An Investigation of Agility Issues in Scrum Teams Using Agility Indicators

Minna Pikkarainen[1] & Xiaofeng Wang[2]

[1] VTT, Technical Research Centre of Finland. minna.pikkarainen@vtt.fi

[2] Lero, the Irish Software Engineering Research Centre. xiaofeng.wang@ul.ie

Abstract: Agile software development methods have emerged and become increasingly popular in recent years, yet the issues encountered by software development teams that strive to achieve agility using agile methods are yet to be explored systematically. Built upon a previous study that has established a set of indicators of agility, this study investigates what issues are manifested in software development teams using agile methods. It is focused on Scrum teams particularly. In other words, the goal of the paper is to evaluate Scrum teams using agility indicators and therefore to further validate previously presented agility indicators within the additional cases. A multiple case study research method is employed. The findings of the study reveals that the teams using Scrum do not necessarily achieve agility in terms of team autonomy, sharing, stability and embraced uncertainty. The possible reasons include previous organizational plan-driven culture, resistance towards the Scrum roles and changing resources.

Keywords: Agile software development, Scrum, agility indicator, autonomous team, context sharing, stability, uncertainty

1. Introduction

Agility is a multifaceted concept and has been interpreted in many different ways both in system development research and practice [5]. It originates from several disciplines including manufacturing, business and management, and has root in several inter-related concepts, such as flexibility and leanness. Based on the comparison and contrast of these concepts, [6] provide a broad definition of agility as "the continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high quality, simplistic, economical components and relationships with its environment" [6, p.40]. In Information Systems Development (ISD) context, ISD agility is concerned with why and how ISD organizations sense and respond swiftly as they develop and maintain information system applications [12].

In software development domain specifically, although agile software development methods, such as eXtreme Programming (XP) [2] and Scrum [20], have

emerged and become increasingly popular in the last decade, the meaning of agility is yet to be fully understood in this domain. [22] identify a set of agility indicators through investigating software development teams using agile methods, but the conclusion they have drawn is based on the study of XP method only. At he same time as the teams using Scrum are not yet evaluated from agility perspective, the generalisability of these indicators to other agile methods has yet to be validated.

Based on this observation, this study sets out to investigate the meaning of agility in software development teams using Scrum development method, utilizing the indicators developed by [22]. Scrum has been pioneered by [20] and is one of the most popular agile methods adopted in many companies. It was originally influenced by Boehm's 'spiral' model, but it was developed based on industrial experiences to simplify the complexity of the project and requirements management in software organizations [21]. Scrum describes practices on an iterative, incremental time boxed process skeleton. At the beginning of the iteration, the team has a sprint planning meeting in which they decide what the team will do during the following iteration. At the end of the iteration, the team presents the results to all the stakeholders in the sprint review meetings to gain feedback on their work. The heart of Scrum is an iteration in which the self-organizing team builds software based on the goals and plans defined in the sprint planning meeting. The team also has a daily 15 minute meeting called the daily Scrum, in which they check the status of the project and plan the activities of the next day [21].

The remaining part of the paper is organized as follows. Section 2 briefly introduces the previous work that leads to this study; Section 3 describes the research method and the context of the empirical study; the findings are presented in Section 4; Section 5 is a discussion of the findings in the light of the relevant studies. A summary section wraps up the paper with the implications and limitations of the study as well as the future work.

2. Agility Indicators for Software Development Teams

[22] identify a set of agility indicators for software development teams. Two facets - autonomous but sharing team and stability with embraced uncertainty - are of particular relevance to this study (see Table 1).

Table 1. Agility indicators (adapted from [22])

Agility facets	Manifested in software development
Autonomous but sharing team	Distributed competences
	Disciplined team
	Knowledge sharing
	Context sharing
	Collective ownership of results
Stability with embraced uncertainty	Short-term certainty
	Team being satisfied, motivated and focused
	Working at a sustainable pace
	Probability to change directions
	Having a whole picture of the project

2.1 Autonomous but sharing team

Agile advocates suggest that software development processes should be organized to improve and distribute both technical and social competences continuously [4]. [1] discover a competence build-up in a team where several agile practices are piloted. [22] suggest that a team composed of autonomous but interconnected developers has tendency to be agile. In agile teams competences are not concentrated on few people so that there is no bottleneck in the development process. Team members are confident and courageous in the interactions with customers and with each other. Meanwhile, contrary to the view that agility means chaos [17], [3] argue the importance of discipline in agile processes. An agile team is composed of disciplined, self-responsible and committed individuals. Discipline is an essential component of an autonomous team [22].

