Agile teams: Do’s and don’ts in agile software development

Öjvind Lindgren
University of Borås
Boras, Sweden
ojvind.lindgren@hb.se

Jennifer McAllister
Middlesex University
London, United Kingdom
j.mcallister@mdx.ac.uk

Abstract—Self-organizing teams are a distinctive feature of agile software development, and they directly affect the effectiveness of the team and the overall project success. Agile software development methods, and Scrum method in particular, emphasize self-organizing teams, but they do not provide clear guidelines on how teams should become and remain self-organizing. The theory of agile teams explains how teams devoted to software development assume informal, implicit, temporary, and spontaneous roles to carry out balanced practices while they face critical factors in order to become self-organizing.

Index Terms—Self-organizing team, agile team, agile management

INTRODUCTION

Traditional software development teams include individuals with different organizational roles such as developers, testers, designers, business analysts, etc. Project managers are responsible for managing the team as a whole, such as setting goals, assigning tasks, monitoring the progress assessments and process improvement. Project managers act as an intermediate layer between the team and the senior management, transmitting to the team the expectations of the top management and raising any issues on behalf of the entire team to senior management for resolution. Project managers in traditional teams are also responsible for managing client relationships and their expectations through coordination between the team and the customers.

Agile software development teams are self-organizing and include individuals who manage their own workload, they can transfer their work if necessary and the decision-making is a responsibility of the team as a whole. The self-organizing teams have autonomy and must have a common focus, mutual trust, respect and ability to re-organize to meet new challenges.

The remainder of this paper is organized as follows: Section 2 gives an overview on teamwork and agile software development. Section 3 examines self-organizing teams and how the affect team performance. Finally section 4 concludes.

AGILE METHODS

The methods of agile software development emerged in the late nineties. Agile methods are follow an iterative and incremental development carried out by self-organizing collaborative dynamically teams adjusted to the changing requirements of the customer. The developers of some of these methods wrote the Agile Manifesto [1] and they coined the term to include several iterative and incremental agile methods. The values of the Agile Manifesto are:

- “Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan”.

Agile methods were developed in response to the perceived weaknesses in traditional software development models. The principles behind the Agile Manifesto are fast and continuous delivery of software under development, response to changing requirements, effective communication encouragement and support self-organizing teams.

Scrum

Scrum was developed by Jeff Sutherland and formalized by Ken Schwaber. It is characterized by working cycles composed by sprints typically in the range of two to four weeks. During each sprint, the self-organizing teams collect tasks from a prioritized customer requirements list, so that the features they develop are the ones with the highest value for the customer. At the end of each sprint a potentially shippable product is delivered to the customer.

The product owner is a representative of the client and she is ultimately responsible for the product creation, she establishes a business plan and a roadmap that includes the general specification of how to carry out several product launches. The product owner is responsible for product features, release dates, and she should give priority to the characteristics of the product according to the market value, she adjusts features and priorities every 30 days, as needed, and accepts or rejects work results of team.

The scrum master is a facilitator who works closely with the team and the product owner. The scrum master should be aware of the tasks that have been completed,, the new tasks that have been identified as well as any change in time estimates. She is responsible for the account and the removal of any obstacles the team finds. She also helps with the resolution
of any discrepancies or problems between team members to ensure full productivity. Her tasks include ensuring the team is totally functional and operational, allowing close cooperation in all roles and functions, removing obstacles, protecting the team from external interference.

The team has cross functionality and typically consists of five to nine members. The team selects the sprint goals and the results of the specified job. The team has the right to do anything within the limits of the project to achieve the goal of the sprint. The team organizes and displays the results of the work to the product owner.

**Extreme programming**

Extreme programming was developed by Kent Beck, supported by Ward Cunningham, Ron Jeffries and Martin Fowler. It is defined as a methodology for small and medium develop software teams with vague requirements or requirements expected to have high changes during the project execution. It was developed to address and resolve some of the classic problems of developing software such as extension of deadlines or inability to solve business problems. It advocates short release cycles, intended to limit the scope of the unforeseen extension of deadlines. This methodology asks the client to select the smallest release that has the maximum business value.

It aims to reduce the amount of issues that can face problems in the software production, thus reducing the risk that the project is canceled. This requires the customer to be part of the team and provide rapid feedback.

The main roles in extreme programming are the following ones. They intend to divide the work and to establish a classification of the responsibilities. Nevertheless several roles can be performed by the same member of the team:

- The coach is responsible for the whole process. She has to remain calm in stressful situations and lead the team. The coach must also understand the process and learn from other extreme programming teams.
- The tracker is responsible for making good estimates and seeing how they match with the actual results. With feedback and practice, the tracker must be able to know the status of iterations and releases.
- The programmer must have good communication skills and keep it simple at work and code.
- The client intends to be an integral part of the team. She needs to learn how to write stories, write functional tests and make decisions about the project activities.
- The tester helps the client to write functional tests, she runs them regularly and sends the results to the rest of team in order to preserve the common knowledge.
- The consultant may be needed to help the team. The work of the consultant is to provide technical knowledge or assist with the process.
- The Big Boss is responsible for the project. She is needed to check the progress of the team activities and she must communicate regularly and honestly with the team. The big boss should take the time to listen to the problems of the concerns the team addresses.

