ORCHESTRATE
Solutions for higher performance!

Agile Methodologies & Key Principles

Series-II

Introduction

In February 2001, a group of 17 software developers met at the Snowbird resort in Utah to discuss lightweight development methods. This grand association of knowledgeable minds later led to the publishing of the “Manifesto for Agile Software Development”.

Declaration of four core, guiding values of the Agile Manifesto by its authors

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

The Agile Manifesto can be broadly classified as

- **Individuals and interactions** Self-organization and motivation are important, as are interactions like co-location and pair programming.
- **Working software** Working software is more useful and welcome than just presenting documents to clients in meetings.
- **Customer collaboration** Requirements cannot be fully collected at the beginning of the software development cycle, therefore continuous customer or stakeholder involvement is very important.
- **Responding to change** Agile methods are focused on quick responses to change and continuous development.

Some of the authors formed the Agile Alliance, a non-profit organization that promotes software development according to the manifesto's values and principles—introducing the manifesto on behalf of the Agile Alliance.

Agile Manifesto 12 Principles Explained

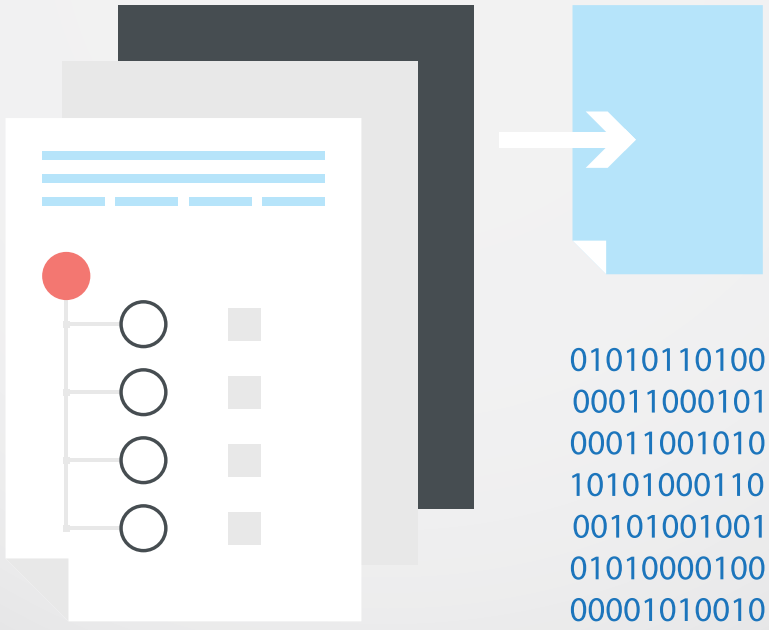
1 **Customer satisfaction by rapid delivery of useful software** Teams work together better when they trust each other. It is common for tension to exist between the customer and the delivery team. When the customer is satisfied by constant delivery of valuable software early rather than later, trust is built.

2 **Welcome changing requirements, even late in development** This principle will scare teams who are used to Waterfall projects. At first glance, it seems odd to welcome change late in the development process. First, we must be successful at implementing the first two principles in this section. If this is not happening, welcoming change is impossible. "Late in development" means late in the release of the complete product.

Scrum delivers features in short sprints. We do not welcome changes in an in-process sprint. Because we are delivering features in short cycles, change is part of the whole process. In Scrum, the change is directed by the product owner. It is up to the product owner to understand what the competitive advantage is for each feature in the backlog.

3 **Working software is delivered frequently (weeks rather than months)** It is important to deliver software frequently. Scrum is built around this principle. Under Scrum, features are delivered in sprints of two to four weeks, with a preference toward two weeks.

4 **Close, daily cooperation between business people and developers** The whole team needs to be available to each other. Scrum uses the daily stand-up meeting as a critical communication mechanism. Here, the team reports what was accomplished since the last meeting, what will be accomplished by the next meeting, and whether there are any impediments to completing the features in the sprint. This meeting exposes issues early so they can be addressed before they become critical.



5 **Projects are built around motivated individuals, who should be trusted** This is an extension of self-organizing teams. There are some important words in this principle. No one would ever admit to not being motivated. The "servant leader" pays attention to the aspirations and goals of the team members and aligns these goals with project needs wherever possible. People perform best when they are doing something they are passionate about.