Sharing is a common characteristic of agile teams, including both knowledge sharing and context sharing [11], [14], [16], [19]. [14] believe that the so-called "background knowledge" about a project is important to achieve effective communication. It is important for all team members to have a common frame of reference - a common basis of understanding. [16] observe that, in the company they have studied, there is a measurable increase in the visibility of what everyone is doing on the team after the adoption of the agile practices. The improvement in visibility is considered one of the greatest successes the company has achieved. [22] argue that, to effectively self-manage, a team needs to share the understanding of their working context in addition to knowledge sharing. Context sharing is a precondition to provide effective feedback, interpret them in a sensible way, and

take appropriate actions. Sharing also means results sharing, such as collective ownership of code and solutions, which reduces the risk of knowledge loss and increases the sense of being a true team. [11] reports the experience of collective ownership of codes. When it is realized, even the most complex business problems can be easily figured out. In contrast, individual ownership of code makes people defensive - people take it personally when someone suggests their code does not work. [19] also document the collective ownership in their experience report where developers took ownership of the features they created and took pride in showing their work to the stakeholders during sprint reviews. [18] notice that in a team they have studied, at every meeting, as small tasks were completed and the team could see progress toward the goal, everyone rejoiced.

2.2 Stability with embraced uncertainty

[22] emphasize that stability is a desired property of development teams that gives developers a sense of security and control over what they are working on. It can be drawn from a short-term certainty provided by a time-boxed development process. Stability for development also means a team is working at a sustainable pace, focused and motivated, working with ease and satisfaction. Several studies have noticed team satisfaction and motivation in agile teams [e.g. [18], [19], [9]. For example, [9] conduct a survey of a team using XP and find that it creates a surge in morale since XP provides constant feedback to the developers and at the end of each day the team has a working product. Team members gain a sense of accomplishment from their daily work, because they could immediately see the positive impact their efforts have on the project. When morale is high, people are excited about their work, leading to a more effective, efficient development team.

Meanwhile, uncertainty is inevitable in software development. It comes from both the environment a team is embedded in and the development process itself. Managing uncertainty does not mean to predict what is going to happen and do future proof work today. It is to ensure the probability to change the direction a team goes towards but meantime not to get short-sighted, to have a whole picture of the project in mind, and to let solutions emerge [22]. It is echoed in [10] who suggest that, when using the XP practices, especially the simple design, one should look ahead and do things incrementally, in order to have a big picture.

3. Research Approach

This study employs a qualitative research approach, treating agility as a qualitative property of a software development team that can be better studied through words and the meanings people ascribe to them. The specific research method used is case study. A multiple-case study design is employed, since the study intends to

be a cross-sectional study. The research results would be more convincing when similar findings emerge in different cases and evidence is built up through a family of cases [15]. A software development team is taken as a case, and the level of inquiry is at the team level. Semi-structured interviews are the main data collection method. Interviews are transcribed verbatim, imported to NVivo and analysed using the agility indicators as analytical lens.

Two companies that were selected for this research were both market leaders of specific products working in dynamic, global market environments. Both companies originally deployed Scrum method because they had a clear need to response to the needs of the changing market situation and to produce products to the markets faster. Both of the companies were SMEs that had the key development group in one European country but the market offices in all over the world.

Case company 1 produces commercial products in information safety domain. The use of Scrum method is integrated to their company level process model. The team involved in this study, Team A, has 4 developers, one quality engineer and a Scrum master. Case company 2 provides hardware driven embedded commercial products. Scrum has been used in some specific project teams for four years. Team B, the team this study investigates is a Scrum team producing a platform product which is strategically important for the company. It consists of 5 developers and a project manager who is also doing the actual development work.

4. Case Analysis

Using the agility indicators as analytical lenses, the analysis of the two Scrum teams revealed a set of issues faced by the teams when they strive to achieve agility using Scrum.

Team A

Team A had a difficult start when time boxing did not work for the first three iterations: *"We didn't do very well with the time boxing in the first two or three springs." (Developer).* Developers felt that, due to the use of Scrum method, they needed more testing and integration skills than before: *"For me testing was difficult, I do not know it so well, that is a reason. It also was a communication problem." (Developer).* Furthermore, it was challenging for developers to take a full responsibility of the project decisions: *"I was not so sure, if I dare to take this decision to myself or not." (Developer).* As soon as the Scrum framework was fully taken into actual use, however, the team made the 'time boxing' working at all levels: *"Basically the setup was so that we had two days [of meetings]. The first day we had review meeting, then the next day we had the sprint planning meeting... Every month two days, only for these activities." (Manager).* As a consequence, two agile practices, unit testing and continuous integration, started to be used effectively by the team through discussion in iteration retrospective meetings. Meanwhile, both knowledge sharing and context sharing are improved

among the team members: *"Information was totally shared within the team." (Manager).* Over the time, the team moved one step closer to their vision and the developers started to take more responsibility of the decisions related to product design: *"In the end of the project we did all the decision as a team… We made all decisions that are related to the design." (Developer).* However, knowledge sharing between developers and stakeholders did not seem working well: *"Everything we got was like second hand information." (Developer).*