**Agile teams**

A distinctive feature of agile software development is it focuses on people and social interactions. The values of Agile Manifesto promote a view focused on the individuals in charge of software development vision. Agile teams must be democratic groups where all members are considered peers at the same level, without a strict hierarchy in practice. The team members are empowered to make decisions as a collective and they should have cross functional skills, which increases their ability to self-organize. Agile team management intends to be performed in a coordinated approach.

Smaller teams can better adapt to democratic structures than bigger ones [2]. For other authors [3] this is the key by which agile teams work better with fewer members. The main challenge to migrate to agile methodologies is the change from the style of solo work to a collaborative environment that requires a slow transition and not without effort. The biggest challenge involves the transition of the project manager from her role of controller to a position as facilitator and collaborator and that the director should waives his authority.

Other authors such as Ruhnow [4] analyze the process of change towards an agile team and identify four kinds of individuals contrary to the establishment of good practice postures:

- The Cowboy is a brilliant programmer usually with an engineering background can rapidly incorporate characteristics to the application, even those not requested by the customer. This kind of workers tend to program avoiding practices of code quality.
- The Pessimist is a person who finds the weaknesses of applications. The rest of the team members do not want this individual to supervise their work.
- The Meeting Master has a business administration background in addition to the technical knowledge. She seeks long meetings to share developments in the project and she may induce dilly-dallying to the team.
- The Private is a competent developer who prefers to work with specific commands. She usually does not express her opinion if not directly asked about it.

It is important to encourage cowboys and pessimists to work with other members of the team. A good way to start the process of becoming an agile team is reducing the duration of iterations. In this way the team would be increasingly aware of the needing for collaboration, ask for help and share their activities. Then the developers should be explicitly encouraged to collaborate in the requirements definition, test implementation and code delivering. This change will come with conflict and it should even promoted in a constructive
way, debating processes and possible ways to implement agile methods in a most effective way for achieving project.

Cowboys should be encouraged to share their knowledge with the team and Pessimists should teach how to find vulnerabilities in order to decrease the importance of these roles in the activities. Privates tend to be more proactive under the new paradigm. Nevertheless, the Scrum manager or the coach may promote active participation of the whole team in the discussion of approaches to problem resolution.

With respect to the size, usually experts refer agile teams should be small but there are no limits to establish a given number of people that should not be exceeded. Tessem and Maurer [5] analyzed an agile team of 70 members with 36 developers and concluded that this team kept all the features that are considered agile. Thus, the size limits of agile teams are a matter of how individuals behave and are not directly relate to size. Nevertheless, under normal circumstances, it is easier to implement and maintain truly agile methodologies in small teams.

AUTO-ORGANIZING TEAMS

The self-organizing teams were described before the emergence of agile methodologies. However, due to their features they have been integrated as a piece within the description of what is meant by agile teams.

The self-organizing teams are teams of ten to fifteen people who take the responsibilities of their former supervisors. Their daily activities are guided by the vision of senior management. The individuals of these teams set their own timetables, show a great commitment to the company and are coordinated with other areas of the organization to carry out their activities in the most effective way.

The self-organizing teams are defined as teams that are informal and temporary, formed spontaneously around issues and are not part of a formal organizational structure. They have a strong sense of commitment and a shared purpose. The team decides what their activities are.

Self-organization is one of the principles behind the Agile Manifesto and has been identified as one of the critical success factors of agile projects. The self-organizing agile teams include individuals who manage their own workload, divide the work among themselves on the basis of need and the best fit, and participate in the decision-making. Self-organizing teams must have mutual trust, respect and the ability to reorganize several times to meet new challenges. Trust between team members is essential and agile practices help to reach it in comparison with traditional methodologies [6] that promote a competitive behavior.

However, agile teams do not operate without leaders. Leadership in self-organizing teams is meant to be soft and adaptable, by providing information and subtle guidance. Leaders of agile teams are responsible for setting direction, aligning people, obtaining resources and motivating teams.

When a task cannot be completed on time the work of the leader is to negotiate with the client, but never put the team members under pressure [7]. One cannot forget a leader is part of the team and she knows the performance of each member as they work together. In addition, any unforeseen difficulty to complete any task is her fault at least as much as the responsibility of developers.

