Introduction to Agile Software Development

Patrick P. Bucher

Summary

In February 2001, representatives from Extreme Programming, SCRUM, and other software development methodologies got together to find an "alternative to documentation driven, heavyweight software development processes" (Highsmith 2001). The *Manifesto for Agile Software Development* was the result of this gathering, setting the following values in contrast to values held important before (Beck 2001A):

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

The group also states twelve principles behind the Agile Manifesto (Beck 2001B):

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The Agile Manifesto was signed by thousends of developers since then, and agile methods, such as Kanban and Scrum, are in widespread use ever since.

Sources

- Jim Highsmith, History: The Agile Manifesto, 2001, Agilemanifesto.org
- Kent Beck et al., The Agile Manifesto, 2001, Agilemanifesto.org
- Kent Beck et al., Principles behind the Agile Manifesto, 2001, Agilemanifesto.org

Questions

1. Is there still a need for a boss in agile projects?
   - In the *Toyota way* of agile projects, the boss is supposed to be a servant leader–somebody that helps the team members to solve their issues on their own.
   - In Scrum, there is no notion of line management. When existing teams are transformed into a Scrum team, the former team leader often takes over the role of the Scrum master.
   - In general, the classic responsibilities of a boss are distributed to different team members in agile teams.
2. Are there projects that are a bad fit for agile methods?
   - Agile methods are not a good fit for projects dealing with deterministic problems. If it is possible to make a plan in advance–the requirements are stable, the technologies well known–there is no point in agility.
3. What is a good size for an agile team?
   - A size of four to seven is a good fit for a Scrum team.
4. How can non-functional requirements be considered in an agile project?
   - It is crucial to always point out the business value of a requirement.
   - For security, a risk analysis could be done–with an estimation of the costs a security breach might cause.
   - For performance: Show impact in production. Are customers moving away or users getting disappointed because of slow transaction time?
   - Also make sure to show the risks and benefits for the business as a whole.
5. How can it be ensured that multiple agile teams–with their own goals and commitments–work well together?
   - Specialists of different teams join up to form guilds (communities of practice) that deal with issues common to all teams and exchange knowledge.