

Manifesto for Agile Software Development

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.**

Principles behind the Agile Manifesto

We follow these principles:

Satisfy the Customer

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Embrace Change

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Frequent Delivery

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Collaboration

Business people and developers must work together daily throughout the project.

Trusted & Motivated People

**Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.**

Conversation to Communicate

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Tested & Working Software

**Working software is the primary measure of
progress.**

Sustainable Pace

Agile processes promote sustainable development.

The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Attend to Quality

Continuous attention to technical excellence and good design enhances agility.

Simplify

Simplicity -- the art of maximizing the amount of work not done -- is essential.

Self-Organizing Teams

**The best architectures, requirements, and designs
emerge from self-organizing teams.**

Continuous Improvement

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Engineering Practices

- Continuous Integration with Automated Builds
- Smoke Testing / Build Verification Tests
- System Metaphor
- Domain Driven Design / Emergent Design / Evolutionary Design
- Behavior Driven Development
- Test Driven Development
- Pair Programming
- Code Reviews
- Automated Software Metrics
- Source Version Control
- Issue / Bug Tracking
- Configuration management
- Unit Testing
- Integration Testing via Mock/Fake/Stub sub-systems
- Exploratory Testing
- Story Testing / Acceptance Tests / Automated Regression Test

- Scrum (Process Framework)
- Extreme Programming (XP) Framework
- Stand-up Meeting
- Velocity Based Planning
- Team estimation in relative units
- Iteration Demo & Feedback
- Information Radiators (Big Visible Charts)
- Cross-Functional Team
- Team based work flow / Teamwork
- Co-located Team / Common Workspace
- Design Improvement via Refactoring
- Small Releases
- Collective Code Ownership
- Coding Conventions & Standards
- Simple Design (Once & Only Once, YAGNI, etc)
- User Stories

Ref: <http://www.c2.com/cgi/wiki?ExtremeProgrammingCorePractices>

Mapping Practices to Agile Principles

Facilitation Guide

Print all material (enlarge if you wish)

Hang the Agile Manifesto - Hang the 12 Principles on the wall - Hang the suggested list of Practices on the wall.

Have multiple colors of sticky notes & lots of pens/markers.

Discuss the Agile Manifesto - tell the history - describe what a process is and is not - ask if “Agile” is a process. Ask if it is a step of things to do to accomplish a task? Describe a philosophy - could it be that?

Switch to the Engineering Practices - describe a few of them - invite them to read the list, to circle the ones which they currently do very well. Invite them to add to the list - debate which practices are redundant (ex: code review & pair programming).

If the group currently has real disciplined practices - use them to map to the principles - however if not don't waste your time - just to the desired future state.

Dot-vote - each person pick three (2-5 is a nice range for this) engineering practices they wish to use in the future - the best practices to make us an Agile team.

Using those top voted practices - have people pair/triple up and one principle at a time decide if the engineering practice “supports” the principle - if so put the sticky on the paper - if weakly supporting - put the sticky below - if not at all no sticky. Same pair do each principle. Use different colored stickies for different practices to create a nice Info-graphic.

Be the example - select two participants and demo the first few principles for something like TDD.

Debrief: Which practice supports the most of the Agile philosophy. Does it work alone - if we just do that one practice - are we Agile? What is the “necessary and sufficient” set of practices for Agile?