**Agile Values, Principles and Practices[[1]](#footnote-1)**



**Agile Manifesto (2001)/Agile Values**

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value[[2]](#footnote-2):

* 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.”

**Principles behind the Agile Manifesto**

“We follow these principles[[3]](#footnote-3):

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.

## Agile Values, More Detail

## Individuals and Interactions[[4]](#footnote-4)

“Individuals and interactions are essential to high-performing teams. Studies of "communication saturation" during one project showed that, when no communication problems exist, teams can perform 50 times better than the industry average. To facilitate communication, agile methodologies rely on frequent inspect-and-adapt cycles. These cycles can range from every few minutes with pair programming, to every few hours with continuous integration, to every day with a daily standup meeting, to every iteration with a review and retrospective.

Just increasing the frequency of feedback and communication, however, is not enough to eliminate communication problems. These inspect-and-adapt cycles work well only when team members exhibit several key behaviors:

* respect for the worth of every person
* truth in every communication
* transparency of all data, actions, and decisions
* trust that each person will support the team
* commitment to the team and to the team’s goals

To foster these types of behavior, agile management must provide a supportive environment, team coaches must facilitate their inclusion, and team members must exhibit them. Only then can teams achieve their full potential.

For teams to achieve these types of behavior is more difficult than it might appear. Most teams avoid truth, transparency, and trust because of cultural norms or past negative experiences from conflict that was generated by honest communications. To combat these tendencies, leadership and team members must facilitate positive conflict. When teams engage in positive conflict, they not only foster more productive behavior, but also work to achieve several other benefits:

* Process improvement depends on the team to generate a list of impediments or problems in the organization, to face them squarely, and then to systematically eliminate them in priority order.
* Innovation occurs only with the free interchange of conflicting ideas, a phenomenon that was studied and documented by Hirotaka Takeuchi and Ikujiro Nonaka, the godfathers of Scrum.
* Resolution of conflicting agendas occurs when teams align around common goals and surface their concerns and potential conflicts.
* Commitment to work together happens only when people agree on common goals and then struggle to improve both personally and as a team.

This last bullet, about commitment, is especially important. It is only when individuals and teams are committed that they feel accountable for delivering high value, which is the bottom line for software development teams. Agile methodologies facilitate commitment and self-organization by encouraging team members to pull items from a prioritized work list, manage their own work, and focus on improving their work practices. These practices form the basis of self-organization, which is the driving force for achieving results in an agile team.

To create high-performing teams, agile methodologies value individuals and interactions over processes and tools. Practically speaking, all of the agile methodologies seek to increase communication and collaboration through frequent inspect-and-adapt cycles. However, these cycles work only when agile leaders encourage the positive conflict that is needed to build a solid foundation of truth, transparency, trust, respect, and commitment on their agile teams.”

**Working Software over Comprehensive Documentation****[[5]](#footnote-5)**

“Working software is one of the big differences that agile development brings. All of the agile methodologies that are represented in the Agile Manifesto stress delivering small pieces of working software to the customer at set intervals.

All agile teams must establish what they mean when they say "working software," which is frequently known as the definition of done. At a high level, a piece of functionality is complete only when its features pass all tests and can be operated by an end user. At a minimum, teams must go beyond the unit test level and test at the system level. The best teams also include integration testing, performance testing, and customer acceptance testing in their definition of what it means to be done with a piece of functionality.

One CMMI Level 5 company, that already has one of the lowest defect rates in the world, has shown through extensive data on many projects the benefits of agile practices. Specifically, they were able to systematically double the speed of production and reduce defects by 40 percent by taking the following actions: This from a company.

* Define acceptance tests when defining the feature.
* Implement features serially and in priority order.
* Run acceptance tests on each feature as soon as they are implemented.
* Fix bugs that are identified as highest priority as soon as possible.

The Agile Manifesto recommends that teams deliver working software at set intervals. By agreeing as a team on what success means is one of the practical ways that agile teams bring about the high performance and high quality that is needed to accomplish this goal.”

**Customer Collaboration over Contract Negotiation****[[6]](#footnote-6)**

“Over the past two decades, project success rates have more than doubled worldwide. These improvements occurred as a result of smaller projects and more frequent deliveries, which allowed customers to provide feedback on working software at regular intervals. The writers of the manifesto were clearly on to something when they stressed that getting the customer involved in the software development process is essential to success.

The agile methodologies foster this value by having a customer advocate work hand-in-hand with the development team. The first Scrum team, for example, had thousands of customers. Because it was not feasible to involve them all in product development, they selected a customer proxy, called a product owner. The product owner represented not only all the customers in the field, but also management, sales, support, client services, and other stakeholders. The product owner was responsible for updating the list of requirements every four weeks as the team released working software, taking into account the current reality and actual feedback from customers and stakeholders. Customers actively helped shape the software that was being created.

