



Agile Software Development: A gentle introduction



Computer science is a young science. Computer programmers my age were trained by engineers. That training dictated how we approached software development for an entire generation. But now after decades of building software to be expensive, unwanted, and unreliable we have come to realize software is different. Building software is more like creating a work of art, it requires creativity in design and ample craftsmanship to complete. Software remains malleable, often illogical, and incomplete forever. Agile software development is based on fundamental changes to what we considered essential to software development ten years ago.

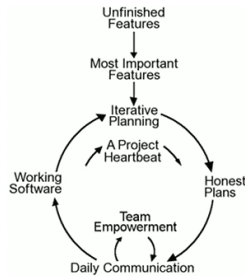
The most important thing to know about Agile methods or processes is that there is no such thing. There are only Agile teams. The processes we describe as Agile are environments for a team to learn how to be Agile.

We realize the way a team works together is far more important than any process. While a new process can easily improve team productivity by a fraction, enabling your team to work effectively as a cohesive unit can improve productivity by several times. Of course to be eligible for such a big improvement you must be working at a fraction of your potential now. Unfortunately, it isn't that uncommon.

The most brilliant programmers alive working competitively in an ego-rich environment can't get as much done as ordinary programmers working cooperatively as a self disciplined and self-organizing team. You need a process where team empowerment and collaboration thrive to reach your full potential.

The second change is making the software development is changing requirements. Agile processes accept the reality of change versus the hunt for complete, rigid specifications. There are domains where requirements can't change, but most projects have changing requirements. For most projects readily accepting changes can actually cost less than ensuring requirements will never change.

We can produce working software starting with the first week of development so why not show it to the customer? We can learn so much more about the project requirements in the context of a working system. The changes we get this way are usually the most important to implement.




Agile also means a fundamental change in how we manage our projects. If working software is what you will deliver then measure your progress by how much you have right now. We will change our management style to be based on getting working software done a little at a time. The documents we used to create as project milestones may still be useful, just not as a measure of progress.

Instead of managing our activities and waiting till the project ends for software, we will manage our requirements and demonstrate each new version to the customer. It is a hard change to make but it opens up new ways to develop software.

Take a guided tour of Agile Development by following the [Agile buttons](#) starting here. Or continue your guided tour of [Extreme Programming](#) by following the [Agile buttons](#). Let's look at how we manage by features next.

[Links and Books](#) | [Manage by Features](#) | [The Paradox of Process](#) | [Agile Modeling](#) | [Process Proverbs](#) | [About the Author](#)

Copyright 2009 Don Wells all rights reserved.



"St. Jude saved our son's life. And they're going to allow him to **live a life.**"

Meet Cole

