour of a greater flexibility of the time boxing, with only optionally choosing a moderator and not showing the elapsed time, but the time of day during the daily meeting. The participants valued the team widget, but requested to have more information being displayed (e.g. absences, vacation days) and allow for more customization. Furthermore, the participants remarked that they should be able to add functionality to aWall on their own and not be dependent on standard functionality as often is the case with other Agile tools.

Agile Collaborative Workspace. Offering tags and avatars as well as the fun view was positively seen as bringing emotions onto the board. One participant mentioned the positive effect of avoiding media disruption, by being able to do all interaction with only one medium:

> "With such a board we could probably avoid media discontinuity."

Filtering and Representation of Information. The participants requested especially to have filter functions, to highlight and show the desired information. As an example, participants requested to highlight all tasks of a team member, when touching that person in the team view. The usage of colors for different types of user stories was suggested to increase readability (e.g. to distinguish between technical tasks, bug reports, or user requirements).

Task Time Recording. Some participants suggested automatically capturing the time spent on a task combined with computing of the work hours on the task would help provide further metric details of performance.

Provenance of Information. Some participants suggested to have automatic recordings of meetings with voice recognition and transcriptions of the discussions form the interactions in front of the display wall for later recollection and analysis of the meetings.

6 Conclusions

Current Agile cardwalls don't fulfil today's requirements for effective software development. We aim to bridge that gap with aWall, a digital cardwall tool to support co-located and distributed Agile teams. aWall provides a collaborative workspace using large multi-touch displays, information transparency, direct information interaction without the need for navigation, support for the whole Agile process, and dedicated views for different types meetings. We conducted a user study with 11 Agile practitioners and found that they especially valued the large-size of the wall due to the physical space affordances, the dedicated views with context specific information, and the always visible and direct information access. Our future work involves deploying aWall within companies.

Acknowledgments. Many thanks goes to the University of Applied Sciences Northwestern Switzerland for funding this project as part of their strategic initiative to fostering interdisciplinary work, and to the companies and people for participating in this project. Thanks to Robert Biddle for feedback on early drafts of this paper.

References

1. Anslow, C., Marshall, S., Noble, J., Biddle, R.: Sourcevis: collaborative software visualization for co-located environments. In VISSOFT, pp. 1–10. IEEE (2013)
2. Anslow, C., Marshall, S., Noble, J., Tempero, E., Biddle, R.: User evaluation of polymetric views using a large visualization wall. In: SoftVis, pp. 25–34. ACM (2010)
3. Atlassian: JIRA (2015). https://www.atlassian.com/software/jira
4. Azizyan, G., Magarian, M.K., Kajko-Matsson, M.: Survey of agile tool usage and needs. In AGILE, pp. 29–30. IEEE (2011)
5. Bragdon, A., DeLine, R., Hinckley, K., Morris, M.: Code space: touch + air gesture hybrid interactions for supporting developer meetings. In: ITS, pp. 212–221. ACM (2011)
6. Esbensen, M., Tell, P., Cholewa, J.B., Pedersen, M.K., Bardram, J.: The dBoard: a digital scrum board for distributed software development. In ITS, pages 161–170. ACM, 2015
7. Gossage, S., Brown, J., Biddle, R.: Understanding digital cardwall usage. In AGILE, pp. 21–30. IEEE (2015)
8. Mateescu, M., Kropp, M., Greiwe, S., Burkhard, R., Vischi, D., Zahn, C.: Erfolgreiche zusammenarbeit in agilen teams: Eine schweizer interview-studie ber kommunikation, 22 December 2015. http://www.swissagilestudy.ch/studies
9. Mayring, P.: Einfhrung in die qualitative Sozialforschung. Beltz-UTB, Weinheim (2003)
10. Morgan, R., Walny, J., Kolenda, H., Ginez, E., Maurer, F.: Using horizontal displays for distributed and collocated agile planning. In: Concas, G., Damiani, E., Scotto, M., Succi, G. (eds.) XP 2007. LNCS, vol. 4536, pp. 38–45. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73101-6_6
11. Muller, M., Wursch, M., Fritz, T., Gall, H.: An approach for collaborative code reviews using multi-touch technology. In: CHASE Workshop. ACM (2012)
12. Version One. Enterprise agile platform. (2015). http://www.versionone.com. Accessed 6 Jan 2017
13. Paredes, J., Anslow, C., Maurer, F.: Information visualization for agile software development teams. In: VISSOFT, pp. 157–166. IEEE (2014)
14. Rubart, J.: A cooperative multitouch scrum task board for synchronous face-to-face collaboration. In: ITS, pp. 387–392. ACM (2014)
15. Sharp, H., Robinson, H., Petre, M.: The role of physical artefacts in agile software development: two complementary perspectives. Interact. Comput. **21**(1–2), 108–116 (2009)
16. CA Technologies. CA agile central (2016). https://www.ca.com/. Accessed 6 Jan 2017
17. Research GmbH VERBI Software, Consult. Maxqda data analysis software (2015). http://www.maxqda.com. Accessed 6 Jan 2017
18. Wang, X., Maurer, F.: Tabletop agileplanner: a tabletop-based project planning tool for agile software development teams. In: TABLETOP, pp. 121–128. IEEE (2008)

Knowledge Sharing in a Large Agile Organisation: A Survey Study

Kati Kuusinen[1(✉)], Peggy Gregory[1], Helen Sharp[2], Leonor Barroca[2], Katie Taylor[1], and Laurence Wood[3]

[1] University of Central Lancashire, Preston, UK
{KKuusinen,AJGregory,KJTaylor}@uclan.ac.uk
[2] The Open University, Walton Hall, Milton Keynes, UK
{Helen.Sharp,Leonor.Barroca}@open.ac.uk
[3] IndigoBlue, London, UK
laurence.wood@indigoblue.co.uk

Abstract. Knowledge is a core resource for agile organisations that is transformed into products and services during the development process. Sharing of knowledge is essential across any organisation, and it has been claimed that the software industry requires more knowledge management than any other sector. Agile methodologies concentrate on team level collaboration, and some techniques for inter-team knowledge sharing have also proved to be successful. But these techniques focus on within-team and between-team knowledge sharing rather than knowledge sharing across the organisation. This paper presents the results of a survey with 81 responses on organisational knowledge sharing in a multinational agile company. The survey focuses on three aspects of knowledge sharing: within agile teams, beyond the team with company colleagues, and with customers. It concentrates on knowledge sharing practices, ease of knowledge sharing and motivation for knowledge sharing. Summary statistics, regression, and test of equity are used as analysis techniques. Results show that knowledge sharing with team members is significantly easier than with customers or company colleagues beyond their team. In addition, using agile practices improves ease of knowledge sharing within teams but not with customers or colleagues. Extrinsic motivators need to be in place to encourage knowledge sharing across the organisation, especially where such knowledge sharing is not an automatic consequence of completing the work.

Keywords: Knowledge sharing · Agile software development · Organisational knowledge sharing · Learning organisation

1 Introduction

Knowledge is awareness or understanding of something such as information or skills [4]. Knowledge creates most of the value in today's economy and the value of knowledge often increases when shared [23]. Organisational knowledge sharing

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 135–150, 2017.
DOI: 10.1007/978-3-319-57633-6_9

aims at transferring to the organisation the information, skills and experience a person or team has [10]. This is essential for sustaining the development of quality in software intensive companies [10]. For agile development companies, knowledge is the core resource that is transformed to products and services in the development process [2]. Moreover, Biao-wen [2] claims that the software industry requires more knowledge management than any other sector.

Agile methods focus heavily on the delivery of product and customer value. Moreover, an agile team focuses on applying knowledge instead of sharing it [10]. Agile methods facilitate knowledge sharing in the team but offer limited support for knowledge sharing outside the team [6,17,18]. Agile methods favour tacit knowledge shared informally using face-to-face communication (*personalisation strategy*) in contrast to traditional knowledge management practices [9]. Although attention has been paid to inter-team knowledge sharing [27], and techniques for distributed agile teams have proved to be successful, the focus here is on knowledge sharing across the organisation and not just between teams. The lack of knowledge sharing practices beyond the team can hinder sharing and sustaining knowledge in agile organisations [17].

This paper presents results of a baseline survey organised in a multinational agile software intensive company as part of their effort to improve organisational knowledge sharing. The results show that knowledge sharing with team members is significantly easier than with company colleagues or with customers. In addition, using more agile techniques is associated with increased ease of knowledge sharing with team members but not with colleagues outside the team and not with customers.

The rest of the paper is structured as follows: Sect. 2 introduces related research, Sect. 3 describes the research method, Sect. 4 presents the results, Sect. 5 considers limitations, Sect. 6 discusses the findings and Sect. 7 presents some conclusions and future work.

2 Related Work

Software engineering is a knowledge-intensive activity [25]. Software development teams are made up of knowledgeable individuals who need to be able to use, share, and communicate their knowledge in ways that foster problem solving and creativity. Whereas traditional software project approaches rely heavily on documentation and role-based working as ways of capturing and managing knowledge, agile approaches focus more on informal communication mechanisms within cross-functional teams [6,10].

Agile approaches employ intensive team work, face-to-face knowledge sharing, and trust as vital elements of working practice [1]. Research evidence shows that good team work is crucial for project success, with important facets including communication, coordination, balance of member contributions, mutual support, effort and cohesion [15]. Studies of agile teams have found that agile practices improve both informal and formal communication, and facilitate team and organisational communication [22]. Information visibility and sharing

are characteristics of agile approaches, especially when documentation is used. Sharp and Robinson [29] discuss how story cards and the Wall play an important part in the collaboration, co-ordination and communication processes of agile teams. Collaborative online tools are used to keep track of decisions and facilitate communication within collocated and distributed teams [8].

Knowledge management and learning theories have been used to explain the distinctiveness of the agile approach. Nonaka and Takeuchi's [21] distinction between explicit and tacit knowledge has been used to characterise the difference between traditional and agile approaches [6]. Explicit knowledge is objective, rational, and is easier to externalise in documents. In contrast, tacit knowledge is subjective, experience-based, and more likely to be context-specific and therefore easier to discuss than to document. Similarly, Hanssen et al. [14] identify two strategies for knowledge management: codification and personalisation. The codification strategy systematises and stores organisational knowledge, whereas the personalisation strategy supports the flow of information through the organisation through fostering connections between people and supporting a culture of communication. Traditional approaches tend towards codification whereas agile approaches tend towards personalisation.

Agile knowledge sharing practices can be roughly divided into practices among peers (e.g. communities of practice, pairing, coding dojos), among different specialists (shared specialists, interdisciplinary pairing, marathons), and among stakeholders and managers (scrum of scrums, review meetings). As agile becomes more widely adopted within companies and across industry, approaches for facilitating inter-team knowledge sharing and cross-organisational knowledge sharing need to be considered [3]. Inter-team personalisation strategies include Scrum of Scrums, project member rotation, communities of practice and open fishbowl sessions [27]. When viewed at an organisational level, knowledge is a significant competitive asset for a company. However, it is also challenging because of the scale and complexity of organisational environments and because the inter-team strategies do not address the needs of knowledge sharing across an organisation beyond teams collaborating in the same project.

Several authors identify that agile methods supply less advice for how to do this [6,17]. Santos et al. [27] propose a model showing how knowledge sharing between agile teams requires three elements: the adoption of practices, organisational support and appropriate stimuli. Recommended practices include face-to-face conversations, an informative workspace, rotation among teams and projects, collective meetings, pair programming between teams and projects, technical presentations, marathons, and coding dojos. Organisational support includes strategy, structure, culture, environment, top management and leadership support, communication flow and channels, integration among teams and projects, and deeper agile adoption. Appropriate stimuli include problems, common goals, incentives and sustainable pace.

3 Method

The research goal for the study was to identify areas that require improvement in organisational knowledge sharing in an agile company and to provide a baseline for assessing the progress and effectiveness of future actions. The study was initiated by the company who approached the authors[1] with a request to investigate their challenge. A survey[2] was used to reach a wide audience, it was sent to company employees (not customers), and concentrated on knowledge sharing between three groups: team members, company colleagues, and customers. The research questions are as follows.

RQ 1 How is knowledge shared in the organisation?
RQ 2 What motivates knowledge sharing in the organisation?
RQ 3 Is there a relation between agility and ease of knowledge sharing?
RQ 4 Is there a relation between frequency of knowledge sharing activities and ease of knowledge sharing?

3.1 Collaborator Company

The company in which the survey was conducted is a large IT service provider that primarily develops software for UK customers but has staff distributed over three continents. The majority of their workforce is based in India, and are sent to work in development teams at customer sites on a temporary basis in several countries worldwide. Development teams are assigned to a specific customer account and thus have a strong customer focus in their job and day-to-day responsibilities; many teams are embedded in the customer organisation and hence distant from each other. While some cross-organisational knowledge sharing tools and practices have been put in place such as wikis, Yammer, and profession-specific groups for training, these are limited.

3.2 Procedure

The survey was developed iteratively in collaboration with our company contacts and piloted first with students and then with a few company representatives. A link to the online survey was then distributed via a contact person in the company and it was advertised on the company intranet. The survey was open from May to July 2016 and there were altogether 113 responses from company employees of which 81 were completed. Of the 81 complete responses, 36 responded to the open-ended question on how to improve knowledge sharing in the company. No incentives were offered and two reminders were sent. The survey was anonymous. Mean completion time was 11 min (SD 19 min).

[1] The authors are members of the Agile Research Network (agileresearchnetwork.org) which is funded by the Agile Business Consortium Ltd. (ABC) Board, The Open University and University of Central Lancashire. Our research approach is explained here: Barroca, L., Sharp, H., Salah, D., Taylor, K., & Gregory, P. (2015). Bridging the gap between research and agile practice: an evolutionary model. IJSA, 1–12.

[2] The survey can be found from here: http://agileresearchnetwork.org/kss.

3.3 Survey

The survey addressed practices, motivators and ease of knowledge sharing with team members, company colleagues and with the customer. The survey had three sections, on (1) agile methods and agile techniques employed, (2) knowledge sharing and (3) background information. Questions on knowledge sharing were related to frequency of use of knowledge sharing practices, motivation towards sharing and experienced ease of sharing. Survey themes were as follows

1. Agile methods employed (question 1, multiple choice)
2. Agile techniques employed (question 2, multiple choice)
3. Frequency of use of knowledge sharing practices with team members (question 3, pre-defined list of practices assessed on four-point frequency scale)
4. Frequency of use of knowledge sharing practices with company colleagues outside the team (question 4, pre-defined list of practices assessed on four-point frequency scale)
5. Frequency of use of knowledge sharing practices with customer (question 5, pre-defined list of practices assessed on four-point frequency scale)
6. Motivation for knowledge sharing with team members, company colleagues and customer (question 6, multiple choice)
7. Ease of knowledge sharing with team members, with company colleagues outside the team and with customer (question 7, five-point Likert scale)
8. Suggestions for how to improve knowledge sharing in the company (question 8, open-ended)

In addition we asked for background information including job role, years of experience in the company, the number of customer accounts and the number of people led if any.

The survey was designed to address the needs of the collaborator company and drew on existing literature. The first two questions on agile methods and techniques were adopted from the annual state of agile survey by Version One [31]. Question six on motivation was adapted from [19] and consisted of six statements measuring intrinsic and extrinsic motivation.