The first XP team began with an internal IT project. The XP team was able to have a company end user on their team work with them daily. About 10 percent of the time, consultants and internal teams can find an end user who can work with the team on a day-to-day basis. The other 90 percent of the time, they must appoint a proxy. This customer proxy, whom XP teams call the customer, works with end users to provide a clear, prioritized list of features for developers to implement.

It is through daily customer (or customer proxy) collaboration that industry data shows that agile projects have more than twice the success rate of traditional projects on average worldwide. Because agile methodologies recognize the value of customer engagement, they have created a place on their development teams that is specifically for the customer representative.”

**Responding to Change over Following a Plan****[[7]](#footnote-7)**

“For teams to create products that will please customers and provide business value, teams must respond to change. Industry data shows that over 60 percent of product or project requirements change during the development of software. Even when traditional projects finish on time, on budget, with all features in the plan, customers are often unhappy because what they find is not exactly what they wanted.

"Humphrey’s Law" says that customers never know what they want until they see working software. If customers do not see working software until the end of a project, it is too late to incorporate their feedback. Agile methodologies seek customer feedback throughout the project so that teams can incorporate feedback and new information as the product is being developed.

All agile methodologies have built-in processes to change their plans at regular intervals based on feedback from the customer or customer proxy. Their plans are designed to always deliver the highest business value first. Because 80 percent of the value is in 20 percent of the features, well-run agile projects tend to finish early, whereas most traditional projects finish late. As a result, customers are happier, and developers enjoy their work more. Agile methodologies are based on the knowledge that, in order to succeed, they must plan to change. That is why they have established processes, such as reviews and retrospectives that are specifically designed to shift priorities regularly based on customer feedback and business value.”

**Agile Principles, More Detail[[8]](#footnote-8)**

1. **“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.** As obvious as this principle may seem, it’s often violated in traditional software development. It’s important to remember that customers are asking you to deliver working software that adds value; they don’t want a prototype or a set of documents. The earlier you can start delivering working software, the earlier you can begin satisfying your customer.
2. **Welcome changing requirements, even late in development.** Agile processes harness change for the customer’s competitive advantage. Your customers are competing in a dynamic market, and therefore they may have to change the requirements for their software in order to gain a competitive advantage. It is important to note that you should welcome changing requirements, but no one said this change is free.
3. **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.** Have you ever shown your customer software for the first time and received no feedback? In most cases, you receive feedback—sometimes minor, but usually major. The trick is to deliver software early so that you can get feedback early. This early feedback can save you re-work down the road.
4. **Business people and developers must work together daily throughout the project.** This principle is careful to say business people and not the customer. In most cases, it would be impractical to work with the customer on a daily basis; but generally there are multiple business proxies. These proxies may not know everything about the customer’s wants and needs, but they usually know more about the business needs than the developers do. These proxies may be analysts, product managers, or program managers. The key is to maintain constant communication between the developers and the business people to ensure that the project never goes off track—not even for a day.
5. **Build projects around motivated individuals.** Give them the environment and support they need, and trust them to get the job done. Remember, people aren’t resources. Software development is different from manufacturing. Software development is more of an art. Project teams need to be motivated and trusted. If you have motivated team members they will find a way to give you their best; and that’s what an Agile process needs—everyone’s best.
6. **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.** Instant messaging or the telephone should never replace face-to-face communication. A lot of context is lost in communication over email and instant messaging—not to mention the fact that ambiguity increases with nonverbal communication. Face-to-face communication also lets you run with less formal documentation.
7. **Working software is the primary measure of progress.** If you recall, the customer is primarily interested in working software. So why would you measure progress in terms of anything else? Today, the progress of most software development efforts is measured in terms of their plan. When requirements are complete, the managers say the project is 30 percent complete. In a plan-driven world, this may be correct; but in a value-driven world, where the value is the working software, the project is 30 percent complete when 30 percent of the required functionality is coded, integrated, tested, and deployed. This is a fundamental difference between the Agile value-driven world and the traditional plan-driven world.
8. **Agile processes promote sustainable development.** The sponsors, developers, and users should be able to maintain a constant pace indefinitely. In traditional development, the team often has to work late toward the end of a project, although at the beginning of the project they may have taken 2-hour lunch breaks. This is primarily due to the way project activities are distributed across the project’s lifecycle. There isn’t much for developers to do at the beginning of the project, but at the end everything is put on their shoulders because of tight delivery schedules. With Agile development, you deliver every two weeks or so, and development begins with the first iteration. Efforts are distributed more consistently throughout the project lifecycle, which leads to a constant development pace for the team.
9. **Continuous attention to technical excellence and good design enhances agility.** A successful gymnast needs strong muscles. Similarly, technical excellence is an essential enabler for a truly Agile software development process. For example, extensible designs and architectures make it much easier to build the product in an evolutionary manner. Automated testing frameworks are needed to ensure that refactoring one part of the system doesn’t affect other parts. Continuous integration is essential if you want assurance that your software is working after every change.
10. **Simplicity—the art of maximizing the amount of work not done—is essential.** No code means no bugs. The more code you write, the more bugs your code may have. If something isn’t essential to the product, then don’t build it. Some developers tend to develop massive underlying frameworks and infrastructures in the system under the assumption that those elements may be needed in the future. The key is simplicity: try not to develop anything that isn’t essential to the features you’re developing now. Remember, the more time you invest in anything, fact that you don’t need a piece of code or that you need to change it.
11. **The best architectures, requirements, and designs emerge from self-organizing teams.** In traditional software development, analysts write requirements, and architects lay out the architecture of the system. Then the requirements and architectures are communicated to the team in a document. In the Agile world, we encourage teams to self-organize. True self-organization involves giving the whole team the task and asking them, as a team, to complete the task without specifying who should do what—they’re left to self-organize. It will naturally occur that architects will lead the discussion when it comes to architecture, but now everyone is free to challenge them and suggest new ideas that may enhance the architecture the architects would have come up with on their own. This form of collaboration also increases the knowledge transfer within the team.
12. **At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.** We believe this is probably the most important principle of agility. The idea of always reflecting on what you’re doing and trying to Figure out better ways to do things is the essence of continuous improvement. Without continuous improvement, people and organizations remain at a status quo. If you adopt only one thing that will make your process better, regularly reflect on your process as a team. You need to identify what you’re doing well and continue doing it, and you need to identify what you’re doing poorly and improve it.”