A good servant leader also shelters the team from outside distractions. In Scrum, a team commits to completing a set of features. Anything that distracts from this is a risk. By being there for the team, the servant leader provides them with the environment and support needed for success. Trust is not automatic but is built over time—and is easy to lose. The team members must trust each other and be comfortable with conflict.

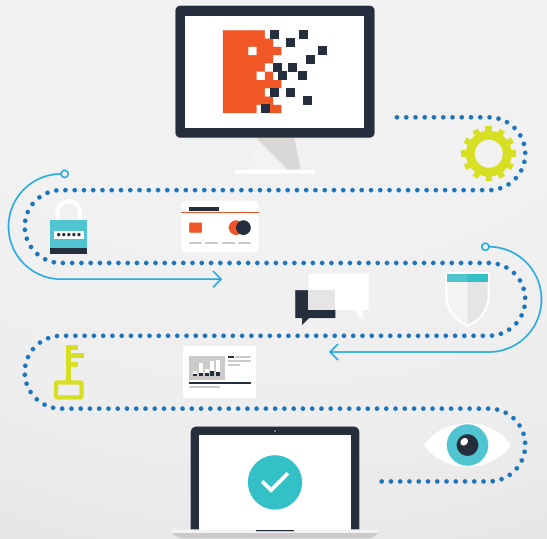
6 **Face-to-face conversation is the best form of communication (co-location)** This principle was authored before geographically separate teams were common. Today, with offshore teams and teams that are divided across the country and the globe, regular face-to-face communication is often not possible. Online meetings and instant messaging tools are available that improve communication when teams are separated. Meetings that include the whole team may be planned so that face-to-face communication is possible.

This does add cost to the project, because portions of the team need to travel to a central location for the meeting. This approach is helpful for important meetings like sprint and release planning. When offshore resources are used, portions of the offshore team may be rotated to the U.S. for a period of time. This allows team members to interact personally and get to know each other. It allows the offshore team to return home with firsthand experience that helps the remote team gain valuable insight. This is often a win-win situation, because offshore team members look forward to an experience in the U.S.

7 **Working software is the principal measure of progress** Software must not only be valuable and delivered often, it must be working or done. Scrum requires the features to meet a team-defined “Definition of Done”. Ideally, this should mean that the feature is potentially shippable.

8 **Sustainable development, able to maintain a constant pace** Sponsors, developers, and users should be able to maintain a constant pace indefinitely. As teams build trust and build and deliver software over and over, a constant pace that is sustainable, without overtaxing anyone, will emerge. This allows the team to work forever—or until enough value has been added to the product.

An important aspect of this is regular releases of a product. If a team can deliver a shippable product each quarter, for example, it makes conversations with the customer much easier. The team learns that they ship every 12 weeks. When a feature request doesn't fit into the current release, it is only a short wait till the next one.



9 **Continuous attention to technical excellence and good design** We need to pay close attention to technical excellence and design as our product evolves. There is a balance between "Building the right thing" and "Building the thing right." We must also be wary of delivering fragile systems. If we make a few changes and our application falls apart like a house of cards, we are not in a good place. Extreme Programming and, to some degree, Scrum recommend test-driven development and automated builds as a way to avoid fragile solutions.

10 **Simplicity—the art of maximizing the amount of work not done—is essential** Agile is all about doing the right amount of something at any given time, and no more. We should author user stories small enough to get the job done and no more. We should build what we know we need now. We should not build some huge framework we think we may need someday. It is critical to have a complete and thorough understanding of the software frameworks we use. Code is evil, and we can eliminate quite a bit if we have a good understanding of our chosen frameworks.

