Chapter 3
The Agile Manifesto

Abstract This chapter introduces the main ideas that form the basis for the agile approach. Originally, the agile approach offers a professional approach for software development that encompasses human, organizational, and technological aspects of software development processes. The main ideas of agile software development processes were first introduced by the Agile Manifesto and second by presenting specific agile practices that enable agile teams to accomplish their development task on high quality. In the chapter, we present the Agile Manifesto as was published for software development and shows how it can be implemented for any projects.

Keywords Human aspects · Organizational aspects · Technological aspects · Agile manifesto · Agile practices · Quality · Agile projects · Embrace change · Customer collaboration · Interaction

3.1 The Agile Manifesto

Figure 3.1 presents the Agile Manifesto. It was formulated by seventeen software practitioners, who gathered together in February 2001 in the Wasatch Mountains of Utah, in order to find common ground for their perceptions of software development processes and to formulate what is common to what some of them have already implemented in different software organizations. The outcome of that meeting was the Agile Manifesto, which presents an alternative approach for software development processes than the approaches that had been applied during the past 40 years, from the early stages of the development of complex software systems.

The mere formulation of the Agile Manifesto implies that though there are agreed upon, common and shared principles and ideas, this common basis can be applied differently by specific development methods. Indeed, the Agile Manifesto is applied by different agile methods, such as Extreme Programming (Beck 2000) SCRUM (Schwaber 2004), Lean (Poppendieck and Poppendieck 2003), DSDM, Adaptive Software Development, Crystal, and others.

In what follows, we examine the Agile Manifesto.

O. Hazzan and Y. Dubinsky, *Agile Anywhere*, SpringerBriefs in Computer Science,
DOI 10.1007/978-3-319-10157-6_3

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

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

That is, while there is value in the items on
the right, we value the items on the left more.

Fig. 3.1 Manifesto for agile software development

3.1.1 Individuals and Interactions Over Processes and Tools

This principle guides us to focus on the individuals involved in the development process rather than on the process and/or the tools. In practice, this principle guides software practitioners to give high priority to the people who participate in the development process as well as to their interaction and communication, when they develop, interact, think, discuss, and make decisions with respect to different issues related to the software development process and environment. In other words, according to this principle, one of the first considerations that should be taken into account when a decision related to the development process is made, is the influence of the decision's outcome on the people who are part of the development environment as well as on their relationships and communication.

For example, instead of investing efforts in the maintenance of a development method by using state of the art hard-to-use tools, that specify difficult-to-follow procedures that their output is useless, efforts should be channeled to the construction of a development environment that enables each of the participants (teammates, customers, management) to understand the development process, to become part of it, to contribute to it and to collaborate with all the other project stake holders.

3.1.2 Working Software Over Comprehensive Documentation

This principle delivers the message that the main target of software projects is to produce quality software products. This idea has three main implications.

First, agile software development focuses on the development itself and the creation of only these documents that are needed for the development process. Some of these essential documents, according to their characteristics and usefulness, are

posted on the wall of the agile collaborative workspace so that they will be accessible to *all* the project stake holders *all the time*.

Second, agile software development processes start coding as soon as possible in order get some sense of the developed product. This early development enables the teammates and the customer to improve their understanding of the developed product and to proceed with the development process on a safer ground.

Third, from the customers' perspective, this principle advocates that customers should get a bug-less high-quality product that meets their requirements. This, of course, has direct implication on quality-related activities that agile teams perform.

As can be seen, this principle supports the first principle of the agile manifesto, by binding the people who participate in the development process with the actual development process. Such a connection inspires a culture in which software quality is one of its main values.

The importance of this principle is highlighted when its implications are compared with development processes which postpone either the beginning of the development stage (sometimes in several years) or the product quality-related activities (mainly testing). In the first case, the fact that the project production starts only after a lot of documentation has been produced, that presumably, but not in practice, captures all the customer requirements, neglects the reality that software development processes are characterized by many changes and are based on a gradual learning process. As a result, in many cases, development processes, that prepare in advance a lot of documentation without starting the actual development, do not provide eventually the customer with the needed system and in practice, inconsistencies exist between the project documentation and the actual product. In the second case, the postponement of quality-related activities leads to a situation in which the practitioners involved in the development process cannot cope successfully with the complexity of the testing activity both from cognitive and managerial perspectives.

3.1.3 Customer Collaboration Over Contract Negotiation

This principle changes the perception of the customer role in software development processes. It guides agile software development methods to base the development process on an on-going and on a daily basis contact with the customer. Such a close contact with the customers enables to cope successfully with the frequent changes that characterize software projects. This principle also points at a conception change with respect to the nature and formulation of software product contracts.

Human relationships, mainly between the customer and the management, are emphasized by this principle of the manifesto. These relations have, in turn, direct implications of the development team, which should employ specific practices to ensure these kinds of relationships and communication. These practices, when employed on a daily basis, influence directly the culture of agile organizations.

Thus, by referring to contact- and communication-related issues that aims at ensuring that the customer gets the desired product, this principle of the Agile Manifesto further supports the second principle of the agile manifesto.

3.1.4 Responding to Change Over Following a Plan

This principle guides agile software development methods to establish a development process that copes successfully with changes that are introduced during the development process, without compromising the high quality of the developed product. The rationale for this principle is derived from the recognition that customers cannot predict a priori all their requirements; therefore, a gradual process, by which the requirements are understood by the customer and are delivered to and shared with the team, should be established. Accordingly, agile software development methods inspire a process that enables to introduce changes in the developed product, that emerged based on an improved understanding of the software requirements, without necessarily increasing the cost of change introduction.