3.4 Analysis

We used basic descriptive statistics such as means to summarise responses on the structured questions. Since the data complied with the assumptions [5] of linear regression (F), a commonly used predictive analysis, we used it to study the relation between experienced ease of knowledge sharing and agility or frequency of knowledge sharing activities. We assumed that agility increases with the number of agile techniques employed. Gandomani et al. [11] propose a model and formula for calculating agility based on practices used. They use a list of 44 practices, of which ours is a sub-set. Thus, we use linear regression analysis to test whether experienced ease of knowledge sharing can be predicted from

1. number of specific agile techniques employed (RQ 3)
2. reported frequency of use of knowledge sharing practices (RQ 4)

Based on Shapiro-Wilk test, the data was non-normal and thus we used a non-parametric hypothesis test. The selected Wilcoxon's signed rank test (Z) is a non-parametric statistical hypothesis test for comparing two related samples, e.g. two responses given by one single individual in a survey. We used the Wilcoxon test for equity to measure if there is a statistically significant difference between the experienced ease of knowledge sharing with team members, company colleagues and customers (RQ3, RQ4) and if there is a difference in the frequency of reporting motivation sources for sharing between those three groups (RQ2).

When sharing with either element of each of the partner pairs (team members and company colleagues, team members and customer, company colleagues and customer) the hypotheses are as follows:

1. *there is no difference between the ease of knowledge sharing;*
2. *there is no difference between the frequency of intrinsic motivation sources;*
3. *there is no difference between the frequency of extrinsic motivation sources.*

The hypotheses are a combination of the interests of the studied company and literature. For the open-ended question the data was collated and thematically analysed using an inductive, qualitative, data-driven content analysis with the aim of generating thematic groupings from the data [26], with no preconceived ideas about what would emerge.

3.5 Respondents

The response rate was 9%. The main job responsibility of the 81 respondents was as follows: software development 42%, architecture 16%, project management 15%, software testing or quality 7%, business or system analyst 6%, design or UX design 4%, configuration/support 1% and other roles 9% (coaching or training or a mixture of development and design roles). Of the 81 respondents, 43% did not lead a team or function, 35% led 1 to 9 persons, 14% led 10 to 19 persons and 9% led over 19 persons. Almost all the respondents worked for customer accounts: 4% had not worked for a customer account, 30% had worked for one customer account, 40% for 2 to 4 customer accounts and 27% had worked for five or more customer accounts. On average, respondents had worked for the company for 7 years, standard deviation 6 years.

4 Results

For answers about agile methods and techniques multiple responses were possible. Scrum was the most used agile method reported by 83% of respondents. Kanban and Scrumban were also often used, reported respectively by 32% and 22% of the respondents. The most often employed agile techniques were daily standups, prioritised backlogs, iteration or sprint planning, retrospectives and short iterations or sprints (Fig. 1).

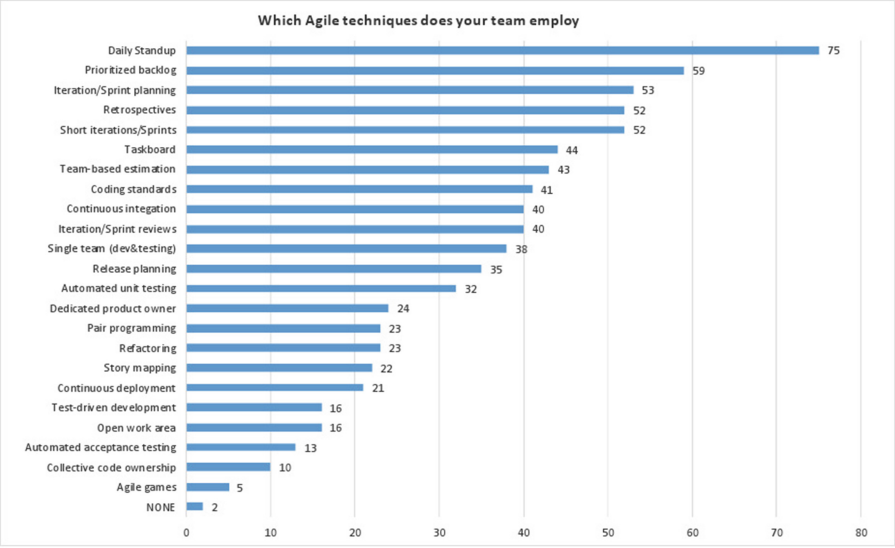

Fig. 1. Employed Agile techniques [31]

4.1 Knowledge Sharing Practices

The most common techniques for knowledge sharing in general were informally, in meetings, and by email (Fig. 2). In general, knowledge sharing was more frequent within teams than with customers or company colleagues outside the team. This is an expected result as teams are often the fundamental social units of an organisation's knowledge creation [16] and Scrum - the most widely used agile method in the company - emphasises the role of collaborative teams. Sharing knowledge with colleagues was most often done informally, whereas when sharing knowledge with customers, meetings were the most frequent technique. Both represent a personalisation knowledge sharing strategy (person-to-person) which is the favoured strategy in agile. The next most commonly used knowledge sharing techniques with customers were email and through the team lead or a senior member of the team.

4.2 Motivation for Knowledge Sharing

The mean number of reported motivation sources per respondent was higher for sharing knowledge with team members than with either company colleagues or customers (Fig. 3). There was a difference between the frequency of intrinsic and extrinsic motivators when sharing with customers compared to when sharing with either team members or company colleagues. When sharing knowledge with team members or company colleagues, a greater number of respondents reported intrinsic sources of motivation than extrinsic sources whereas when sharing with

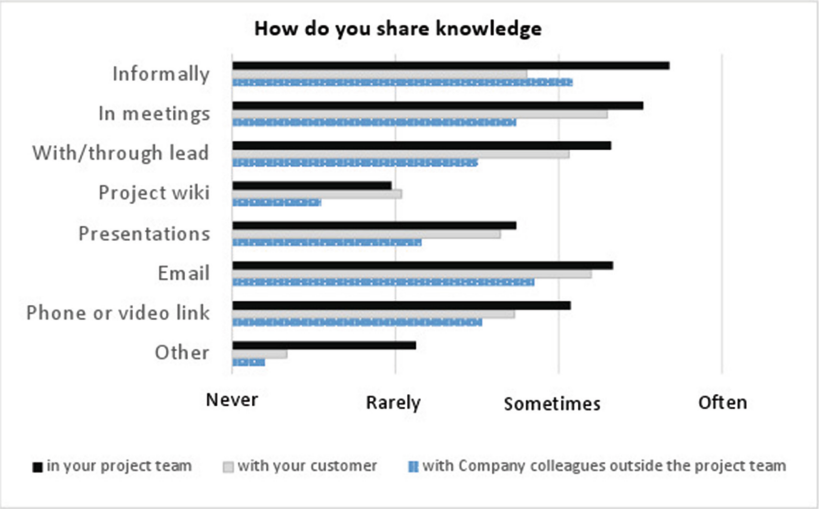

Fig. 2. Mean frequency of use of knowledge sharing practices in team, in company and with customer. N = 81

Table 1. Percentage of respondents reporting motivation source types per sharing partner. N = 81.

Motivation source	Team	Colleague	Customer
Both extrinsic and intrinsic	85%	63%	59%
Intrinsic only	14%	16%	10%
Extrinsic only	1%	10%	21%
None	0%	11%	10%

customers a greater number of respondents reported extrinsic sources of motivation than intrinsic sources (Table 1).

Enjoyment was the most common motivator for knowledge sharing with team members (90% of respondents mentioned it) and with company colleagues (67%) whereas with customer it was strengthening ties (64%) (Fig. 3). Enjoyment is an intrinsic motivator whereas strengthening ties is an extrinsic motivator [16,19].

The Wilcoxon signed rank test was applied to the data. Based on the results, all hypotheses considering motivation sources were rejected apart from the following: *there is no difference between the frequency of intrinsic motivation sources* (1) *when sharing with company colleagues* and (2) *when sharing with customers* (Table 2). However there is a significant difference in the frequency of reporting *extrinsic* motivation sources between sharing knowledge with company colleagues and customers. The most obvious difference is that strengthening ties was an especially frequent source of motivation for sharing knowledge with customers, which is important for maintaining the relationship with the customer.

Table 2. Wilcoxon signed rank test on the frequency of motivation sources for sharing knowledge with team members, company colleagues and customer. N = 81.

Compared sharing partner pair	Test outcome (Z)	Level of significance (p)
Intrinsic: Team members - Colleagues	Z = −3.98	p <.001
Intrinsic: Team members - Customer	Z = −4.94	p <.001
Intrinsic: Colleagues - Customer	Z = −1.53	n.s.
Extrinsic: Team members - Colleagues	Z = −4.12	p <.001
Extrinsic: Team members - Customer	Z = −2.33	p <.05
Extrinsic: Colleagues - Customer	Z = −2.00	p <.05

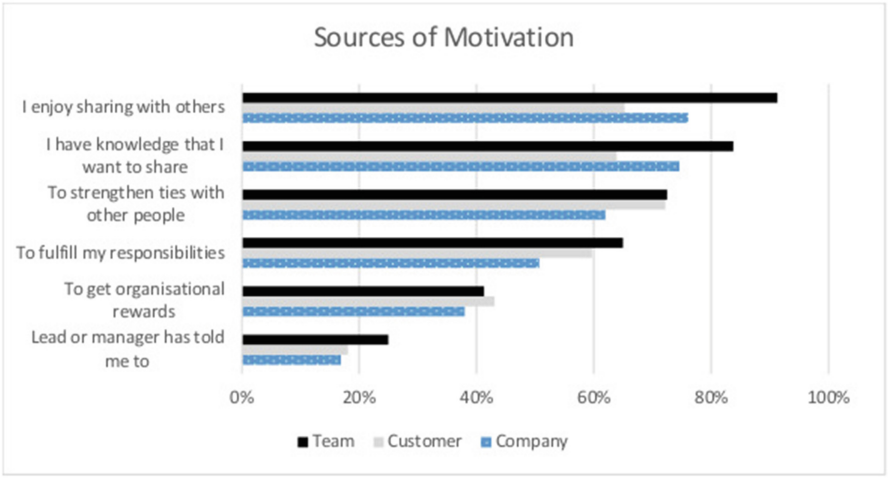

Fig. 3. Frequency of motivation sources for knowledge sharing with team members, customer and company colleagues outside the team. N = 81.

In summary, this test showed differences between the frequencies of *extrinsic* motivation sources for sharing with *all* the sharing partners and between the frequencies of *intrinsic* sources between all the sharing partners except *company colleagues* and *the customer*.

4.3 Ease of Knowledge Sharing

Knowledge sharing within teams was reported to be easy whereas knowledge sharing beyond the team with company colleagues and with customers was less easy (Fig. 4). A Wilcoxon signed rank test was applied to the findings. This revealed that knowledge sharing with team members was significantly easier than with customers ($Z = -4.51,\ p < .001$). It also revealed that knowledge sharing with team members was significantly easier than with company colleagues outside the team ($Z = -4.52,\ p < .001$). Based on the test, the hypotheses *there*

is no difference between the ease of knowledge sharing with team members and customers and *there is no difference between the ease of knowledge sharing with team members and company colleagues* were rejected while the hypothesis *there is no difference between the ease of knowledge sharing with company colleagues and customers* was accepted. Of the respondents, 62% strongly agreed that knowledge sharing with team members is easy whereas 28% and 27% strongly agreed that knowledge sharing with customers or with company colleagues, respectively, is easy. Knowledge sharing with customers was considered slightly easier than with company colleagues (Fig. 4). Only 9% did not agree that knowledge sharing is easy with team members whereas 30% did not agree that knowledge sharing is easy with customers and 33% did not agree that knowledge sharing is easy with company colleagues outside the team. Thirty-six employees suggested improvements for organisational knowledge sharing in an open-ended question. Almost all of the suggestions were about knowledge sharing in the company outside the team. Half of the respondents suggested having small informal sessions among interested individuals to share knowledge, for example, about architectural solutions or new technologies. Also, half of respondents suggested either creating new knowledge bases, or repositories, or using the current ones more efficiently. Other ideas included fostering the company culture to embrace knowledge sharing. Such a culture would build on trust and encourage people to share their knowledge instead of making them fear they are replaceable if they share.

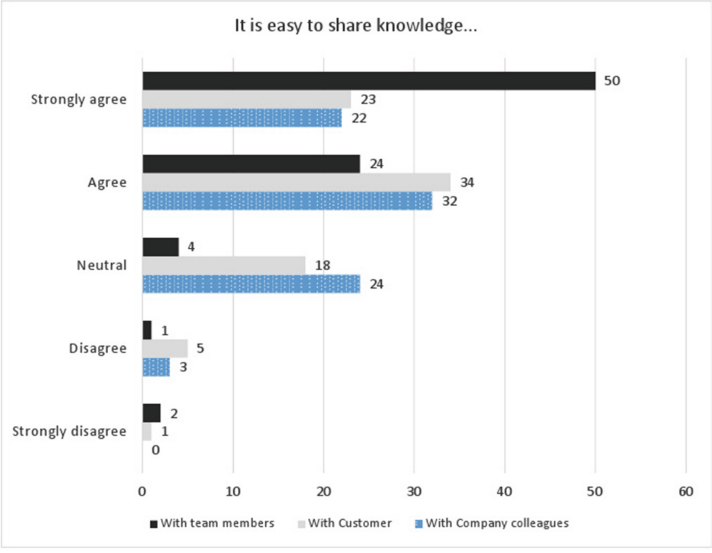

Fig. 4. Perceived ease of knowledge sharing with team members, customer and colleagues. N = 81.

4.4 Relation of Agility and Ease of Knowledge Sharing

Experienced ease of knowledge sharing with team members could be predicted from the number of agile techniques employed using linear regression $F(80,1) = 10.7$, $p < .01$. Thus, the greater the number of agile techniques employed, the easier knowledge sharing with team members was experienced.

There is no direct association between the number of agile techniques employed and experienced ease of knowledge sharing with company colleagues, $F(80,1) = 0.0$, *n.s*, nor between the number of agile techniques employed and experienced ease of knowledge sharing with customers, $F(80,1) = 2.7$, *n.s.*

4.5 Relation of Frequency and Ease of Knowledge Sharing

There is a direct association between the frequency of use of knowledge sharing practices and experienced ease of knowledge sharing with team members: the more frequently knowledge sharing practices are used, the easier knowledge sharing is, $F(78,12) = 3.6$, $p < .001$. When ease of knowledge sharing with team members was calculated from knowledge sharing practices, using the whiteboard ($t = 3.8$, $p < .001$) and doing it informally ($t = 2.8$, $p < .01$) are significant positive predictors whereas using Yammer ($t = -2.0$, $p < .05$) was a significant negative predictor. Thus, the more frequently whiteboards are used for knowledge sharing or the more frequently knowledge is shared informally with team members, the easier knowledge sharing with team members is experienced. On the contrary, the more often knowledge is shared via Yammer with team members, the less easy knowledge sharing with team members is experienced.

Using a whiteboard requires face-to-face contact whereas Yammer moves people away from physical presence, which may explain the negative association. These results indicate that knowledge sharing is easier where frequent, informal sharing takes place, including using whiteboards as a knowledge sharing tool.

Experienced ease of knowledge sharing with company colleagues can be predicted from the frequency of use of knowledge sharing practices using multiple linear regression, $F(80,11) = 1.9$, $p < .05$. When ease of knowledge sharing with colleagues was calculated from knowledge sharing practices, giving presentations ($t = -2.0$, $p < .05$) was a significant negative predictor.

In general, the more frequently knowledge sharing practices are used, the easier knowledge sharing appears to be. However, giving presentations is a negative predictor. A possible explanation for this negative association is that presentations are often one-directional: the presenter shares their information with the audience. Furthermore, the company also shares presentations via email. Using one-directional practices for knowledge sharing may decrease the experienced ease of knowledge sharing.