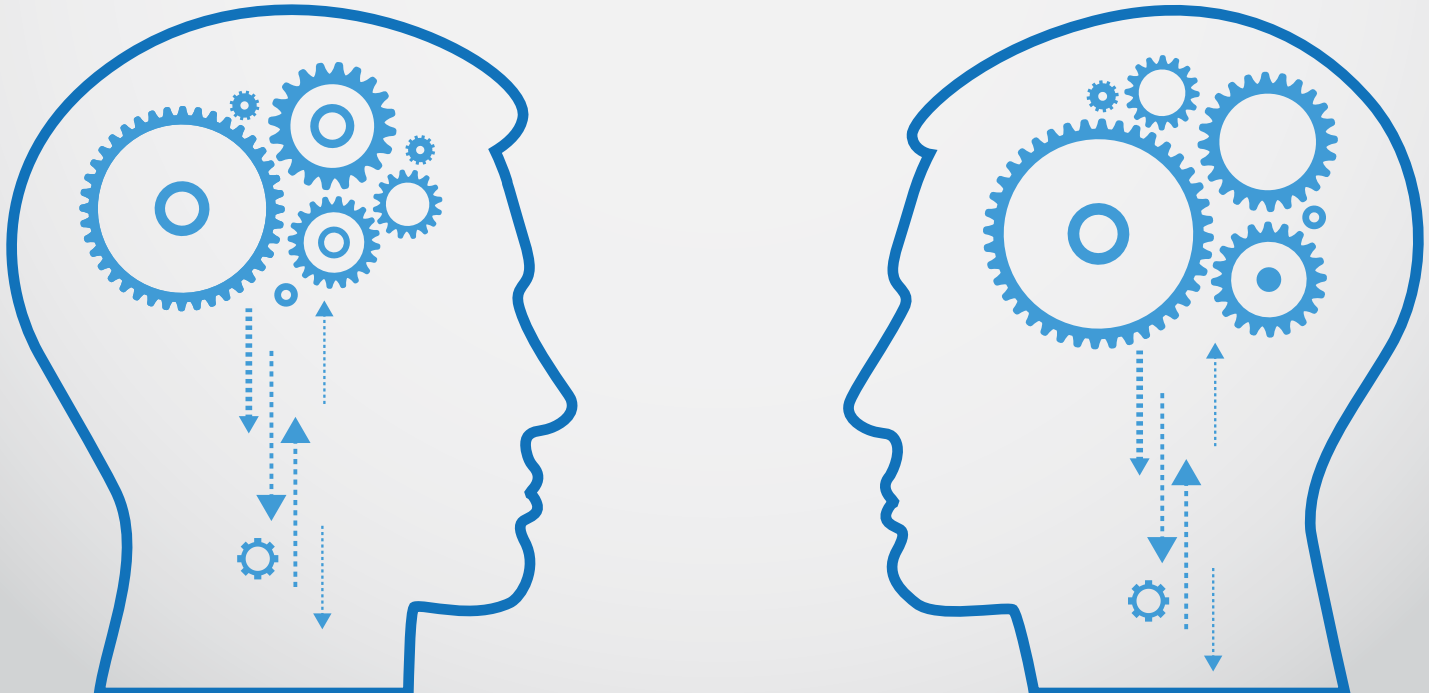
11 **Self-organizing teams** The team knows the best way to get something done. They are the experts. However, this does not mean the right outcome will happen on its own. Each individual is at a different place in his or her personal growth and career. The term "servant leader" has emerged in the Agile community and replaced the typical command-and-control project manager. Self-organizing teams do not happen automatically. They emerge under the proper guidance and advice of a servant leader.

12 **Regular adaptation to changing circumstance** Scrum uses the retrospective for this purpose. Teams often need help for this activity to be effective. People may be challenged when it comes to engaging in true self-reflection. This is all part of the Agile journey. Each of the Agile principles are interrelated. The retrospective is the perfect place for the team to reflect and improve. It is up to the ScrumMaster to elicit self-reflection. Once we have identified areas for improvement, we need to really improve. If teams spend time reflecting and do not improve, they see the retrospection as a waste of time.

Conclusion

In many ways, the Agile Manifesto gives us a road-map and lays a firm foundation for efficient software development. There are naysayers among those who swear by traditional methods; but these criticisms do not hold water because the entire agile movement rests on robust methodologies and concepts. So what does this augur for the future? No one can tell with certainty.

Agility encompasses believing and relying on one's ability to respond to unpredictable events, rather than banking on the competence to indulge in pre-planning. At the end of the day, the methodologies remind us that even though we create and work with software, the human element, and the resultant collaboration it enhances, is all too important in the larger scheme of things.



About Orchestrate

Orchestrate is a US based business process management organization with Headquarters in Dallas, Texas. Orchestrate offers services to the diverse outsourcing requirements of clients in an extensive range of businesses including IT, finance, mortgage and contact center. We provide a comprehensive suite of technology and services to our clients that help accelerate sales and boost their profit. Our comprehensive solutions and services help SMEs and enterprises to implement technologies and processes that boost their profitability across the organization.

ORCHESTRATE
Solutions for higher performance!

1330 Capital Parkway, Carrollton TX 75006

Toll Free: 800-232-5130 | success@orchestrate.com

www.orchestrate.com