Team A was good at keeping their short term certainty at the beginning. The product management and the team made requirements prioritization and analysis during the sprint planning meetings, and all the stakeholders were happy about the results presented in the sprint reviews. However, the situation changed radically when the project started to have increasing amount of features in the product backlog and the Scrum master and product owner lost the control over the backlog management: *"As I said I run it, the backlog wasn't properly organized, it wasn't prioritized." (Manager).* The time boxed meetings were not enough to assure neither the long or short term certainty: *"So most of the time we didn't know what the feature was, we did not spend a lot of time analyzing the feature."* (Developer), and the project lost the short term goal: *"we don't really have a clear goal." (Manager).* Instead of embracing changes and uncertainty in the project, the purpose of the developers was to reduce the amount of changes: *"We tried to reduce the risks on the interface side by trying to reduce the amount of changes around the project." (Developer).* Furthermore, the developers refused to comment on the technical solutions of the product in Sprint planning meetings because they did not have possibilities to communicate with actual customers: *"We were never able to talk to the people who might have an impression of how it should behave." (Developer).* As a result, the developers were not satisfied with the way they were working. Some developers particularly did not like Scrum meetings: *"Some of the developers were not very happy at all with the sprint." (Manager).*

Team B

Team B was deploying Scum in a quite late phase of the overall development cycle. Due to this late deployment, the developers and the project manager found them suddenly start to participate in sprint planning meetings. Meantime however, project manager and the senior management did not want to change their way of communicating. They still had their own weekly status meetings: "*Originally those meetings were held to solve conflicts, now they are used in status monitoring." (Manager).* Context knowledge about the project was not shared with the whole team due to the separation of meetings. However, although the developers were located in separated rooms, all specification and technology knowledge were well shared between the team members. The strong communication was achieved using workshop techniques: *"We talk about all techniques and specifications with the whole product group, the purpose is to get everyone's opinion." (Developer).* However, decision making was not distributed in Team B. In fact, the project

manager was still the sole responsible in the decision making: *"Project manager alone is responsible of all decision making." (Manager).*

The developers were happy during the first two increments as long as the manager gave them peaceful time to work towards the increment goals. However, the situation changed during the third increment because of new emergent customer demands. The consequence of the demands was that the team had to suddenly work on a different project: *"After second increment, we lost this possibility when other work tasks appeared; the third sprint was terrible." (Manager).* At the implementation stage, also the management faced a new challenge. They had difficulties in managing resourcing activities. Thus, the developers had to work in several projects at the same time, changing between projects during the working sprints: *"Those resources that were booked for the project have been stolen later on." (Manager).* As a consequence the team lost the short term certainty: "*It is not possible to say what you are doing next Thursday, because you never know what emails you have got during the night before!*" *(Developer).* Somehow, the project team managed to make releases after every two months, but the goals of the increment could not often be achieved due to the resourcing problems. Meanwhile, the developers found that it was not easy to change project directions due to a lack of tool support: "*We do not really have requirements management tool, the tool that we use is more like bug fixing data*" *(Developer).* Change management is complex in the case of Team B also due to the fact that the team uses software product line techniques. For example, feature impact analysis (where features are reported in a product backlog including possible release information and estimations. In the Sprint planning, features are divided into smaller working tasks and pointed out for each available resource) turned to be a challenge for both developers and project management in software product line environment. The use of Scrum did not help the team to maintain reusability and to implement commercial off-the-shelf products using technical standards. The issues discovered in the two Scrum teams are summarized in Table 2.