There is a direct association between the frequency of use of knowledge sharing practices and the experienced ease of knowledge sharing with customers. Experienced ease of knowledge sharing with customers can be predicted from the frequency of use of knowledge sharing practices using multiple linear regression, $F(80, 7) = 5.8$, $p < .001$. When ease of knowledge sharing with customers

was calculated from knowledge sharing practices, using a wiki *($t = 3.6$, $p < .01$)* and having meetings *($t = 2.6$, $p < .05$)* were significant predictors. Thus, knowledge sharing is easier when a collaborative exchange of information is frequently used and meetings with the customer are frequent.

5 Limitations

Construct validity relates to the appropriateness of the survey as a measure. Several techniques were used to mitigate this threat. Questions 1, 2 and 6 in the survey were based on questions found in existing literature, to ensure that terminology used was in common use. The survey was developed iteratively and piloted with practitioners. Some of the questions contained repetition asking respondents to consider knowledge sharing from three perspectives, with team members, company colleagues and customers. Factors such as boredom and practice could have impacted the results. Question randomisation or counterbalancing were not used because of limitations of the surveying tool. Multiple response was controlled by allowing only one response per device. **Internal validity** relates to causal conclusions drawn. We used the number of specific agile techniques employed as a measure of agility in the survey, an approach used by [11,24]. This is not sophisticated, however in the context of this survey it provides a useful indication of how agility varies within the company. The strength of motivators was not asked for and therefore it is unknown if some of them are more powerful than the others. The measures of agility and motivation were both used in the linear regression analysis. Moreover, statistical tests are always prone to incorrect rejection or retaining of the null hypothesis and multiple hypothesis testing increases the risk. We did not use adjustments for these error types since correction of one of the types increases the risk to the other type. **External validity** relates to generalizability of the findings. As only one company was surveyed, the results are specific to that company. Moreover, only a number of employees responded to the survey which makes it prone to non-response bias.

6 Discussion

The summary answers to our research questions are as follows:

RQ 1 *How is knowledge shared in the organisation?* The top three knowledge sharing practices are: sharing informally, in meetings, and through email. Sharing knowledge with colleagues is most often done informally whereas with customers the most common means is in meetings.

RQ 2 *What motivates knowledge sharing in the organisation?* Respondents cited more motivators for sharing with team members than with company colleagues or customers. Motivators for knowledge sharing with team members and with company colleagues are more frequently intrinsic than extrinsic; motivators for knowledge sharing with customers are more frequently extrinsic.

RQ 3 *Is there a relation between agility and ease of knowledge sharing?* Sharing knowledge within teams is statistically significantly easier than with customers or company colleagues. The regression analysis shows that using agile techniques improves ease of knowledge sharing within agile teams but not with company colleagues or with customer.

RQ 4 *Is there a relation between the frequency of knowledge sharing activities and ease of knowledge sharing?* In general the more frequently knowledge sharing practices are used, the easier knowledge sharing is. However, there are nuances in the data with some practices improving knowledge sharing and some hindering it. Our findings suggest that knowledge sharing is easier if face-to-face and informal contact is used, whereas one-way presentations decrease the perceived ease of knowledge sharing.

Our findings indicate that specific agile techniques improve ease of knowledge sharing within teams. This confirms findings from Pikkarainen et al. [22] who found that agile practices improved both informal and formal communication, and [20], who suggest that the "knowledge-as-relationship" focus of agile teams facilitates team knowledge sharing. It also confirms common-sense expectations that agility improves knowledge sharing and communication within the team.

Our findings also suggest that a high level of agility helps knowledge sharing to some extent with customers, but has little impact on knowledge sharing with company colleagues. This finding confirms the view that simply using agile techniques does not help much with inter-team knowledge sharing [6,17].

Software engineers are outcome-oriented and motivated by technically interesting content and the work itself [28]. One of the characteristics of agile working is that all of the team's effort is focused on producing code that provides business value, and that plays directly into this motivation profile. In this context, sharing experiences with company colleagues who are not directly involved in the same project or with the same customer, requires compelling extrinsic motivators. Yet motivators for knowledge sharing with company colleagues were intrinsic rather than extrinsic. Therefore, it seems clear that this organisation does not have sufficient extrinsic motivators in place to encourage knowledge sharing with company colleagues.

These results are influenced by the collaborator's specific cicumstances, and these require further investigation. For example they are mostly based in India and the Indian agile community faces a range of challenges [30], are embedded in customer sites around the world, and hence at a distance from each other.

7 Conclusions and Future Work

Our survey study contributes to an understanding of how knowledge is shared in agile organisations. We provide evidence to support claims that knowledge sharing is easier within agile teams. In this instance, we find that these benefits do not apply to knowledge sharing across the organisation. Extrinsic motivators need to be in place to encourage knowledge sharing across the organisation,

especially where such knowledge sharing is not an automatic consequence of completing the work.

Further research is required to investigate how knowledge sharing may be improved across this organisation, to compare their situation with other companies, and to understand better how the organisation's specific situation influences knowledge sharing behaviour. Suggestions from literature will be used to guide the next stage, for example ecosystems, communities of practice, shared specialists, coding marathons and project members' rotation [6,7,27,32]. Santos et al's [27] model of inter-team knowledge sharing suggests that three elements are important in inter-team knowledge sharing: the adoption of specific practices, organisational support and appropriate stimuli. Some of their suggestions for practices, such as job rotation, role pairing between projects and informal cross-organisational networks are not currently in place, but could be introduced. Han and Anantatmula's [13] model for knowledge sharing in large IT organisations identifies organisation, technology, learning and leadership as important components. Their suggestions for leadership highlight the importance of aspects such as a management help with knowledge sharing, verbal praise, encouragement, and career promotion. These observations could be characterised as cultural issues, such as those identified in [12].

References

1. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al.: The Agile Manifesto (2001)
2. Biao-wen, L.: The analysis of obstacles and solutions for software enterprises to implement knowledge management. In: 2010 The 2nd IEEE International Conference on Information Management and Engineering (ICIME), pp. 211–214. IEEE (2010)
3. Bjørnson, F.O., Dingsøyr, T.: Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used. Inf. Softw. Technol. **50**(11), 1055–1068 (2008)
4. Charband, Y., Navimipour, N.J.: Online knowledge sharing mechanisms: a systematic review of the state of the art literature and recommendations for future research. Inf. Syst. Front., pp. 1–21 (2016)
5. Chatterjee, S., Simonoff, J.S.: Handbook of Regression Analysis, vol. 5. Wiley, New York (2013)
6. Chau, T., Maurer, F., Melnik, G.: Knowledge sharing: Agile methods vs. Tayloristic methods. In: WETICE, vol. 3, pp. 302–307 (2003)
7. Cockburn, A., Highsmith, J.: Agile software development, the people factor. Computer **34**(11), 131–133 (2001)
8. Deshpande, A., Sharp, H., Barroca, L., Gregory, P.: Remote working and collaboration in agile teams. In: International Conference on Information Systems, ICIS 2016. AIS Electronical Library (2016)
9. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**(9), 833–859 (2008)
10. Ersoy, I.B., Mahdy, A.M.: Agile knowledge sharing. Int. J. Softw. Eng. (IJSE) **6**(1), 1–15 (2015)

11. Gandomani, T.J., Nafchi, M.Z.: An empirically-developed framework for agile transition and adoption: a grounded theory approach. J. Syst. Softw. **107**, 204–219 (2015)
12. Gregory, P., Barroca, L., Sharp, H., Deshpande, A., Taylor, K.: The challenges that challenge: engaging with agile practitioners concerns. Inf. Softw. Technol. **77**, 92–104 (2016)
13. Han, B.M., Anantatmula, V.S.: Knowledge sharing in large it organizations: a case study. Vine **37**(4), 421–439 (2007)
14. Hansen, M.T., Nohria, N., Tierney, T.: Whats your strategy for managing knowledge? In: The Knowledge Management Yearbook 2000–2001, pp. 55–69 (1999)
15. Hoegl, M., Gemuenden, H.G.: Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence. Organ. Sci. **12**(4), 435–449 (2001)
16. Hung, S.Y., Durcikova, A., Lai, H.M., Lin, W.M.: The influence of intrinsic and extrinsic motivation on individuals' knowledge sharing behavior. Int. J. Hum. Comput. Stud. **69**(6), 415–427 (2011)
17. Karlsen, T.J., Hagman, L., Pedersen, T.: Intra-project transfer of knowledge in information systems development firms. J. Syst. Inf. Technol. **13**(1), 66–80 (2011)
18. Kettunen, P., Laanti, M.: Combining agile software projects and large-scale organizational agility. Softw. Process Improv. Pract. **13**(2), 183–193 (2008)
19. Lin, H.F.: Effects of extrinsic and intrinsic motivation on employee knowledge sharing intentions. J. Inf. Sci. **33**(3), 340–359 (2007)
20. Melnik, G., Maurer, F.: Direct verbal communication as a catalyst of agile knowledge sharing. In: Agile Development Conference 2004, pp. 21–31. IEEE (2004)
21. Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, New York (1995)
22. Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J.: The impact of agile practices on communication in software development. Empirical Softw. Eng. **13**(3), 303–337 (2008)
23. Quinn, J.B., Anderson, P., Finkelstein, S.: Managing professional intellect: making the most of the best. In: Strategic Management of Intellectual Capital, pp. 87–100 (1998)
24. Qumer, A., Henderson-Sellers, B.: An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Inf. Softw. Technol. **50**(4), 280–295 (2008)
25. Qumer, A., Henderson-Sellers, B.: A framework to support the evaluation, adoption and improvement of agile methods in practice. J. Syst. Softw. **81**(11), 1899–1919 (2008)
26. Ritchie, J., Lewis, J., Nicholls, C.M., Ormston, R., et al.: Qualitative Research Practice: A Guide for Social Science Students and Researchers. Sage (2013)
27. Santos, V., Goldman, A., De Souza, C.R.: Fostering effective inter-team knowledge sharing in agile software development. Empirical Softw. Eng. **20**(4), 1006–1051 (2015)
28. Sharp, H., Baddoo, N., Beecham, S., Hall, T., Robinson, H.: Models of motivation in software engineering. IST **51**(1), 219–233 (2009)
29. Sharp, H., Robinson, H.: Three 'C's of Agile Practice: Collaboration, Co-ordination and Communication. In: Dingsøyr, T., Dybå, T., Moe, N.B. (eds.) Agile Software Development, pp. 61–85. Springer, Heidelberg (2010)

30. Srinivasan, J., Lundqvist, K.: Agile in India: challenges and lessons learned. In: Proceedings of ISEC 2010 the 3rd India Software Engineering Conference, pp. 125–130. ACM, New York (2010)
31. VersionOne: Annual State of Agile Survey (2016). http://stateofagile.versionone.com. Accessed 29 Nov 2016
32. Wenger, E., McDermott, R.A., Snyder, W.: Cultivating Communities of Practice: A Guide to Managing Knowledge. Harvard Business Press (2002)

Teaching Agile Methods to Software Engineering Professionals: 10 Years, 1000 Release Plans

Angela Martin[1], Craig Anslow[2(✉)], and David Johnson[3]

[1] Xero, Wellington, New Zealand
angela.m.martin@gmail.com
[2] School of Engineering and Computer Science,
Victoria University of Wellington, Wellington, New Zealand
craig@ecs.vuw.ac.nz
[3] Oxford E-Research Centre, University of Oxford, Oxford, UK
david.johnson@oerc.ox.ac.uk

Abstract. Agile methods are an essential resource for software engineers. The Agile movement evolved out of industry and is the common approach to software development today. Teaching Agile methods challenges students' working attitudes, where putting Agile into practice is not possible through simply applying methods prescriptively, but by having an Agile mindset. In this paper we present and discuss our experiences over the last decade of teaching a novel intensive Agile methods week long course as part of a professional Masters of Software Engineering degree programme at the University of Oxford. We describe the typical shape of the course, discuss how students experience Agile values and management practices to foster an Agile mindset, and provide student feedback indicating a consistently positive response to our approach at teaching Agile methods to software engineering professionals. Our reported experiences and material can help other educators who want to run similar courses and adapt where required.

Keywords: Agile software development · Experience report · Group work · Graduate programs · Software engineering professionals

1 Introduction

Since the introduction of the Agile Manifesto, Agile methods in software engineering have gained popularity year on year, and today Agile is not just commonplace, but often expected as a standard industry practice in software development teams. Agile methods were evolved by and are applied by industry [6]. This growth in applying Agile principles in the software industry went hand-in-hand with a growth in Agile training being offered to software engineering professionals in the work place, as well as more recently in undergraduate and some graduate computer science and software engineering degrees.

The University of Oxford Software Engineering Programme (SEP) was established in 1993 and exists to create strong connections between theory and practice in software engineering and to make the expertise of the university available

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 151–166, 2017.
DOI: 10.1007/978-3-319-57633-6_10

to those who wish to study part-time while continuing in full-time employment. Most students on SEP are practicing software engineering professionals who often already have university degrees or extensive industry experience. Week-long intensive courses in a variety of subjects are offered, with up to 16 students per class. Each student must take 10 courses in any order over a four year period and are used as credit towards a Masters' degree (MSc) in Software Engineering awarded by the University of Oxford. In 2007 the Agile Methods (AGM) course[1] was introduced in response to the growing needs for software engineering professionals to understand and introduce Agile in their work places.

Teaching Agile methodologies often focuses on learning a particular method [6], such as Scrum [14] or XP [2]. It was recognized that the intensive nature of the week long part-time courses at Oxford made it difficult for an in-depth dive into the variety of Agile methods and practices to fit into the short time span of an individual course. To this end, the Agile Methods course is devised to bring students into an Agile mindset – through a combination of (1) coupling lectures with simulated exercises of Agile management practices, (2) critical analysis and debate around case studies on Agile adoption, and (3) hands-on approach of Agile practices within the classroom.

In this paper, we present an approach to teaching Agile to software engineering professionals and discuss our experiences over the last 10 years of delivering the course. We give a course outline describing the pre-course assignment, case studies, lecture content, group exercises, post-course assignment, and finally discuss lessons learned from teaching the course over a long period of time. Other educators who wish to run similar courses can learn from our experiences and material reported in this paper and adapt where required.

2 Course Outline

The Agile Methods course aims to give an overview of Agile to software engineering professionals and help them understand and adopt an Agile mindset. The learning objectives of the course are as follows: (1) compare and contrast the different agile methods, (2) determine the suitability of agile methods for a particular project and organization, (3) evaluate how well a project is following agile principles, and assist the project to become more agile (where appropriate), (4) understand the relationship between the customer and the development team in agile projects and the responsibilities of both communities, and (5) how to foster organizational change to build better software.

The course is scheduled for a week and spans five consecutive days (Monday to Friday), where each day is timetabled from 0900 to 1730, except for Friday where the class concludes at lunch time (see Table 1 for an example Agile Methods course schedule). The week long class is split into discrete time boxes, with three sessions in the mornings, and three in the afternoons, concluding each day with a learning stand up. Each time box consists of a lecture, an exercise, or a case study discussion, with breaks in between each session.

[1] https://www.cs.ox.ac.uk/softeng/subjects/AGM.html.

Table 1. An example University of Oxford Agile Methods (AGM) course schedule. Encoding: Lectures – yellow, Group Exercises – blue, Case Study – green. Three sessions in the morning and three in the afternoon followed by a learning stand-up. Small coffee and tea breaks happen between each session.