3.2 Application to Agile Projects

Based on common understandings encapsulated by the Agile Manifesto, the agile approach is applied by several basic practices that support any projects (with some modifications according to the project theme). In this section, some of these practices are introduced.

Whole team. The practice of whole team means that the project team (including all role holders and the customer) communicate in a face-to-face fashion as much as possible. It is applied in several ways.

First, the development team is colocated in a collaborative workspace—a space which supports and facilitates communication. Second, all team members participate in all the product presentations to the customer, hear the customer requirements and are active in the actual process planning. Third, role holders, that traditionally belong to separate teams (e.g., testers and designers), are integrated into the team and process.

On a daily bases, each day, during the working hours, the team is located in one space; in addition, each team member has a private space for personal tasks and professional tasks that should be carried out individually and personally. The walls of the development workspace serve as a communication means, constituting an informative and collaborative workspace. Thus, all the project stake holders can be updated at a glance at any time about the project progress and status. In addition, the entire team holds daily stand-up meetings, which usually take place in the morning. In these meetings, each team member presents in 2–3 sentences the

status of his or her tasks and what he or she plans to do during the day to come, both with respect to the (development) tasks and his or her personal role.

Short releases. Agile processes are based on short releases (of about 2 months), divided into short iteration of one or 2 weeks, during which the scope of what has been decided to be delivered in the said iteration is not changed. At the end of each iteration, the deliverable is presented to the customer and the customer provides feedback to the team and sets the requirements to be delivered in the next iteration.

The detailed plan of each short iteration is carried out during a Business Day which is specifically allocated for this purpose at the beginning of each iteration. In the Business Day, all the project stake holders participate—customer, team members, users, management representatives, representative of related projects, and so on. The Business Day includes three main parts: a presentation of what was delivered in the previous iteration along with any relevant measures taken, a short reflective session in which the project process performed so far is analyzed and lessons are learnt, and the actual planning of the next iteration. At the end of the Business Day, a balanced workload is ensured among all team members.

The nature of the activities that take place during the Business Day, and the fact that a Business Day takes place every week or 2 weeks, enable all the project stake holders to construct gradually their knowledge related to the project deliverable and process, based on what they see, hear, and perform during each iteration. Specifically, during this process, the teammates improve their understanding of what should be performed, mainly due to the fact that they hear the requirements directly from the customer during the planning session.

Time estimations. In agile projects, two important practices are performed with respect to time estimation. First, the teammate, who is in charge of a specific task, also estimates the time needed for it; this practice increases the team member's responsibility and commitment to the project. Second, tasks are formulated in a way that their time estimation is possible to be set in hour resolution. This fact is important because the greater a task is, the harder it is to estimate its development time, and vice versa: the smaller the segment estimated, the more accurate its time estimation is. Consequently, the progress pace can be planned more precisely. This inspires a culture that delivers the message that plans can be set and followed in such a way that deadlines should not be postponed.

From the team perspective, since time estimations are performed at the Business Day with full team attendance, all teammates know what each team member has committed to in terms of tasks and time estimations. This fact increases the project transparency, and consequently, the teammate's responsibility to perform well. Further, the load balance, that is ensured among all team members, further reinforces trust and communication among team members.

Measures. The agile processes are accompanied with measures on which all the project stake holder decide according to their needs.

Measures enable the team to improve the process, and consequently, the deliverables. Measures also convey the message that the process should be monitored

and that this monitoring should be simple, transparent, and known to all the project stakeholders.

Customer collaboration. The agile approach welcomes the customer to become part of the process. The target is to get an ongoing feedback from the customers and to move on according to their needs. This avoids the need to speculate the customers' needs, which may lead to incorrect working assumptions.

This practice implies that in agile projects all team members have access to the customer during the entire process. This direct communication channel increases both the individual interaction and the chances that the requirements are communicated correctly. Consequently, it helps the teammates to cope successfully with changes: first, there is no need to speculate the customer's needs; second, the overhead of dealing with change introduction at later stages is reduced significantly.

3.3 Summary

The Agile Manifesto established a framework, based on which a cultural (Hazzan et al. 2010) and organizational (Dubinsky et al. 2010; Talby and Dubinsky 2009) changes were introduced into the profession of software engineering. In the spirit of this book, we propose that a similar manifesto can be formulated for any theme, according to its specific characteristics and needs. Nevertheless, the spirit of the actual application of agility does not change from theme to theme.

References

Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Boston (2000)

Dubinsky, Y., Yaeli, A., Kofman, A.: Effective management of roles and responsibilities: driving accountability in software development teams. IBM Syst J **54**(2), 1–4 (2010)

Hazzan, O., Seger, T., Luria, G.: How did the creators of the agile manifesto turn from technology leaders to leaders of a cultural change? AgileQ, InfoQ, http://www.infoq.com/articles/manifesto-originators (2010). Accessed 18 Feb 2010

Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit. Addison-Wesley Professional, Boston (2003)

Schwaber, K.: Agile Project Management with Scrum. Microsoft Press, US (2004). (Developer Best Practices)

Talby, D., Dubinsky, Y.: Governance of an agile software project. 31th International Conference of Software Engineering, ICSE, Workshop on Software Development Governance (SDG), Vancouver, Canada (2009)