The case of Yahoo is interesting because this company has been gradually migrating from traditional to self-organizing agile teams [8]. When asked about the new situation the 80% responded the goals were clearer, 89% that was more collaborative environment, 64% thought the value generated was greater and 68% believed less time was spent on tasks with no practical use [9]. Such experiences show how one can change practices in a reasonably short time with widespread satisfaction. Furthermore, in contrast to other studies that show how workers are more satisfied after the change and this increase in satisfaction impacts on the turnover of performance and productivity [10] Jobs satisfiers depend on ten main factors:

1. Opportunity for advancement
2. Ability to impact decisions that affect her
3. Ability to impact day-to-day firm success
4. Prospect to work on interesting projects
5. Wage
6. Connection between salary and performance
7. Job security
8. Workload
9. Relations with colleagues
10. Relations with customers

We conducted an anonymous survey in Scandinavian countries and the United Kingdom during the second half of 2013. We collected the results of 104 agile teams and 207 non-agile teams. The remaining 27 respondents were working in teams combining agile methodologies with traditional ones or they had migrated to agile methods less than one year ago.

The evidence shows how agile teams report higher levels in several factors that have a positive effect on job satisfaction, especially in ability to impact decisions that affect her, ability

![Fig 1. Job satisfaction by groups](image-url)

Source: Melnik & Maurer (2006)
to impact day-to-day firm success, workload and relations with customers. Our results agree with the ones obtained by Melnik and Maurer [10] in 2006. It seems agile methodologies have no negative impact in any of the ten job satisfiers but they have a sizeable positive impact only in five of them.

With respect to the overall satisfaction (Fig. 1) our findings are similar to those in [10]. Members of agile teams tend to report higher levels of satisfaction than members of non-agile teams (78.5% vs. 47.2%) and even more important, less than a half of individuals report to be dissatisfied with their current job in the case of agile teams (10.3% vs. 22.2%). This provides further evidence on the effect of agility on the satisfaction of workers.

Software development depends significantly on team performance, as it involves a high degree of interaction between the members. The transition to self-managing teams is a significant challenge that cannot be ignored. [11] The Dickinson and McIntyre teamwork model provides definitions of teamwork components.

Team orientation is about the tasks and the attitudes that team members maintain with respect to other members. It includes the acceptance of team norms, the level of group cohesiveness. Members may assign high priority to team goals and keenly participate in team organization. It is important to foster team participation in a way that team members are willing to be involved in the decision-making process and they consider it a part of their work at least as important as the proper software development activities.

Team leadership comprises making available direction, structure, and supporting other team members. In contrast to traditional software teams it does not refer to an individual with authority over the rest. Leadership in self-organizing teams can be performed by any member. All of them should be encouraged to explaining to other members what is needed from them and listening to their concerns. It is foreseeable some individuals will be more willing to explain activities and trying to coordinate the work. It is important to alternate these responsibilities in order to prevent the emergence of bossy attitudes that would diminish cooperation in the team. Nevertheless, team leadership is a necessary part of auto-organizing teams and should not be neglected.

Monitoring refers to examining the behavior and performance of the remaining team members and identifying when a member performs properly. This involves each one is individually proficient and that they must be aware about their colleagues performance. This may not be understood as a surveillance task, but as a good feedback mechanism that involves all the team. Positive feedback is as important as negative issues. Good performance serves as a model to mimic while bad habits can be eradicated in an easier way.

Feedback involves three actions: giving, looking for, and getting information among team members. It is as important seeking feedback as giving it. One team member should promote open discussion about her practices and she may be willing to accept positive and negative information. It is important to maintain a calm environment to promote confidence and information sharing. The absence of authoritarian figures helps to obtain such situation. It is especially helpful accepting time-saving suggestions.

Backup requires being accessible to support other team members. This implies that all the team understands other members’ tasks. It is as important to provide help as to seek it when needed. Not only mistakes have to be corrected but tasks unexpectedly time-consuming may be shared by other members of the team.

Coordination is reached when all team members carry out their activities in a timely and synchronized mode. In multi-team project it implies the coordination among all the related teams. All members should transmit relevant data for the correct performance of other members and ease the activities of other members as much as possible without interfering with the progress of their own tasks.

Finally, communication comprises the exchange of information using the proper terms. The purpose can be to acknowledge the receipt of information, to verify it and to repeat it in order to ensure understanding.

CONCLUSIONS

There are plenty of agile methods. We highlight scrum and extreme programming as two of the most extended ones that allow agile teams self-organizing with none or little external supervision and interferences. Agile teams focus on personal features in order to build teams in which confidence is a key factor among all the members.

The transition to agile teams from traditional software developing is not an easy process for developers and the employees who before played the role of project managers. Everyone has to adapt to the new roles and requirements. Particularly two kinds of individuals may be supported to change their behavior: cowboys and pessimists.

Auto-organizing teams show higher satisfaction levels, as we show with empirical evidence that shows a similar result as other surveys. This higher satisfaction involves a higher productivity of self-organizing agile teams.

Team members should combine in a proper way tasks not developed by traditional software teams: monitoring, feedback, backup, coordination and communication.

Confidence of the remaining members of the team and frequent cooperation allow self-organizing and overcoming the arising difficulties of any software development project.
REFERENCES