Time	Monday	Tuesday	Wednesday	Thursday	Friday
0900-1000	Introduction	Case Study	Case Study	Case Study	Case Study
1015-1115	Agile Manifesto	XP	Empirical Research	Personas	Retrospectives
1130-1230	Communication	Release Planning & User Stories	Lean & Kanban	User Stories	Retrospective Q&A, Survey
1230-1330	*Lunch*	*Lunch*	*Lunch*	*Lunch*	*Lunch*
1330-1430	Case Study	Case Study	Case Study	Estimation	
1445-1545	Scrum	Iteration Planning & Estimation	Kanban Game	Release Planning	
1600-1700	Marshmallow Challenge	Coffee Machine Game	Kanban Game Cont'd	Iteration Planning	
1700-1730	Learning stand-up	Learning stand-up	Learning stand-up	Learning stand-up	

The only prerequisite is that a student must be already enrolled in the SEP. To cover enough Agile background material and different methods we use Agile Software Development Ecosystems [8] as the text book. A pre-study assignment is given to each student to help them prepare in advance of the teaching week and a post-course assignment as the student's assessment.

2.1 Pre-study Assignment: Case Study

It is important for students to begin their study about Agile in advance of the teaching week, and we cater for this by sending them an assignment four weeks in advance of the course. One of the main themes that the course explores is that of Agile adoption; and not just the idealized version of Agile adoption, but the in-the-trenches realities of Agile adoption. We incorporate a case-based learning approach which is common in MBA programs [7].

Students are assigned a case study on Agile adoption and prepare a short presentation to be delivered during the class, followed by a mediated class discussion. Table 1 highlights the case study presentations schedule in green and Sect. 6 lists the case studies for the 2016 course editions. The case study papers are Agile adoption experience reports from past XP and Agile conferences. The case studies involves students actively discussing different industry-based case studies that focus on different organizations that go through an Agile adoption process. No single case study describes an easy Agile adoption story; each highlights a different discussion point around adoption or organizational change.

Each student presents their case study once throughout the week for up to 30 mins. The student summarizes the paper and leads a discussion. The students are

asked to first present on who the organization is, who the author is and what their role is within the organization, and what they are trying to achieve or improve in the organization. The class then discusses what should the organization do based on this information. The presenter then describes what the authors actually did and what the outcome of the case study was. Finally, the class discusses if what the authors did made sense, if something different should be recommended, and compares this study with other case studies that have already been presented.

This case-based learning approach enables students to gain an appreciation of how difficult Agile adoption is at an organizational level. By discussing a range of case studies we aim to equip students with knowledge of a broad range of situations that may arise and be able to think critically about where Agile methods can and should be applied in practice.

2.2 Lectures

During the week lectures are delivered that fit into one hour time boxes and consist of presentations and class exercises. Throughout the week we disperse the lectures among case study sessions and group exercises, to keep class activity varied and to ensure the theory is backed up with practical exercises. Table 1 highlights the lectures in yellow.

Agile Manifesto. After an introductory lecture, we present the Agile manifesto and the 12 underlying principles. We focus on the main idea of the manifesto that is: *We are uncovering better ways of developing software by doing it and helping others do it.* We emphasize the importance of the main items of the manifesto, discuss the principles of the manifesto and give some examples to illustrate these principles and values. Finally, we finish the lecture with some of the common misconceptions about Agile methodologies and from our own experience such as *If you're going to adopt Agile development, you should do it 100%* and *Switching to Agile development offers excellent immediate benefits.*

Agile Methods. We present lectures on time-boxed methods in Agile, where we give an overview of both Scrum and XP. For each method we give an overview of the main features, the different practices and roles that team members have, and explain the core values and contributions. In both methodologies we focus on explaining delivering business value with regular steps, monitoring features delivered, and adjusting plans according to results. Then we discuss balancing allowing the business to change their mind while the development team continues to get work done on a stable scope. We present the different team roles, different practices such as sprints/iterations, maintaining a product backlog, planning, daily meetings, and iteration reviews. We emphasize the values of Agile teams: commitment, focus, openness, respect, and courage. Scrum and XP have similar and overlapping structures, roles, and values. There are however some subtle differences that we highlight in the Scrum and XP lectures, for example where XP has a greater emphasis on engineering practices such as pair programming and Test-Driven Development (TDD). We feel it is important to cover both of these time boxed methodologies, as Agile training frequently champions one

method over the other. Our approach is to give students an understanding of what Agile methodologies are available to them, with a view to helping them to think in an Agile mindset and not focus on just the methods. We additionally give overviews of other methods including: Kanban, Lean, Crystal, DSDM, and CRISP-DM.

Release and Iteration Planning. Understanding user stories is a an important aspect to software release planning in Agile. Not only are they used to elicit requirements from customers and communicate ideas among a team, they are used as units of customer value. In Agile, delivering customer value is a priority, and by creating user stories teams can plan releases, as well as iterations, around maximizing value for their customers. We present lectures on how to generate personas (fictional end-users as a focus for delivering value to somebody) and use them in-turn to generate candidate user stories. We then show how to estimate the amount of work a user story might require to be implemented. One of the key things we try and get across is that user stories are not all equal, and that an estimation of the amount of work required to implement one varies from team to team. Estimation is difficult, and requires team discussion and agreement, and we illustrate this idea with students playing Planning Poker[2], among other methods, to estimate animal points (see Fig. 1(a)). We then show how user stories with estimates can be used to plan releases (e.g. a release after 4 week sprints) by selecting a series of user stories that delivers minimum demonstrable value for customers (in order to receive feedback in as short a time as possible). At the same time iteration planning (e.g. weekly) is discussed to show how an Agile team should aim to have working software as early as possible and often. Coupled with the team release planning exercise, our aim is to put students through the motions of becoming customer-focused and in the mindset of team collaboration to achieve goals in an iterative manner. We also discuss how to effectively track progress during release and iteration planning using various techniques such as information radiators (e.g. burn down/up charts).

Guest Lectures. We typically invite expert guest lecturers (industry practitioners or other academics) to deliver specialist content. In particular we have had lectures on Example Driven Development (xDD) (e.g. TDD, ATDD, BDD), Lean & Kanban, change management, and empirical research on how Agile methods are used in practice.

Retrospectives. The final lecture is on retrospectives, where we typically have a guest lecture present and then perform a retrospective exercise with the class. This lecture focuses on the ideas from Derby and Larsen [5] and Kerth [9]. Once the retrospective has completed, the post-course assignment (Sect. 2.4) is explained and handed out and then finally students conduct a course survey which is used for course evaluation purposes.

[2] http://www.planningpoker.com/.

2.3 Group Exercises

One of the key approaches we take with teaching our Agile Methods course is to encourage peer learning and learning-by-doing. To reinforce the lectures students participate in a number of exercises, working as pairs and groups. Table 1 highlights the group exercises in blue.

Communication. We explain to the students how important it is to communicate effectively with customers on Agile projects by illustrating the customer design cartoon[3]. To reinforce this message the students complete a communication exercise – Offing the Off-Site Customer [4]. This exercise involves pairs of students where one acts as a "customer" and the other as a "programmer". The aim of the exercise is for the programmer to elicit requirements from the customer in order to draw a diagram of the product vision on paper (see Fig. 1(b)). The customers are given a drawing that they need to communicate to the programmers to recreate, without visually communicating the drawing. The exercise is played out over two rounds. In the first round the customers must only communicate with the programmer via handwritten text messages on index cards. In the second round, the customers and programmers are allowed to use verbal communication. After each round, the students reflect on the experiences of trying to communicate the drawings between them. The point of this exercise is to suggest that people in close proximity to each other with minimal physical barriers have a better chance of communicating effectively. We encourage the students to think about their own work place and to find a way to set up environments to encourage regular and meaningful collaboration.

Prototyping. To get a feel of prototyping with time-boxed methods, students complete the Marshmallow Challenge[5] and are asked to prototype a fully functioning coffee machine out of cardboard based on ideas from Cockburn [4] (see Fig. 1(c)). The goal of the coffee machine exercise is to try Scrum for an iteration and then walk a mile in a product owner's shoes as part of a second iteration. The students are divided into teams (up to four). During the first iteration the teams have 7 min to write stories about how the machine will work and prepare materials (e.g. boxes, tape, scissors, cups, water). For each iteration they have 5 min to plan what stories they will implement, followed by 10 min to design and implement the stories, and finally a group presentation to the class. The purpose of this exercise is to get the students to work in a simulated environment as a team in a time-boxed manner, and to understand that prototyping and early releases only need to demonstrate a concept to a customer to deliver value.

Agile Debate. To help students gain a better understanding of the differences between the Scrum and XP methodologies we ask them to perform a debate. The class is separated into two sides: Scrum and XP. We give them two well-known,

[3] http://projectcartoon.com.
[4] http://www.jamesshore.com/Presentations/OffingTheOffsiteCustomer.html.
[5] http://www.tomwujec.com/design-projects/marshmallow-challenge/.

and highly contrasting, quotes, from Martin Fowler[6] and Ken Schwaber[7] as the basis for their arguments. The students use the session to come up with their own arguments and perform the debate with the lecturer as the adjudicator. The aim of the debate is for the students to understand that there is no silver-bullet when it comes to applying and adopting Agile methods.

Kanban Game. To help students gain an appreciation of the Kanban method we get them to play the getKanban[8] board game (see Fig. 1(d)). getKanban is a physical board game designed to teach the concepts and mechanics of Kanban for software development in a classroom setting. Each team can have up to six people. Each team has a playing board representing a Kanban task board, and a collection of story cards representing work to be done. Teams compete to maximise profit by optimizing the flow of work. We simulate the game for up to 21 days. During the game the teams construct charts based on data from the game including a Cumulative Flow Diagram, a Run Chart, and a Lead Time Distribution Chart. To help make the game more realistic there are a number of simulated events that occur throughout the game that challenge the teams (e.g. a developer needs to attend a training course) and require them to make various system design, prioritization, and resource allocation decisions. We allow a couple of hours to play the game and have a debrief session at the end to help students understand the intricacies of the method.

Team Release Planning. Building on the accompanying lecture sessions, students carry out a team release planning exercise which covers most of Thursday. This puts into practice everything they have been taught about Agile. In this exercise, we split the students into groups of no more than four per team. In this exercise, we do not mandate any team structure – we allow the students to self-organize, much like a real Agile team would be expected to do. The lecturer sets a particular domain area (e.g. solve London's transport issues) in which each team can then pick their own idea for a small product or service. They create a release plan (including personas and user stories) over four weeks and four time-boxes on a card wall, with the aim of being able to release a first version of their product to a customer after the four weeks. At the end of the exercise, each team presents their release plan to the rest of the class (see Fig. 1(e)).

Retrospective. A retrospective on the course is performed on the last day where an external guest lecturer usually facilitates. Students record their thoughts about the course on post-it notes into three categories: positive, could be better, and aspects that were a surprise. Students place the ideas into different days of the course on a card wall based on the timetable (see Fig. 1(f)). The facilitator walks through the card wall and identifies and discusses key themes. The aim of this exercise is for students to reflect upon what they have learned.

Learning Stand-up Meeting. At the end of each day the students perform a learning stand-up meeting similar to a daily stand-up meeting (see Fig. 1(g)).

[6] http://martinfowler.com/bliki/FlaccidScrum.html.
[7] http://kenschwaber.wordpress.com/2010/06/10/waterfall-leankanban-and-scrum-2/.
[8] https://getkanban.com/.

(a) Estimation: planning poker with animal cards.

(b) Communication: customers & programmers diagrams.

(c) Prototyping: cardboard coffee machine.

(d) getKanban board game.

(e) Team Release Planning Exercise.

(f) Retrospective on the class.

(g) Learning Standup that happens at the end of each day.

Fig. 1. Agile methods course – some sample class exercises that simulate some key points about different Agile practices including: estimation, communication, prototyping with cardboard, Kanban, release planning, class retrospectives, and daily learning stand up meetings.

The stand-up meeting is for each student to address the questions like they would in a daily stand-up, hence fostering Agile team values of openness, respect, and courage. The questions focus on what have they learned during the day and what they would like to learn. Students write answers on post-it notes, present them to the group, and then put them on a learning card wall.

2.4 Course Assignment: Essay and Release Plan

The course is assessed with an assignment where students are given six weeks to develop a mock four-week release plan and complete an essay. The *release plan* is based on a fictitious product idea (e.g. develop an application for a hospital to help support children who suffer from a medical condition like autism). The release plan is documented as a report, outlining the different personas, user stories (based on one persona), and the release plan itself, similar to the team planning exercise performed in class. They need to include the rationale for deciding the team's capacity for each sprint, and why they think that this release plan makes the most sense for the customer. Developing the release plan in the assignment aims to assess what the students learned in class, and we ask them to reflect on this aspect comparing with their experience on the team release planning exercise. The *essay* involves comparing and contrasting different Agile adoption paths from two of the case studies (Sects. 2.1 and 6). One question that students are asked to address is, "is there a one-size fits all Agile adoption strategy?" The assignments are assessed following a marking guide[9] where all submissions are awarded a numerical grade between 0 and 100, interpreted as follows: 0 and 49 denotes a fail, 50 and 69 denotes a pass, and 70 and 100 denotes excellence. Most students are awarded a pass, some with excellence, and few with fail; and grades are released approximately six weeks from submission. Students can defer submitting the assignment and wait for a later edition, but this is rare.

3 Discussion

Teaching the AGM course over the last decade has given us in-depth experiences from which to draw upon, that we would like to share. Throughout the duration of this course, we have gathered student feedback to help inform and evolve the course along the way, as well as having gathered formal feedback from students, some of which we now discuss.

Lectures. One of the biggest challenges in designing this course is catering for the intensive nature of the course delivery. While there are just over 30 h of face-to-face time allocated to the course, we were conscious not to overwhelm students with only lectures. To this end, about a third of the class time was allocated for lectures, broken up with exercises and case study discussion sessions. While lectures are useful for delivering information to students, our key aim was to enable the Agile mindset. In this case, we put further emphasis on learning-by-doing with hands-on exercises and class discussion. For each lecture, we ensured that a relevant exercise or case study followed. Keeping the lectures to a planned limit of only one hour per session we felt also deferred any feelings of fatigue or boredom, which is vitally important in a short but intensive learning environment. The students found the theory was important to learn and appreciated the lecture content on empirical studies of Agile project teams which showed evidence about the use of different practices within industry.

[9] http://www.cs.ox.ac.uk/softeng/handbook/examinations.html.

Group Exercises. The exercises were carefully chosen to help the students put into practice, or to help them quickly understand, the lecture material. As discussed above, the lectures were normally paired with relevant exercises or case studies. For example, to take the learning from the release planning lecture and put it into practice, we would follow the lecture with an exercise on story estimation. Each of the exercises aimed to teach important aspects about the different Agile methods. The communication exercise highlights the importance of fast and frequent customer feedback. The prototyping exercise looks at how Agile teams are formed and how to respond to change. The estimation exercise put the use of abstract story points into practice on non-software artifacts to help understand that estimation is a team effort and not a formula that is uniformly applied. The Kanban board game exercise aims to demonstrate the Kanban method and allows students to put the method in action to understand workflow.

Team Release Planning Exercise. The team release planning exercise involves putting into practice the learning from all the previous lectures and exercises. We encouraged the students to self-organize and gave them reminders and guidance on the course material. We mainly left them to make their own decisions on how to plan their product releases. This exercise always proves to be the most challenging for the students, as it was designed to simulate the real planning of a hypothetical product or service, where the exercise often took many hours or a whole day. The task also requires a level of creativity that many students were uncomfortable with, but it was essential to move them away from their comfort zone in order to get into an Agile mindset where all members of a team should be able to feel they can contribute.