Table 2. Agility issues in the two Scrum teams

Agility facets	**Manifested in software development**	**Agility Issues in the Scrum teams**	
		Issue	**Reason**
Autonomous but sharing team	Distributed competences	Communication was difficult about testing competencies	Testers had resistance towards the change
		Lack of continuous integration competence	No previous experience, difficulties to understand the real meaning
	Disciplined team	Resourcing was not working, planned features	Not enough people, customer projects always in the first priority
	Knowledge sharing	Customer information was second hand for developers	Authority of product owner and scrum master in customer communication
	Context sharing	Difficulties to understand the stakeholders	Previous organization culture
	Collective ownership of results	Difficulties for a developers to take responsibility	Previous organization culture
Stability with embraced uncertainty	Short-term certainty	There was not clear short term goal in the project	there was not enough time to analyse features
		Changes were not enough deeply enough analysed	Agile way of working not support change request analysis
	Team being satisfied, motivated and focused	Some of the developers did not like meetings	Meeting time consuming; lack of preparation, not enough time for feature analysis
		Changing resource situation decreased developers' motivation	No peaceful time to work planned working tasks
	Working at a sustainable pace	No knowledge of what will happen in next week	Developers were resourced in several projects at the same time, some projects in maintenance mode
	Probability to change directions	Changes in the plans time consuming; avoiding of changes	Product backlog was poorly managed
		Tools do not support management of evolving requirements	Company policy
		Change management was complex	Use of software product line architecture
		No time to analyse features	Poorly organized product backlog, no requirements prioritization before iteration planning meetings

5. Discussion

The analysis reveals that there are several significant agility issues in both Scrum teams. It seems that neither of the two Scrum teams is agile in terms of team autonomy or stability with embraced uncertainty. Case 1 is more agile in terms of short term certainty and team autonomy but fails to maintain the whole picture of the project and knowledge sharing with customers. Case 2 is able to hold the overall picture quite well, but has no ability to achieve short term certainty and knowledge sharing between the team members. In both cases, developers are unsatisfied and unmotivated. The reasons behind these issues appear to be: 1) Previous organizational plan-driven culture; 2) Resistance towards the Scrum roles; 3) Use of technical standards; 4) Reusability goals; 5) Lack of tool support; 6) Lack of needed agile competence/skills; and 7) Evolving resources.

Notwithstanding the implications of current agile development models, methods and frameworks as well as the increasing interest of industry as described in previous research [6, 8, 13, 18], there are still concerns that the use of agile method itself do not answer to the company goals to be flexible, and rapidly produce working software. Analysis provided in this paper supports an assumption that the Scrum teams do not necessary fulfil the goals of agility in terms of continual readiness and proactively or reactively embrace to change as described in [6]. Although there are several experience reports that describe success stories of the use of Scrum method in information system development team, most often they do not reveal critical issues that the real development teams are dealing with.

[18] report on use of the Scrum method in three small software development teams. Similarly to our cases, these teams have: 1) difficulties with developers' attitude towards every day meetings; 2) lack of competence on estimation and continuous activity planning; and 3) complexity of product backlog management. One consequence of the situation is that the Scrum team overcommitted themselves taking too many responsibilities and managers had to modify the product backlog to reflect to the new strategy and company process model which was against the Scrum development.

[7] reports Solystics experience of the use of Scrum method focusing especially on the issues revealed among the developers and managers in Scrum development. According to the experiences of large and complex system development, it is shown that from project level it takes often much time from people to really understand the meaning of Scrum meetings. Additionally, use of scrum demands new communication skills in the situation in which the individual contribution is easily hidden. Furthermore estimation process needs to be well shared with developers. From organizational level long term visibility is difficult to manage in Scrum project.

[13] report on the Scrum impacts on customer satisfaction and overtime work in the teams. Based on the empirical analysis of the case study in which Scrum was used in a development project, they reveal that although it is sometimes difficult to follow sprints of 30 days, hold daily Scrum meetings as a Scrum practice, facilitating customers to keep up to date with the development work and planning meetings helps to reduce confusion about what should be developed from the customer perspective. Similar to Mann and Maurer's case, following time-boxed 30-day sprints and holding daily meetings and time-boxed sprint planning meetings turn out to be difficult also for the two cases we have studied.

[19] examine the use of Scrum in Primavera. Based on their experiences they reveal that in a Scrum team developers have sometimes difficulties to manage increasingly growing bug lists which can cause a situation tin which the team produces high amount of features that, however, are not in good enough condition to show to stakeholders in Sprint review meetings. Furthermore, developers, in this case, put too much emphasize on stakeholders comments taking them always into the development sprint. In the analysed cases, from management perspective, Scrum made it difficult to predict releases and to assure product maintainability in long term. For example:

- People were worried about the role and responsibility change
- Developers took stakeholders comments in too easily, although it would not always been necessary
- Developers showed features that were not fully tested in Sprint review
- Backlog of bugs was growing, many features were not in good enough condition
- Losing sight of technical infrastructure and long term maintainability
- Scrum made it difficult to determine how far you are from release because of the requirements change

[8] report results of action research made for Scrum development team in Avinor. As an issue they have identified problems with estimation and backlog management. The problems with effort estimation, lack of model for the action between the different parties and the lack of time to complete the backlog were the same defined also in our case study.