**History of Agile**



**Comparison between Agile Software Development Styles[[9]](#footnote-9)**



**Comparison between other Agile Software Development Styles**[[10]](#footnote-10)



**Self-Assessment: Agile Manifesto**

* Define and describe the four values of Agile as stated in the Agile Manifesto.

**Self-Assessment: Agile Mindset**

Learner needs to understand that Agile is a mindset, not a methodology[[11]](#footnote-11).

* History & Mindset: Understand the Agile Mindset and history.
* Individuals & Interaction: Understand that individuals, working in teams, create value. The learner will recognize the purpose, concepts and theory supporting this concept, along with applications supporting it.
* Value-Driven Development: Understand why Agile development focuses so heavily on working products, including the role of value in driving work. Understand iterative and incremental approaches, continuous integration and continuous delivery.
* Customer & User Involvement: Understand concepts and theory for working with stakeholders, customers and users. Understand the related benefits and apply potential approaches.
* Planning & Adapting: Understand the value, concept and theory for learning and adapting at various levels, including product, process, team and organization.

**Comparison between Agile and Traditional Methods**[[12]](#footnote-12)



1. Greg Smith & Ahmed Sidky, “Agile Values, Primciples, and Practices”, [https://msdn.microsoft.com/en-us/library/hh273032(v=vs.88).aspx](https://msdn.microsoft.com/en-us/library/hh273032%28v%3Dvs.88%29.aspx) [↑](#footnote-ref-1)
2. Manifesto for Agile Software Development, <http://agilemanifesto.org/> [↑](#footnote-ref-2)
3. Principles Behind the Agile Manifesto, <http://agilemanifesto.org/principles.html> [↑](#footnote-ref-3)
4. Jeff Sutherland, <http://www.scruminc.com/agile-manifesto-elaborated-2/> & <https://msdn.microsoft.com/en-us/library/dd997578.aspx> [↑](#footnote-ref-4)
5. Jeff Sutherland, <http://www.scruminc.com/agile-manifesto-elaborated-2/> & https://msdn.microsoft.com/en-us/library/dd997578.aspx [↑](#footnote-ref-5)
6. Jeff Sutherland, <http://www.scruminc.com/agile-manifesto-elaborated-2/> & https://msdn.microsoft.com/en-us/library/dd997578.aspx [↑](#footnote-ref-6)
7. Jeff Sutherland, <http://www.scruminc.com/agile-manifesto-elaborated-2/> & <https://msdn.microsoft.com/en-us/library/dd997578.aspx> [↑](#footnote-ref-7)
8. Greg Smith & Ahmed Sidky, “Agile Values, Primciples, and Practices”, [https://msdn.microsoft.com/en-us/library/hh273032(v=vs.88).aspx](https://msdn.microsoft.com/en-us/library/hh273032%28v%3Dvs.88%29.aspx) [↑](#footnote-ref-8)
9. J. R. J. W. Pekka Abrahamsson, Outi Salo. Agile software development methods. VTT PUBLICATIONS 478, 2002, page 89. [↑](#footnote-ref-9)
10. J. R. J. W. Pekka Abrahamsson, Outi Salo. Agile software development methods. VTT PUBLICATIONS 478, 2002, page 90. [↑](#footnote-ref-10)
11. ICAgile Track: Agile Fundamentals, The International Consortium for Agile. [↑](#footnote-ref-11)
12. M. A. Awad. A comparison between agile and traditional software development methodologies. Master’s thesis, The University of Western Australia, 2005, page 35. [↑](#footnote-ref-12)