Assignments. The assignment tasks mirrored much of what was taught during the class, where students are asked to write a short essay comparing and contrasting two case studies, and then to create a release plan similar to their team release planning exercise. The essay question was generally straight forward as the case study presentations and debates prepared students well for this part of the assignment. The release planning exercise, however, proved challenging for many. The main difficulty in this task was that in class it was done as a group exercise, while in the assignment the students were asked to do a similar plan but on their own. Some students took the initiative to simulate the group environment by asking work colleagues to carry out the collaborative parts of the exercise, such as estimation. Others on occasions, however, fell back into old habits or forgot the learning in class and strayed on a tangent to what was expected. The creativity aspect of the release planning exercise, on occasions, proved problematic, where students either could not come up with an idea for a product that was suitable to generate a good number of personas and user stories, or that sometimes a student would get carried away and produce an assignment submission that was all about their great idea, but little in substance for demonstrating what they learned in class. What we learned is that setting such an assignment, care should be taken to ensure the students remember they need to focus and demonstrate their learning in their submissions.

Practical Approach to Teaching Agile. The design of the class delivery gives students a practical approach to an Agile environment. The time-boxed class sessions planned throughout the week reflects how sprints or iterations are planned in time-boxed Agile methods such as Scrum and XP. We used pairstairs[10] to encourage students to pair with their colleagues during the week, much like when applying the XP practice of pair programming. The daily learning stand-up reflects the practice of daily stand-up meetings in a Scrum or XP team. The use of visual cues around the classroom to learning material, but also to the collective class experiences such as in the prototyping and the release planning exercises, provides the tactile experience that Agile working environments give. Finally, getting the students to perform the retrospective exercise gives them the experience of participating in a realistic retrospective.

Evolving Agile Teaching Content. One of the things we have observed that is changing in the students over the last few years is that more students attending the course identify themselves as already having Agile training, or as being Agile practitioners in their organizations. This reflects what we see today in the software industry – that Agile is no longer a niche, and is an expected workplace practice in software engineering teams. To this end we have taken the opportunity, through the retrospectives, to be Agile in our planning of the course itself, and to take on board "customer feedback" from our students and make continual changes based on this feedback. For example we have introduced new games and techniques into the course such as the getKanban board game and used more recent and up to date Agile adoption case studies over time.

Student Feedback. Over each iteration of the Agile Methods course, as discussed earlier, feedback is gained by putting students through a retrospective exercise at the end of the teaching week to help inform any changes to the next instance of the class. Alongside this, students have been returning student feedback questionnaires since 2010, where overwhelmingly the feedback has been positive. The students were asked to rate between 1 (strongly disagree) and 5 (strongly agree) their level of agreement on 12 statements, for example:

- The lectures added significant value to the course material
- The lectures included valuable contributions from other students in class
- The exercises helped me to understand the topics covered in the lectures

The aggregate and average score over the time period for which we have data is 4.32 out of 5 based on 132 completed questionnaires[11] (see Fig. 2). The last three editions of the course feedback scored the most highly and were above the SEP all courses average of 4.55, at 4.6, 4.73, and 4.66 respectively. This feedback shows some perceived evidence that the course structure and content has matured where students are generally satisfied with what is being delivered.

While the feedback was generally positive, there were however some negative comments on the course content in the feedback questionnaires. Many of these

[10] http://pairstair.com.

[11] Statements and raw data located in https://tinyurl.com/AGM-Student-Feedback.

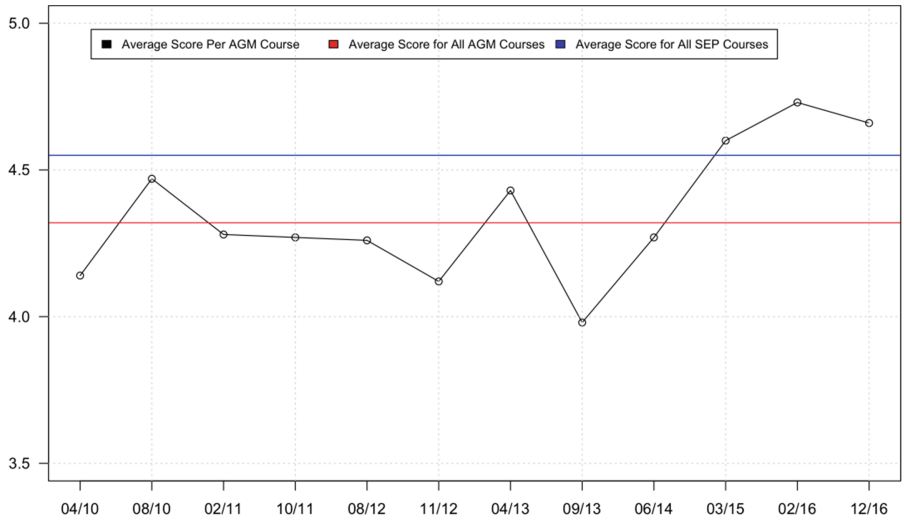

Fig. 2. AGM course evaluation as perceived feedback by students since 2010. Black – average score for each AGM course by the students. Red – average score for all AGM courses. Blue – average score for all SEP courses. Last three editions scored the highest. (Color figure online)

focused on the fact that given such a short space of time allocated in class, there was far too broad material in the Agile space to impart in further depth onto the students. In particular, we received comments such as:

"The course content is not enough to stretch 5 days."

"The large amounts of material made timing difficult."

This is a clear acknowledgment that the amount of material around Agile methods makes teaching the subject very difficult, especially in an intensive teaching environment. There was some skepticism about the game-like exercises, however, on further reflection with students several weeks after the course had completed even those students acknowledged that they had taken on board the learning from the exercises when they returned to their own organizations to share their new-found knowledge around Agile. Some of the positive comments we received included:

"Excellent way of teaching! You have expanded my horizon and gave me an excellent introduction to Agile."

"One of the best courses I have studied at Oxford. The lecturer and their use of guests made the course better and maybe of benefit to other courses! Initially I had doubts about the lecturer coming from industry but for this course it works better as they can draw on experience."

"A useful course with a timeliness of current industry trends."

"Good and helpful lecturers. The idea of students studying and presenting a case study is brilliant and helped a lot with understanding and discussing."

We have kept the Agile Methods course content timely by consciously putting Agile into practice in how we prepare and deliver the Agile Methods course itself. Feedback, in particular via the retrospective exercise, has allowed us to keep the course up to date with student and industry needs.

4 Related Work

Related work has mainly focused on teaching Agile to undergraduate and graduate students as part of computer science and software engineering curriculum's. The majority of these courses are typically group based projects, last 10–16 weeks, and teach Scrum and or XP. Our work reports on teaching a novel Agile methods course to *software engineering professionals* that are already working within industry and likely already have a degree, potentially in a computing subject, or have extensive software industry experience.

Lu and DeClue [12] discuss how Agile skills improve the marketability of new graduates. They also highlight the challenges posed in teaching Agile to undergraduates that stem from prerequisite experience and maturity. These challenges include fostering Agile approaches to skills such as communication, self-organization, and teamwork, where students who have less experience in a workplace may find mastery of these skills more difficult.

A panel at SIGCSE 2016 [3] raised a number of issues for teaching Agile methods in software engineering courses at a variety of computer science programs. The panel focused only on undergraduate university teaching (100 to 400 levels), hence novices to Agile with limited development experience.

Anslow et al. [1] reported their experience of teaching Agile methods to undergraduate and graduate students and presented a course outline along with associated teaching materials. They recommended not to teach the course to different levels simultaneously due to the nature of different levels of assessment required, abilities of the students, and additional administrative overheads.

Steghöfer et al. [16] reported on their efforts to improve teaching Agile, and Scrum in particular. They aimed to teach in a realistic manner but without encountering the technical difficulties of creating a real product by introducing exercises decoupled from software, such as LEGO Scrum.

Kropp et al. [10,11,13] looked at the status of Agile in education and industry and proposed a competency model on which to base integration of Agile into undergraduate teaching at two different universities. They found the most difficult competencies to teach are Agile values and management practices which they put significant emphasis on. Our AGM course also focuses on values and management practices and we have a complimentary course that focuses on Agile engineering practices[12] such as TDD and continuous integration.

Soundararajan, et al. [15] developed an advanced graduate-level course (to non-software professionals) in Agile software engineering at Virginia Tech. Their

[12] http://www.cs.ox.ac.uk/softeng/subjects/APE.html.

course has similarities to our approach where they focus on Agile product development, host guest talks from industry experts, and encourage students to present and debate Agile case studies within the class.

5 Conclusions

For today's computer science students who look towards entering a career in software engineering, skills beyond programming and technical excellence are essential. For any new graduate entering the tech industry, knowledge of Agile is essential. We hope that by sharing our extensive experiences in teaching Agile we can help foster excellence in Agile methods education in formal educational settings, such as in high school, university degree programs, and perhaps also in industrial training. From our experiences in teaching the Agile Methods course at the University of Oxford, we can extract many aspects of what we taught to graduate students that could be applied in any Agile teaching course. We believe that putting Agile theory into practice with a hands-on approach will lead to more effective learning. Based on the material reported in this paper other academics who wish to run similar courses can learn from our experiences.

6 Agile Methods: Case Study Papers for 2016

P1. M. Albisetti. Launchpad's quest for a better and agile user interface. In *XP*, pages 244–250. Springer, 2010.
P2. K. Boekhout. Mob programming: find fun faster. In *XP*, pages 185–192. Springer, 2016.
P3. C. Fry and S. Greene. Large scale agile transformation in an on-demand world. In *AGILE*, pages 136–142. IEEE, 2007.
P4. S. Hublikar and S. Hampiholi. Pause, reflect and act, the pursuit of continuous transformation. In *XP*, pages 201–208. Springer, 2016.
P5. M. Keeling. Put it to the test: Using lightweight experiments to improve team processes. In *XP*, pages 287–296. Springer, 2010.
P6. T. Little, F. Greene, T. Phillips, R. Pilger, and R. Poldervaart. Adaptive agility. In *AGILE*, pages 63–70. IEEE, 2004.
P7. S. McCalden, M. Tumilty, and D. Bustard. Smoothing the transition from agile software development to agile software maintenance. In *XP*, pages 209–216. Springer, 2016.
P8. B. Pieber, K. Ohler, and M. Ehegötz. University of Vienna's u:space turning around a failed large project by becoming agile. In *XP*, pages 217–225. Springer, 2016.
P9. D. Poon. A self funding agile transformation. In *AGILE*, pages 342–350. IEEE, 2006.
P10. M. Rajpal. Lessons learned from a failed attempt at distributed agile. In *XP*, pages 235–243. Springer, 2016.
P11. N. Robinson. A technical story. In *AGILE*, pages 339–343. IEEE, 2007.

P12. K.H. Rolland, V. Mikkelsen, and A. Næss. Tailoring agile in the large: Experience and reflections from a large-scale agile software development project. In *XP*, pages 244–251. Springer, 2016.
P13. C. Sudbery. How XP can improve the experiences of female software developers. In *XP*, pages 261–269. Springer, 2016.
P14. A. Takats and N. Brewer. Improving communication between customers and developers. In *AGILE*, pages 243–252. IEEE, 2005.
P15. I. Tsyganok. Pair-programming from a beginner's perspective. In *XP*, pages 270–277. Springer, 2016.
P16. B. Victor and N. Jacobson. We didn't quite get it. In *AGILE*, pages 271–274. IEEE, 2009.

Acknowledgments. Thanks to Jeremy Gibbons and Jim Davies from the Software Engineering Programme at the University of Oxford for their support. Thanks to guest lectures by Antony Marcano, Duncan Pierce, Lazaro Wolf, and Robert Biddle. Thanks to Rob Chatley for expert advice. Thanks to Clint Sieunarine and Ross Gales for being teaching assistants.

References

1. Anslow, C., Maurer, F.: An experience report at teaching a group based agile software development project course. In: SIGCSE, pp. 500–505. ACM (2015)
2. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. Addison Wesley, Reading (2004)
3. Campbell, J., Kurkovsky, S., Liew, C.W., Tafliovich, A.: Scrum and agile methods in software engineering courses. In: SIGCSE, pp. 319–320. ACM (2016)
4. Cockburn, A.: Agile Software Development: The Cooperative Game. Addison Wesley, Boston (2006)
5. Derby, E., Larsen, D.: Agile Retrospectives: Making Good Teams Great. Pragmatic Bookshelf, Raleigh (2006)
6. Hazzan, O., Dubinsky, Y.: Why software engineering programs should teach agile software development. SIGSOFT Softw. Eng. Notes **32**(2), 1–3 (2007)
7. Lee, S.H., Lee, J., Liu, X., Bonk, C.J., Magjuka, R.J.: A review of case-based learning practices in an online MBA program: a program-level case study. Educ. Technol. Soc. **12**(3), 178–190 (2009)
8. Highsmith, J.: Agile Software Development Ecosystems. Addison Wesley, Boston (2002)
9. Kerth, N.L.: Project Retrospectives: A Handbook for Team Reviews. Dorset House Publishing Co., New York (2001)
10. Kropp, M., Meier, A.: Teaching agile software development at university level: Values, management, and craftsmanship. In: International Conference on Software Engineering Education and Training (CSEET), pp. 179–188. IEEE (2013)
11. Kropp, M., Meier, A.: New sustainable teaching approaches in software engineering education. In: EDUCON, pp. 1019–1022. IEEE (2014)
12. Lu, B., DeClue, T.: Teaching agile methodology in a software engineering capstone course. J. Comput. Sci. Coll. **26**(5), 293–299 (2011)

13. Meier, A., Kropp, M., Perellano, G.: Experience report of teaching agile collaboration and values: agile software development in large student teams. In: International Conference on Software Engineering Education and Training (CSEET), pp. 76–80. IEEE (2016)
14. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Pearson, Upper Saddle River (2001)
15. Soundararajan, S., Chigani, A., Arthur, J.D.: Understanding the tenets of agile software engineering: Lecturing, exploration and critical thinking. In: SIGCSE, pp. 313–318. ACM (2012)
16. Vogel, B., Kilamo, T., Kurti, A.: Teaching distributed agile development to software professionals: a flexible approach. In: European Conference on Software Architecture Workshops, ECSAW, pp. 31:1–31:8. ACM (2015)

Are Software Startups Applying Agile Practices? The State of the Practice from a Large Survey

Jevgenija Pantiuchina[1], Marco Mondini[1], Dron Khanna[1], Xiaofeng Wang[1(✉)], and Pekka Abrahamsson[2]

[1] Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
{Jevgenija.Pantiuchina,marco.mondini,dron.khanna,xiaofeng.wang}@unibz.it
[2] Norwegian University of Science Technology, 7491 Trondheim, Norway
pekkaa@ntnu.no

Abstract. Software startups operate under various uncertainties and the demand on their ability to deal with change is high. Agile methods are considered a suitable and viable development approach for them. However, the competing needs for speed and quality may render certain agile practices less suitable than others in the startup context. The adoption of agile practices can be further complicated in software startups that adopt the Lean Startup approach. To make the best of agile practices, it is necessary to first understand whether and how they are used in software startups. This study targets at a better understanding of the use of agile practices in software startups, with a particular focus on lean startups. Based on a large survey of 1526 software startups, we examined the use of five agile practices, including quality related (regular refactoring and test first), speed related (frequent release and agile planning) and communication practice (daily standup meeting). The findings show that speed related agile practices are used to a greater extent in comparison to quality practices. Daily standup meeting is least used. Software startups who adopt the Lean Startup approach do not sacrifice quality for speed more than other startups do.

Keywords: Software startups · Agile practice · Lean startup · Minimum viable product · Pivot · Quality vs Speed

1 Introduction

Startups are organizations designed to create new products or services under the conditions of extreme uncertainty, which constantly seek repeatable, profitable and scalable business models and aim at rapid growth [1,2]. Software startups are startups that have a primary focus on developing new and innovative software-intensive products or services from which business value is created. Even though sharing common characteristics with other types of startups, such as resource scarcity and a lack of operational history [3], software startups are often caught up in the wave of technological change frequently happening in software industry, such as new computing and network technologies and devices [4].