6. Conclusion

The use of agile methods has increased dramatically during the past years. However, the meaning of agility is not fully understood either part of the research communities or ISD enterprises. In this paper, two Scrum teams were analysed using previously identified agility indicators 1) autonomous but sharing team and 2) stability with embraced uncertainly. In the future, we continue analysis of the presented learning aspects of agility in the Scrum teams. Because the use of two cases, are not enough to do generalisation of the results, we also intend to continue

analysis with the additional agile cases. This is done in order to further validate the presented agility indicators.

Acknowledgements

This work was supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre (www.lero.ie) and TEKES to VTT, Technical Research Centre of Finland.

References

1. Auvinen, J., R. Back, J. Heidenberg, P. Hirkman and L. Milovanov (2006). Software process improvement with agile practices in a large telecom company. In Proceedings of Product-Focused Software Process Improvement, Springer-Verlag, Berlin, LNCS 4034, 79-93.
2. Beck, K. (1999). *Extreme Programming Explained*. Addison Wesley, Reading, MA.
3. Beck, K. and B. Boehm (2003). Agility through discipline: a debate. *Computer*, 36(6), 44-46.
4. Cockburn, A. and J. Highsmith (2001). Agile Software Development: The People Factor. *Computer*, 34(11), 131-133.
5. Conboy, K. (2009) Agility From First Principles: Reconstructing the Concept of Agility in Information Systems Development, *Information Systems Research* (forthcoming)
6. Conboy, K. and B. Fitzgerald (2004). Toward a Conceptual Framework of Agile Methods. In *Proceedings of Extreme Programming and Agile Methods - XP/ Agile Universe 2004*, Springer-Verlag, Berlin.
7. Derbier, G. (2003) Agile Development in the old economy, *Agile development conference*, IEEE Computer Society.
8. Dingsoyr, T. Hanssen, K.G. Dybå, T. (2006) Developing Software with Scrum in a Small Cross-Organizational project, *EuroSPI Conference*, pp.5-15.
9. Drobka, J., D. Noftz and R. Raghu (2004). Piloting XP on Four Mission-Critical Projects. *IEEE Software*, 21(6), 70-75.
10. Elssamadisy, A. and G. Schalliol (2002). Recognizing and Responding to "Bad Smells" in Extreme Programming. In *Proceedings of the 24th International Conference On Software Engineering*. Association Computing Machinery, New York, 617-622.
11. Fredrick, C. (2003). Extreme Programming: Growing a Team Horizontally, *Extreme Programming and Agile Methods - XP/Agile Universe 2003*. Springer-Verlag, Berlin. LNCS 2753, 9-17.
12. Lyytinen, K. and G. M. Rose (2006). Information System Development Agility as Organizational Learning. *European Journal of Information Systems*, 15(2), 183-199.
13. Mann, C. & Maurer, F. (2005) A case study on the Impact of Scrum on Overtime and Customer Satisfaction. *Agile 2005 Conference*, Denver.
14. Melnik, G. and F. Maurer (2004). Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing. In Proceedings of the Agile Development Conference. *IEEE Computer Soc*, Los Alamitos, 21-31.
15. Miles, M. B. and A. M. Huberman (1994). *Qualitative Data Analysis: an Expanded Sourcebook*. Sage, Thousand Oaks, California.
16. Poole, C. and J. Huisman (2001). Using Extreme Programming in a Maintenance Environment. *IEEE Software*, 18(6), 42-50.

17. Rakitin, S. (2001). Manifesto Elicits Cynicism. *IEEE Computer*, 34(12), p. 4.
18. Rising, L. & Janoff, N. S. (2000) The Scrum software development process for small teams. *IEEE Software*, Vol. 17, No. 4, pp. 26–32.
19. Schatz, B. & Abdelshafi, I. (2005) Primavera Gets Agile: A Successfull Transition to Agile Development. *IEEE Software*, Vol. 22, No. 3, pp. 36–42.
20. Schwaber, K. and A. Beedle (2002) Agile Software Development with SCRUM. Prentice-Hall, Upper Saddle River, NJ.
21. Schwaber, K. (2003) *Agile Project Management with Scrum*. Washington: Microsoft Press.
22. Wang, X. and K. Conboy (2009) "Understanding Agility In Software Development Through A Complex Adaptive Systems Perspective", *ECIS 2009.*