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 167–183, 2017.
DOI: 10.1007/978-3-319-57633-6_11

As the ability to accommodate frequent change is essential in the startup context, agile methods have been considered the most suitable process model since they enable software startups to embrace change, and allow development to adapt to business strategies [5]. Fast release with an iterative and incremental approach shortens the lead time from idea conception to production to market, which is especially important for software startups as "done is better than perfect" and "move fast and break things" are the slogans or mantras that they follow in order to respond to the challenges they are confronted with [6].

However, since software startups are constantly under the huge pressure of time-to-market and need to move really fast, product quality may be treated with a low priority and technical debt is accumulated to gain the speed to market [7]. As a result, certain agile practices that ensure the quality of software, such as refactoring and test-driven development, may not be considered viable practices for software startups, especially at the early stages [8]. But the accumulated technical debt, if not paid back in time, will eventually slow down the development speed [7], which means software startups cannot afford to ignore quality and related engineering practices as they progress through the stages of development.

The adoption of agile practices can be further complicated when software startups follow the Lean Startup approach to develop their business, which puts even more emphasis on quick prototyping [9], testing prototypes with potential customers, and getting early feedback. The use of Lean Startup approach may intensify the so-called "developers dilemma"—the balancing act between quality and speed to achieve fast product iteration [10], and render the agile practices related to quality even less viable to software startups.

To understand how software startups can better use and benefit from different agile practices for their needs for quality and speed, it is important to understand firstly if and how software startups are currently using agile practices. The existing software engineering literature has accumulated a growing body of knowledge on the application of agile methods in established companies, large or small. However it casts very few lights on the use of agile practices in software startups, let alone in the startups who adopt the Lean Startup approach. Based on this observation, the study presented in this paper targets at understanding the state of the practice of agile practices in software startups, and the potential influence of the Lean Startup approach on the use of agile practices. The overall research questions that guide our study are:

RQ1: *Are software startups applying agile practices?*
RQ2: *Are software startups that adopt Lean Startup applying agile practices?*

To depict the state of the practice, we utilize the data collected in a large online survey conducted from September 2013 to September 2014. Based on the responses from 1526 surveyed software startups worldwide, we could obtain a good understanding of the state of the practice of agile practices applied in software startups.

The rest of the paper is organized as follows. Section 2 reviews the related work that has been conducted so far to understand the use of agile practices

in software startups. In Sect. 3, we explain how the online survey is utilized to answer the research questions. The findings are presented in Sect. 4 and further discussed in Sect. 5, together with the reflection on the limitations of and validity threats to the study. The paper ends with Sect. 6 in which potential future work is outlined.

2 Related Work

2.1 Agile Methods in Software Startups

The emergence of agile methods was a response to the inability of heavy-weight, waterfall-like development methodologies to allow software organizations to respond to change. Popular agile methods, such as Scrum and XP, have been adopted by both small and large companies worldwide over the years, rendering agile a mainstream software development approach [11]. At their core, agile methods focus on incremental and iterative development. The nimbleness and flexibility allowed by different agile practices, such as short iterations, continuous integration, etc., enable software organizations to address change effectively [12,13]. The effective adoption and use of agile methods in established companies have been manifested in a growing body of agile research [14–16].

When the context is switched to software startups, the picture is less clear-cut. Some studies suggest in a general manner that agile methods are viable and suitable for software startups (e.g., [17,18]). For example, Duc and Abrahamsson [9] find that four out of five startups they studied have adopted agile development processes. However, these studies do not specify clearly which particular agile method or agile practices have been used in software startups.

Other studies suggest a different picture. Coleman and O'Connor [5] argue that startups are creative and flexible in nature and are reluctant to introduce process which may hinder their natural attributes. They have very limited resources and typically wish to use these resources to support product development. Giardino et al. [18] observe that, to quickly validate the product in the market, software startups tend to use agile methods, but in an ad-hoc manner. Yau and Murphy [8] go further and contend that, given that the communication and cooperation dynamics in startups are very different from more established companies and the fact that the initial focus of a startup might be significantly different from its final objective, even the agile approach seems to impose too much rigidity and process on them. Without denying that agile methods offer clear benefits to startups over some of the more traditional methods, the authors question whether they are appropriate in tackling the problems faced by startups. Doubts are cast on the usefulness of agile practices including test-driven development, pair programming, user stories, velocity and backlogs [8].

2.2 Lean Startup and Agile Practices

The Lean startup approach is considered a variant to agile methods in software engineering literature [18]. It advocates the identification of the most risky parts

of a software business and the use of Minimum Viable Products (MVPs) to systematically test them and change the course of the development if needed. According to Ries [1], a MVP is "[the] version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort." MVPs should be the main focus of both business and product development activities in software startups [9]. The strategic change is termed *pivot* in Lean Startup.

Even though the Lean Startup approach is seen as a recent advancement in agile community, and regarded by some as a more "extreme" agile approach than extreme programming (XP) or Scrum [19], difference between the two has been argued. Agile methods seem to be able to prescribe on how to develop a working software faster, but are unable to provide the answer to what product should be developed in the first place [20]. Although agile methods advocate to build software iteratively, they only work when problems are known to the stakeholders. Instead, startups typically are looking for right problems to solve and need to figure out who are their customers [2]. Lean Startup advocates startups to build products iteratively and get early feedback to test riskiest assumptions about their business models. The combined use of agile and Lean Startup seems a sensible approach for software startups.

The research conducted by Duc and Abrahamsson [9] is focused on different types of MVPs that software startups utilize and what are their main purposes. They argue that the adoption of MVP might be influenced by many contextual factors, and one most relevant factor is the product development methodology. They further suggest that the continuous integration—one of the agile practices—might be the impetus for the popular adoption of evolutionary prototypes and single-feature MVP in four of the five cases they studied.

However, Yau and Murphy [8] contend that certain agile practices may not be in consistency with the primary focus of software startups that adopt the Lean Startup approach. Quality is important for a software startup but cost and time may be larger deciding factors. A small scale startup that has not obtained much funding will probably have a short runway, and thus a limited amount of time and money. The priority in this case should be to create an MVP, which may lack in quality but is functional enough to show to investors. Terho et al. [10] also argue the need of balancing between quality and speed in creating MVPs, the intensified "developers dilemma" faced by software startups. As a consequence, the agile practices that are focused on quality of software, such as test-driven development and refactoring, may be compromised or even not taken on board.

3 Research Method

3.1 Survey Questions

This study utilizes a large online survey that was conducted between 2013 and 2014. The original survey explored various aspects of startups and covered a large set of questions. The authors had the opportunity to access the survey data and select the questions that were pertinent to the purpose of this study.

Table 1 shows the list of questions used in this study, as the result of the selection process. The questions are mainly divided into three categories:

- questions related to the demographic information of the respondents and the background information of the surveyed software startups;
- questions related to agile practices, which form the core category. We used the list of agile practices reported in the 10th annual agile report from VersionOne [11] as the commonly accepted agile practices. The original survey includes five questions relevant to agile practices: regular refactoring, test first, frequent release, agile planning, and daily standup meeting. All are close-ended questions. Four are ordinal and the one about daily stand-up meeting is binary. All of them require a single answer and are not mandatory.
- questions related to the Lean Startup approach, which allow us a more focused examination of the use of agile practices in software startups that adopted the Lean Startup approach. We identified three questions from the original survey which indicate whether a startup is following the Lean Startup approach or not. These questions reflect the key Lean Startup concepts: hypothesis-driven, MVP and pivot.

Table 1. Survey questions used in the study

Background questions	
About respondent	Select your gender
	How old are you?
	What is your motivation with this startup?
About startup	What kind of startup are you a part of?
	What is the total size of your team?
	How many founders are there on your team that own a significant piece of equity?
	What's the stage of your primary product?
	How many core features does your product have?
Agile practice related questions	
Regular Refactoring	How often do you refactor code?
Test First	When do you start writing tests?
Frequent Release	What is the frequency of your product release cycle?
Agile Planning	How far ahead do you plan your product development pipeline?
Daily Standup	Do you do daily stand up meetings?
Lean Startup related questions	
Hypothesis-driven	We identified the riskiest hypotheses about our business in order to test them first
MVP	We built minimally viable products to test our hypotheses
Pivot	How many pivots have you had?

3.2 Data Cleaning and Validation

To ensure the quality and validity of the survey data, we went through a careful data cleaning and validation process on the original dataset, which is described in detail in this section. The process was mainly automatized using R software package. Additionally, we have removed suspicious data entries manually.

To start with the data cleaning process, we set the threshold of 50 (out of a total of 278 original survey questions) as the minimum number of answered questions that a data entry should contain. All the rows with less than 50 answers were removed from the dataset. Afterwards, we merged rows if they were answered by the same person and for the same startup, because the survey collection application saved the data as a separate entry if the survey was interrupted and then restarted again. We also removed duplicate columns that might have been introduced during the data exporting process. We also fixed various obvious errors that may be attributed to the original survey design or data exporting process.

After this rudimentary step, we started automatized and manual data cleaning column by column (question by question). We removed all the rows where startup names were missing, to ensure that respondents have answered the questions by referring to specific startups. We also removed the rows with empty emails. We have decided to exclude from the sample the answers referring to the same startup but answered by different respondents because there was not a convincing rationale as to which answer to keep: the CEO's or the developer's. Each of them has its pros and cons. Fortunately there were not many duplicated startups. We also checked startup names, emails and websites and further removed the rows with suspicious values, for example, the answers that containing "none", "not", "test", "xyz", "untitled", etc. We then applied the regular expressions to all the columns that had a fixed set of values to further remove invalid answers. For example, if a question was Boolean, we ensured that only "0"s and "1"s were in the corresponding column. In the last step, we printed all the possible values for each closed question and ensured that only the valid answers were present in the dataset.

After the initial cleaning, we checked the validity of the data using a set of validation cases that we discovered based on a close inspection of all the survey questions. The validation cases detected a set of unrealistic, impossible, invalid combinations of answers which rendered certain data entries invalid, which in turn were removed from the dataset. All the validation cases we used are described in an online document that can be accessed at https://figshare.com/s/08c35ec98fd85e827594

The original dataset had 10171 entries. After applying the data cleaning and validation process, the final cleaned dataset has a sample size of 1526. By performing such strict data cleaning and validation steps, we may have removed some valid entries unintentionally. But removing some valid entries is a trade off that is worth making in order to obtain a clean dataset to conduct the data analysis.

3.3 Data Analysis

To answer the research questions posed in Sect. 1, we analyzed the data in two steps:

Step 1: To answer RQ1, firstly the structure of the five questions related to agile practices was analyzed using exploratory factor analysis. Two factors fit the model and the practices group in pairs: regular refactoring with test first, and frequent release with agile planning. Instead daily standup meeting does not show a significant correlation with any of the two factors. Therefore three dimensions can be defined to group the five agile practices: quality (regular refactoring and test first), speed (frequent release and agile planning), and communication (daily standup meeting). Next the internal consistency between the two items in the quality and speed dimensions was analyzed using Cronbach's alpha. However a low level of reliability estimates ($\alpha = 0.41$ and $\alpha = 0.50$ respectively) was obtained, which meant that the two items within each dimension were not suitable to aggregate. Therefore, the further analysis was conducted on each individual agile practice, rather than at the group level. To allow a sharper comparison, for each agile practice, we divided the startups into using the practice vs. not using it based on their answers to the question. In this way we converted the four agile questions that were categorical (ordinal) into binary. We examined the frequency of the use of five agile practices in the surveyed startups. Since software startups at different product development stages may adopt agile practices differently, we further investigated the difference using Chi-square. The hypothesis for each of the agile practices can be formulated as the following:

H_{a1}: There is significant difference in the use of [the agile practice] among software startups at different product development stages.

Step 2: The focus of this step was to analyze the use of agile practices in the software startups that adopted the Lean Startup approach, in order to answer RQ2. To identify lean software startups in the sample, we used the three questions related to the Lean Startup approach, as explained in Sect. 3.1. The software startups that answered "yes" to the first two questions and have pivoted at least once were considered adopting the Lean Startup approach therefore lean software startups. 229 out of 1526 are lean startups. The use of the five agile practices in these lean startups was compared to that in the rest of the whole sample, to understand if there was difference in agile practice use between the two sub samples. For this purpose again Chi-square tests were used. The hypothesis under the test regarding each agile practices can be formulated as the following:

H_{a2}: There is significant difference in the use of [the agile practice] between lean software startups and non lean software startups.

Since pivot is an important aspect of software startups, we also examined the number of pivots the surveyed startups made as part of Step 2 analysis.

The data analysis process was conducted using R software environment.

4 Results

The cleaned dataset contains information about 1526 software startups, provided by 1526 respondents who either founded or worked for these startups. Not surprisingly only a very small percentage (8%) are females in comparison to the much larger percent of males (76%, the remaining 16% didn't reveal gender information). The age of these respondents spread from 18 to 72 (based on 1219 cases that contain age information), with a mean of 34 and a median of 32 (sd = 9.58). A slight majority of the respondents (52.3%) have the age between 25 and 35. It is intriguing to understand what motivated the respondents to found or work for these startups. As expected the majority of answers reflect an entrepreneurial mindset: "Build a Great Product" covers the 52% of the motivations, followed by "Change the world" (29%). "Make a Good Living", "Get Rich" and "Create a quick flip" are motivations for only less than 20% of the respondents.

Regarding the types of these software startups, more than half of them (877) are working on web-based products. 264 software startups provide both web and mobile solutions. Mobile applications are the focus of 171 startups. Only 65 startups provide non web-based software solutions. The remaining 149 startups either work on products where software plays a less significant role or did not provide specific information regarding the types of their startups.

1461 software startups answered the question "What is the total size of your team?" with meaningful values. The distribution of the sample is skewed right significantly, with 81.2% of the software startup teams with less than 9 members. The mean of the team size is 7.23 and the median is 4 (sd = 19.15). When the number of founders is concerned, even though we could not obtain the direct data from the survey, we could infer from the question "how many founders are there on your team that own a significant piece of equity?" that most often an entrepreneurial team has two co-founders that have significant equity of the company, followed by 1-significant-founder and trio co-founder teams.

The distribution of the software startups across product development stages is shown in Fig. 1. It can be seen that it follows a normal distribution, with software startups that have functional products with limited users as most common, and those with mature products as the minority. A closer look at the number of core features that these products have reveals that the average number of the core features of a product is 5 ($mean = 5.2, median = 4$, sd = 4.07). 72% of the startups work on products that have 5 core features or less.

Product stage	Number of startups
Concept	121
In development	230
Working prototype	267
Functional product with limited users	721
Functional product with high grow	123
Mature product	60

Fig. 1. Startup distribution with respect to product stages

4.1 Agile Practices in Software Startups

Two agile practices, regular refactoring and test first, allow software startups to focus on the quality of their products. 1240 startups responded to the question related to regular refactoring, and 1273 to test first. As shown in Fig. 2, regarding refactoring, slightly less than 45% of the startups do care about the quality and refactor the code every few weeks or even once a week. However, a bit more than one fourth of those rarely or never do refactoring. If refactoring "once a week" and "every few weeks" are considered regular therefore an agile practice (blue bars in Fig. 2), the other options indicate that regular refactoring is not practiced in the startups. It can be concluded that a slight majority of the startups surveyed are not doing regular refactoring.

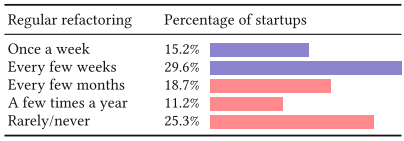

Fig. 2. Startup distribution with respect to the frequency of code refactoring (Color figure online)

Similar results are shown in the test first practice. It is evident from Fig. 3 that around 32% of them are writing tests as soon as they write features, therefore practicing test first (blue bar). However, again one fourth of the startups never write tests. Among the other options, "as soon as we know we're going to keep a feature" indicates clearly the test first practice is not used. Even though we could not interpret properly the options "as soon as we reach a legal agreement with a customer" and "other" due to a lack of access to the original survey design, we could still conclude that the majority of the startups surveyed do not adopt the test first practice.

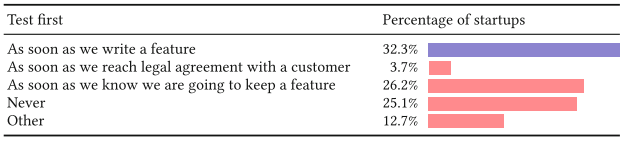

Fig. 3. Startup distribution with respect to the frequency of code testing (Color figure online)

Agile planning and frequent release are the two practices that allow software teams to be able to collect feedback on their products and adjust their development speed accordingly. 1391 startups responded with their release frequencies and 1290 indicated how far ahead they planned their product development

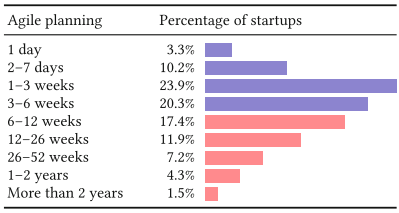

Fig. 4. Startup distribution with respect to agile planning (Color figure online)

pipelines. Regarding planning, Fig. 4 shows that most often the software startups plan ahead for 1 to 3 weeks (about 24%), more than 10% plan for 2 to 7 days, and about 3% are doing daily planning. Only less than 6% put up a yearly or longer-term plan. In total, more than 57% of the startups plan in an agile manner in terms of the time frame covered by the planning (shown by the blue bars. We used 30-day sprint to draw the division). Agile planning should be for 3 to 6 weeks (30 working days) or shorter.

As shown in Fig. 5, the most common (about 21%) release frequency used by these software startups is every 2 to 3 weeks, followed by every 1–3 months (about 19%). It is interesting to see that more than 13% of the startups are practicing continuous delivery and release product once per day or even multiple times per day. However, more than 15% other startups have really low release frequency (every 3–6 months or even more than 6 months), which is worrying given the fact that they are software startups and moving fast is not an option but a must for many of them. The bars in Fig. 5 are divided into two groups: those with release frequency of 2–3 weeks or less (blue bars) therefore indicating frequent release (again the 30-day sprint length was used as the division line), and those indicating low release frequency (taking more than one sprint to release a new version). It can be seen that more than 64% of the startups do frequently release their products.

Daily standup meeting is an agile ceremony used to facilitate communication among software development teams and organizations. Among the 1286 software startups that answered the question, more than 70% are not using the practice, in contrast to about 30% that said "yes" to the question.

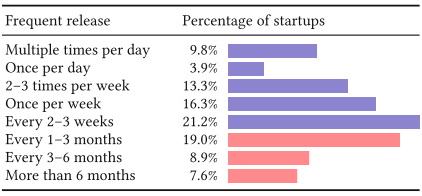

Fig. 5. Startup distribution with respect to frequency of product releases (Color figure online)

Table 2. The use of agile practices in software startups across product stages

Product development stage	Regular refactoring		Test first		Frequent release		Agile planning		Daily standup meeting	
	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No
Concept	49	47	41	41	2	0	63	36	23	76
In development	93	93	74	97	144	80	118	78	53	146
Working prototype	111	107	87	119	158	108	138	91	59	170
Functional product with limited users	246	337	190	323	483	230	350	257	198	403
Functional product with high growth	40	63	42	51	79	44	53	51	32	72
Mature product	16	35	24	19	29	31	20	32	16	35

Table 2 shows the use of the five agile practices by the software startups across different product development stages. As explained in Sect. 3.3, the use of the agile practices are simplified into "yes"/"no" Boolean options, to allow a sharper comparison. Table 2 does show that for each agile practice, the percentage of software startups using it varies across the product development stages. However, there is no discernible pattern in the variance of the percentages.

To test H_{a1}, Chi-square tests were applied. A pre-examination excluded frequent release from the test since the assumptions requested to run Chi-square test were not met. We run the tests on the cleaned sample ($n = 1526$). Since the data entries that have empty answers to each agile practice and/or product development stage were removed, each test has a different sample size (as shown in Table 3, Column 2). The test results show that regular refactoring and agile planning are linked to the development stages (the respective H_{a1} is supported). Instead, H_{a1} regarding test first and daily standup meeting cannot be supported.

4.2 Agile Practices in Lean Software Startups

Regarding the individual responses to the three Lean Startup questions from the whole sample, 489 out of the 1526 replied with a definitive "yes" to the statement

Table 3. Agile practices across product stages—Chi-square test results

Practice	n	Chi-square	Degrees of freedom	p-value	Result
Regular refactoring	1237	13.638	5	0.01808	H_{a1} supported
Test first	1108	11.06	5	0.05021	H_{a1} rejected
Agile planning	1287	12.365	5	0.030121	H_{a1} supported
Daily standup meeting	1283	7.736	5	0.1714	H_{a1} rejected

Table 4. Pivoting in lean startups across product stages

Product stage	No. of lean startups	Mean of number of pivots
Concept	3	2.0
In development	40	2.1
Working prototype	57	2.4
Functional product with limited users	107	2.0
Functional product with high growth	17	2.4
Mature product	5	2.6

"We identified the riskiest hypotheses about our business in order to test them first", and 55% claimed that "We built minimally viable products to test our hypotheses". It is interesting to explore the pivoting behavior of these startups in terms of the number of pivots they have made. 1440 out of 1526 gave valid answers to the number of pivots. The mean is 1.528 and median is 1 (sd = 2.06), in a range from 0 to 30 pivots.

229 out of the 1526 software startups are considered following the Lean Startup approach based on the selection criteria specified in Sect. 3.3. When looking closely at the pivoting in this subset, the number of total pivots the surveyed startups experienced ranges from 1 to 15, with the mean equal to 2.153 and the median to 2 (sd = 1.73). From the perspective of product development stages, we can see that, as shown in Table 4, the mean of the number of total pivots of startups at different stages ranges from 2 to 2.6. The lean startups that progressed to the stages of having functional or mature products in total have not pivoted more than those at the early product development stages.

Table 5 shows the use of the five agile practices in lean startups in comparison to that in the rest of the sample. It can be seen that there is a higher percentage of lean startups using each of the agile practices for all the five agile practices.

To test H_{a2}, we used the Chi-square test on the two groups: lean startups vs. non-lean startups. The results are shown in Table 6. The difference between

Table 5. The use of agile practices in lean startups vs. non lean startups

Agile practice	Lean startup subset		Non lean startup subset	
	Yes	No	Yes	No
Regular refactoring	99	99	456	586
Test first	86	86	372	567
Frequent release	164	61	733	433
Agile planning	119	83	626	462
Daily standup meeting	76	124	306	780

Table 6. Agile practices in lean vs. non-lean startups—Chi-square test results

Practice	n	Chi-square	Degrees of freedom	p-value	Result
Regular refactoring	1240	2.3723	1	0.1235	H_{a2} rejected
Test first	1111	6.0471	1	0.0139	H_{a2} supported
Agile planning	1290	0.0815	1	0.7752	H_{a2} rejected
Frequent release	1391	7.8438	1	0.0051	H_{a2} supported
Daily standup meeting	1286	7.3417	1	0.0067	H_{a2} supported

the two groups is not significant in terms of the use of regular refactoring and agile planning. Instead, the percentage of lean startups using test first, frequent release or daily standup meeting is significantly higher than that of non-lean startups.

5 Discussion

So are software startups using agile practices? The results of our study reveal that a majority of software startups do not use quality related agile practices, such as regular refactoring and test first. It reflects the major concern expressed in the literature that quality has a low priority and technical debt is accumulated in software startups, especially at their early stages. When the agile practices regarding the speed of development are concerned, our study shows that a large majority of software startups do move fast by adopting frequent releases and short-term agile planning. This is in line with the literature that emphasizes that speed matters significantly to software startups [7]. However, the under use of quality related agile practices in comparison to speed related practices is not unique to software startups. The same pattern has been manifested in the surveys of agile and lean adoption in software organizations in general. For example, in the 10th annual agile survey conducted by VersionOne (based on 3,880 completed responses) [11], it is shown that speed related practices (e.g., short iterations, iteration planning, release planning) are employed more often in the surveyed organizations than quality related practices (such as unit testing, refactoring, test-driven development). A smaller scale academic survey on agile and lean usage in Finnish software industry with 408 responses demonstrates the same tendency [21]. It seems that, in terms of balancing speed and quality concerns, software startups are not so different from the general population of software organizations. Agile practices related to speed are more often used by both software startups and established companies alike.

In contrast, our findings regarding daily standup meeting indicate that this well-known agile practice is not used in software startups to the same extent as in established software organizations. According to the VersionOne survey [11], daily standup meeting is the most popular agile practice among the surveyed

organizations, with an adoption rate of 83%. Its popularity is echoed in the academic survey too [21]. In our survey instead, daily standup meeting is the least frequently used practice among the five agile practices studied. Only about 30% of the software startups use this practice. One explanation of such different could be that daily standup meeting is a typical agile ceremony used by software development teams and organizations to facilitate communication. Because most startup teams have very small sizes (as described in Sect. 4), informal communication happens frequently, which renders formal communication practices less necessary. Yau and Murphy [8] offer similar arguments. They contend that, in small scale startups with only a few members, many problems that agile methods set out to solve do not exist, e.g., the communication issue.

In this study we further examined the use of agile practices by software startups at different stages of product development. The results of the hypothesis testing (H_{a1}) show that the use of agile practices including regular refactoring and agile planning does vary across the product development stages. Instead, the use of test first and daily standup meeting is not significantly associated with the stages. We cannot draw any conclusion regarding frequent release. This finding provides partial support to the claim in the literature that not all software engineering practices are usable or beneficial in different stages of startups [22]. It is an interesting direction to investigate which software engineering practices are most useful and beneficial to which stages of startups.

Another specific angle investigated in our study is the use of agile practices by software startups that adopted the Lean Startup approach. Some studies have expressed the concerns that startups adopting the Lean Startup approach have to sacrifice certain agile practices or product quality due to limited funding and short runway in order to move fast and test business hypotheses with MVPs [8,10]. However, the findings reported in Sect. 4.2 do not substantiate these concerns. On the contrary, they reveal that lean software startups tend to use agile practices more than the rest of the startups surveyed. Especially in terms of test first, frequent release and daily standup meeting, significantly higher portions of lean startups practice them. With these practices that address both needs of quality and speed, lean software startups may be in a good position to manage the "developers dilemma" [10], better at balancing between quality and speed to achieve fast product iteration.

Even though not a main focus of this study, it is worth noting the somehow surprising finding regarding the number of pivots made by lean startups across different product development stages. Pivot is considered a key component of the Lean Startup approach, an action that startups are encouraged to take based on the validated learning they obtain through testing risky business assumptions early and often [1]. Therefore, one would expect that the total number of pivots increases as startups progress along the development stages and pivot continuously. However, the result regarding pivoting reported in Sect. 4.2 does not conform to this expectation. Further investigation is needed to understand the pivoting in software startups.

Lastly, the results reported in this paper need to be viewed in the lights of the limitations of and validity threats to the study. The lack of access to the original survey design and no control to the quality of collected data pose the biggest limitation to our study, constraining the types of analysis that can be conducted and consequently the results that can be obtained. For these reasons, we went to great lengths to clean and validate the data to ensure its quality. Another limitation is due to the fact that there are a very limited number of questions in the original survey that can be associated with agile methods and practices with an acceptable level of confidence. At the end only five agile practices were brought into the study. In addition, each agile practice had only one corresponding question (item), so the risk of not obtaining valid data was increased due to the lack of multiple items to probe the same practice. These concerns pose a potential threat to the construct validity of the study. Instead, the external validity is ensured by the size and random nature of the sample. Therefore the findings of this study can be generalized to a general population of software startups.

6 Conclusion

In the past years agile methods have become main-stream software development approaches in established companies, small or large. They are considered natural choices for software startups too, since startups operate under various uncertainties and the demand on their ability to deal with change is high. Meanwhile software startups have to focus on business development as well as product development. Lean Startup is the approach that an increasing number of startups adopt to test the riskiest business assumptions in their business models. This study provided a better understanding of the state of agile practices in software startups, with a particular focus on lean startups. Based on a large survey of 1526 software startups, we found out that different agile practices are used to different extents, depending on the focus of the practices. Speed related agile practices are used to a greater extent in comparison to quality related practices. Communication practices represented by daily standup meeting is least used. In addition, unlike what is speculated in the literature, software startups who adopt the Lean Startup approach do not sacrifice quality for speed more than other startups do. Our study is the first step towards more in-depth understanding of how software startups can better use agile practices and eventually benefit from them.

In our current study we could not identify any questions specific to lean practices, such as kanban, from the original survey questions. Future work can investigate how lean practices are used in software startups. Meanwhile, "doing agile", using agile practices, does not ensure software startups of "being agile", being able to respond to change and uncertainty. This study was focused on "doing agile". Future work can assess the agility of software startups, and establish the link between "doing" and "being" agile to startup success. It would be also effort worth spent to design a new survey that is focused on investigating the adoption of agile and lean methods as well as Lean Startup in software startups.

Acknowledgement. Thanks a lot to Carmine Giardino who shared the original survey data with us.

References

1. Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business, New York (2011)
2. Blank, S.G.: The Four Steps to the Epiphany: Successful Strategies for Products that Win. Cafepress.com, Foster City (2005)
3. Sutton, S.M.: The role of process in a software start-up. IEEE Softw. **17**, 33–39 (2000)
4. Unterkalmsteiner, M., Abrahamsson, P., Wang, X., Nguyen-Duc, A., Shah, S., Bajwa, S.S., Baltes, G.H., Conboy, K., Cullina, E., Dennehy, D., et al.: Software startups-a research agenda. e-Informatica Softw. Eng. J. **10**(1), 89–123 (2016)
5. Coleman, G., O'Connor, R.V.: An investigation into software development process formation in software start-ups. J. Enterp. Inf. Manag. **21**(6), 633–648 (2008)
6. Thomas, S.: Done is better than perfect: how to beat perfectionism paralysis (2016). http://engageme.online/done-is-better-than-perfect-how-to-beat-perfectionism-paralysis/
7. Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: the greenfield startup model. IEEE Trans. Softw. Eng. **42**(6), 585–604 (2016)
8. Yau, A., Murphy, C.: Is a rigorous agile methodology the best development strategy for small scale tech startups? Technical report (CIS), Paper980, p. 9 (2013)
9. Duc, A.N., Abrahamsson, P.: Minimum viable product or multiple facet product? The role of MVP in software startups. In: Agile Processes, in Software Engineering, and Extreme Programming, vol. 251, pp. 118–130 (2016)
10. Terho, H., Suonsyrjä, S., Systä, K.: The developers dilemma: perfect product development or fast business validation? In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., Mikkonen, T. (eds.) PROFES 2016. LNCS, vol. 10027, pp. 571–579. Springer, Cham (2016). doi:10.1007/978-3-319-49094-6_42
11. VersionOne: The 10th Annual State of Agile Report. Technical report (2016)
12. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer **34**(9), 120–127 (2001)
13. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley Professional, Boston (2004)
14. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**(9), 833–859 (2008)
15. Abrahamsson, P., Conboy, K., Wang, X.: "lots done, more to do": the current state of agile systems development research. Eur. J. Inf. Syst. **18**, 281–284 (2009)
16. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B.: A decade of agile methodologies: towards explaining agile software development. J. Syst. Softw. **85**(6), 1213–1221 (2012)
17. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: a systematic mapping study. Inf. Softw. Technol. **56**(10), 1200–1218 (2014)
18. Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., Abrahamsson, P.: What do we know about software development in startups? IEEE Softw. **31**(5), 28–32 (2014)

19. Gilb, T., Gilb, K.: “Lean Startup” - the most extreme agile method by far. Agile Rec. (9), 53–54 (2012)
20. Bosch, J., Holmström Olsson, H., Björk, J., Ljungblad, J.: The early stage software startup development model: a framework for operationalizing lean principles in software startups. In: Fitzgerald, B., Conboy, K., Power, K., Valerdi, R., Morgan, L., Stol, K.-J. (eds.) LESS 2013. LNBIP, vol. 167, pp. 1–15. Springer, Heidelberg (2013). doi:10.1007/978-3-642-44930-7_1
21. Rodríguez, P., Markkula, J., Oivo, M., Turula, K.: Survey on agile and lean usage in finish software industry. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM 2012, p. 139 (2012)
22. Eloranta, V.P.: Towards a pattern language for software start-ups. In: 19th European Conference on Pattern Languages of Programs, pp. 1–11 (2014)

Adopting Test Automation on Agile Development Projects: A Grounded Theory Study of Indian Software Organizations

Sulabh Tyagi[1(✉)], Ritu Sibal[1], and Bharti Suri[2]

[1] Netaji Subhash Institute of Technology, Delhi University, New Delhi, India
sulabhtyagi2k@yahoo.co.in, ritusib@hotmail.com
[2] Guru Gobind Singh Indraprastha University, New Delhi, India
bhartisuri@gmail.com

Abstract. The role of test automation in Agile Software Development projects is of paramount importance. It is absolutely necessary to automate tests on agile projects as the number of test cases will continue to grow with each successive sprint. Through a Grounded Theory study involving 38 agile practitioners from 18 different software organizations in India, we identified five key challenges faced by agile practitioners and different strategies to overcome those challenges while practicing test automation. Understanding these challenges and strategies would help agile teams in streamlining their test automation practices.

Keywords: Test automation · Test driven development · Agile software development · Grounded theory

1 Introduction

The widespread use and popularity of agile methodologies are primarily due to their ability to produce quality software in less time with limited manpower. Most of the software industries are using scrum and XP methodologies of agile software development. Testing is an integral part of development in agile projects rather than a distinct Software Development Life Cycle (SDLC) phase [1].

Software test automation refers to the activities and efforts that intend to automate engineering tasks in a software test process using well-defined strategies and systematic solution [2]. According to [3] test automation is one of the most effective solution for projects which have strict deadlines as it speeds up the test execution and increases the test coverage.

Automation on a scrum project is not optional, for a team to sprint effectively and deliver value quickly, it needs to rely heavily on test automation [4]. Crispin and Gregory [5] argued that test automation is the key factor for successful agile software development and the core of agile testing. In a study by Puleio's [6] test automation was seen as a key factor in agile testing to keep development and testing in synchronization. It is evident from the above studies that test automation is a crucial ingredient of agile software development projects. Further, a study from Collins [7] reported that test

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 184–198, 2017.
DOI: 10.1007/978-3-319-57633-6_12

automation works very well if the agile teams find the right way to implement test automation in their projects and presented some strategies to minimize the risk during test automation implementation.

The objective of this study is to create an understanding on different challenges faced by agile practitioners while adopting test automation on agile projects and to present some possible strategies to overcome those challenges. To provide more empirical insight in this area, a grounded theory study has been conducted that involved 38 agile practitioners from 18 different software organizations in India. We hope our research will help in understanding the issues while adopting test automation on agile projects and streamlining it through proper strategies.

The rest of the paper is structured as follows: in the next section a brief overview of the Grounded Theory is presented; the third section describes the findings of this study; the fourth section discusses these findings; the fifth section presents limitations of this study and the last section concludes the paper.

2 Research Method

2.1 Grounded Theory

Grounded Theory (GT) is a systematic research method where prominence is on the generation of theory that derived from systematic and rigorous analysis of data [8, 9]. The emphasis in GT is on new theory generation which means rather than beginning with a pre-conceived theory in mind, the theory evolves during the research process itself and thus the product of continuous interplay between data collection and analysis of that data [10].

Which version of ground theory. Glaser GT states that researchers should start with the general 'area of interest' and beginning a GT study with specific research questions can lead to pre-conceived ideas or hypothesis of the research phenomena [11]. Other two versions of GT are Straussin GT [12] and Charmaz's constructivist GT [13]. This study employed the Glaserian version as our objective was to find out the issues from the real life experience of the agile practitioners related to our general area of interest i.e. Agile Project Management rather than imposing our own pre-conceived ideas and concerns that could influence this study and also due to plenty of resources available on Glaserian GT [8]. GT has been chosen as our research method for many reasons. Firstly, agile software development focuses on people and interactions, and GT, allows us to study social interactions and the behavior of people. Secondly, GT is most suited to areas of research that have not been explored in detail, and according to our knowledge, the research studies on test automation practices in agile software development is also scarce. Thirdly, GT focuses on theory generation rather than extending or verifying existing theories [14]. Finally, GT is being liberally used to study the agile teams [11, 15–18]. Following Glaser's guidelines, the study started with a general area of interest – Agile project management – rather than beginning with a specific research problem. Problems and its key concerns will emerge in the initial stages of data analysis and it did [19].

2.2 Data Collection

Data collection in GT is guided through theoretical sampling whereby researchers iteratively collect and analyze their data to decide what data to collect next and where to find the data [20]. A GT study requires the theoretical sampling to be continued until theoretical saturation is reached that is when no more new concepts or categories emerge from the data, and further data collection would be a waste of time [21].

Recruiting Participants. This study involved 38 agile practitioners from 18 different software organizations in India with size varied from 50 to 200,000 employees located in Bengaluru, Mumbai, Pune, Noida and Gurgaon. The project duration varied from 6 to 36 months and team size varied from 7 to 20 people on different projects with wide range of domains like software consultancy, e-commerce platforms. Due to ethical considerations and to keep our participants identity confidential, we used codes P1 to P38 to identify our participants. Table 1 shows the participants and project details of this research study. We contacted members of Agile Software Community of India [22] and also took part in Agile India 2016 International Conference [23] on Agile that provided us the platform to collaborate with many agile practitioners across India and abroad. Many practitioners agreed to be a part of our research and participated in this study.

Interviews. Face-to-face semi-structured interviews were conducted with agile practitioners using open-ended questions over a period of eighteen months. Normally, each interview lasted for about one hour and was scheduled at the mutually agreed location. The interviews were audio recorded with the consent from the participants on ensuring full confidentiality, so that we could concentrate on the conversation. Ten participants were interviewed from four different software organizations in first phase of our study. Interview began with warm up questions regarding participants experience, their roles, nature of duties and different agile project management practices in their respective projects. Each participant had a 3–4 or more years of hands-on experience on either scrum, XP or both. Initial sample of participants comprised Scrum Masters, Developers, Product Owners (PO's) and Testers. Then we progressed to our second phase of interviews and expanded our sample participants to Senior Management people (Chief Technical Officer, Vice-President), Agile coaches and Devops to gain the well rounded perspective from participants, also the set of questions were gradually modified as per Glaser [20] to achieve theoretical saturation of our core category - Adopting Test Automation. After completion of each interview, it was transcribed and analyzed line by line to identify key points, codes, concepts and categories. Data collection and its analysis were performed iteratively. Constant comparison of interview transcripts helped us in guiding future interviews, and then we continuously fed back the analysis of interviews and observations from our study into the emerging results. All the data was personally collected and analyzed by the primary author so that consistency can be maintained in the application of GT.

Observations. We also performed passive observations in two projects denoted as Sigma and Delta in two different Indian software organizations denoted as X and Y. X

Table 1. Summary of participants and project details. (Agile Position: Agile Coach (AC), Chief Technical Officer (CTO), Developer (DEV), Devops (DO), Product Owner (PO), Scrum Master (SM), Senior Agile Coach (SAC), Senior Developer (SD), Senior Quality Analyst (SQA), Senior Tester (ST), Test Analyst (TA), Tester (TES), Vice-President (VP)

Participant (Code)	Agile Position/ Experience (yrs)	Project distribution location	Agile method	Domain	Team size	Project duration (Mos)	Sprint duration (Wks)
P1, P2	TES/3, SM/10	India-UK	Scrum	Finance	10–12, 16–18	12, 24	2
P3, P4	ST/4, PO/5	India-USA	Scrum & XP	Network Mgmt. Services	10	10 to 12	2–3
P5, P10	SM/6, ST/5	India-South East Asia	Scrum & XP	Insurance	12–14, 12	8–10, 15–16	3–4
P6	TES/4	India-Europe	Scrum & XP	Mobile Retail	18	18–20	3–4
P7, P8	TES/3, SD/5	India-USA	Scrum & XP	E-Commerce	14	12–14	1–2
P9	SD/4	India-Australia	Scrum & XP	Banking & Finance	20	24	3–4
P11, P12	AC/12, CTO/16	India-USA - Australia	Scrum & XP	Software Consultancy & Services	14–15, 18–20	12–14, 15–16	2–4
P13, P14	AC X 2/8, 10	India-New Zealand	Scrum & XP	IT & Agile Training	7–8	36	2–3
P15	DEV/3	India-UK	Scrum & XP	Telecom	12–13	42	3–4
P16, P17	TA/5, VP/12	India-UK	Scrum & XP	Insurance	9–10	12	2–3
P18, P19	SM/7, AC/8	India-Western Europe	Scrum	Health Care	18–19	24	3–4
P20, P21	TES/3.5, DO/4	India-USA	Scrum & XP	Energy Metering Solutions	10–12	36	3–4
P22, P23	TES/4, DEV/4.5	India-Canada	Scrum & XP	Finance	9–10, 12	24, 18–20	3–4
P24, P25	ST/5.5, TA/5	India-Australia	Scrum & XP	E-Commerce	10, 9–10	12, 12–14	1–2, 2–3
P26	SQA/4	India-South East Asia-Australia	Scrum	Information Security	8–9	18	3–4
P27	TES/3.5	India-Western Europe	Scrum	Web Portal	12–13	10–12	1–2
P28	SM/8	India-Western Europe	Scrum & XP	IT & Agile Training	10–12	12–14	2–3
P29, P30	SM/10, VP/12	India-USA	Scrum & XP	IT Infrastructure	12–14	16–18	2–3
P31	SAC/12	India-Europe	Scrum & XP	IT & Agile Training	8–9	24	3
P32	SM/6	India-Europe	Scrum & XP	Agile Training	10–11	12	3–4
P33	PO/3	India-Europe	Scrum & XP	Finance	12–13	15–16	2–3
P34	ST/4.5	India-UAE	Scrum	Banking & Finance	15–18	24	2–3
P35	TES/4	India-USA	Scrum	Telecom	10–11	10–12	3–4
P36, P37	SM/7, DEV/4	India-USA	Scrum & XP	E-Commerce	12–14, 18–19	6–8, 18	2–3 1–2
P38	PO/4.5	India-UK	Scrum & XP	Telecom	20	12–14	2–3

is into smart metering and energy management solutions with presence in over 30 countries and Y is into e-commerce business with presence in over 4 countries.

Observation period in Sigma and Delta was 8 and 6 months respectively. Sigma was practicing agile mainly blend of scrum and XP from past 3 years but Delta was relatively new to agile and practicing scrum from past 1 year. We observed daily stand ups, sprint

retrospectives, sprint review meetings, end sprint demos, pair programming practices, daily smoke and regression tests and we had taken field notes along the way about our observations and transcribed them for analysis. Moreover, we compared the codes emerged during observations with the codes from the interviews that helped us in achieving triangulation. The interview data was further strengthened by our observations from these two projects.

2.3 Data Analysis

Coding. Following Glaser's two successive stages of substantive coding: open and selective coding, we began our data analysis with open coding. It helps us in directing our research by identifying a core category and serves as the initial step of the theoretical analysis in GT [14]. Then, selective coding was performed to identify the categories that were related to the core and to ascertain theoretical saturation.

Constant Comparative Method. Here, codes are compared with other codes to produce concepts, codes are compared further with concepts to produce new concepts and finally concepts are compared with other concepts to produce categories [14].

Memoing. Memos are written notes to log reflections between data, codes and their relationships as they occur in researchers mind [20]. In our case, we wrote memos as soon as we had some ideas about emerging codes and their relationships.

Phase 1: Identifying the core category. We commenced phase 1 of our interviews on our general area of interest "Agile Project Management" and performed open coding on data that generated initial codes, which guided us on further data collection as per theoretical sampling process of classic GT [20]. We continued collecting and analyzing our data iteratively that gradually led us to our core category i.e. "Adopting Test Automation" on agile projects.

Open Coding. In open coding interview transcripts are being analyzed in detail and key points are identified from each interview transcript [24]. In the next step, key points are collated and particular code is assigned to each key point [25]. Code is a phrase used to summarize the key point in 2 to 3 words. Using the constant comparative method, the codes from each interview were compared constantly with the codes from the same as well as from other interviews and also with data based on our observations and written memos. The constant comparison and grouping of similar codes lead to the second level of abstraction, called concepts. Further, this method is repeated on concepts to produce the third level of abstraction, called categories.

Open coding was ended on identifying our core category *"Adopting test automation"*. Two potential near core categories were also emerged like "Quality work delivery" and "Manage changing requirements", but we selected "Adopting test automation" as our category as it is related to most other categories in a meaningful way. An example of open coding process is shown in Table 2 that depicts the emergence of our core category from the combined analysis of interviews and observations.

Table 2. Example of Open Coding Process

Open coding	Interview Quotation – P5, Scrum Master	Observation (Org.: Y, Project: Delta)
Statement/Field note	"Most important question…whether or not your project is truly time driven, whether or not you are delivering high quality product, time is speed for us and we can achieve that [speed and quality] by embracing automation."	Acceptance testing was practiced manually till sprint 3, consuming lot of time and effort. UI changes were frequent due to constant new product launches, decision to automate acceptance tests, acceptance tests automation started
Key point	Need for timely delivery of quality products, Achieving speed, Quality through automation	Manual acceptance consumes time and effort, Frequent UI changes, Automating acceptance tests
Code	Timely delivery, Quality products, Embracing automation	Time and effort loss, Constant UI changes, Acceptance tests automation
Concept	Achieving quality and speed by embracing automation	Achieving speed by embracing automation
Category	Adopting test automation	

Phase 2: Refining the core category. As per theoretical sampling process, selecting new interviewees and sites for data collection should come from the results of the coding process [14]. We progressed into phase 2 and continued our data collection process.

Table 3. Example of selective coding process

Selective coding	Interview Quotation – P6, Tester	Observation (Org.: X, Project: Sigma)
Statement/Field note	"Our project…lot of business logic, we handle lot of features additions & changes…has accumulation effect on our tests too…which makes them grow in numbers with every sprint and it is really difficult to maintain [test scripts]"	Frequent change requests received from the customers, constant addition, modification of page elements, effect on test scripts size, making test script maintenance difficult for the team
Key point	Adding new features, Test scripts continue to grow, Difficulty in test script maintenance	Frequent change requests, Constant changes in test scripts, Difficulty in test script maintenance
Code	Grow in test scripts, Difficulty in maintaining test scripts	Constant test script changes, Difficulty in maintaining test scripts
Concept	Difficulty in test script maintenance	Difficulty in test script maintenance
Category	Test script maintenance	

Selective Coding. Here, only those interview transcripts were coded that were related to our core category i.e. "Adopting Test Automation". Constant comparative method