

Agile Software Development and Business Analysis

By Scott W. Ambler

Abstract:

In this article Scott W. Ambler, Practice Leader Agile Development in the IBM methods group, overviews agile software development and the implications for business analysts.

In February of 2001 a group of 17 methodologists met for a ski trip at Snowbird Utah, with the intention of discussing software process in the evening. An interesting thing about this group is that they all came from different backgrounds, and yet they were able to come to an agreement on issues that methodologists typically don't agree upon. They concluded that to succeed at software development you need to focus on people-oriented issues and follow development techniques that readily support change—Not exactly a new realization, but as you'll see the way that they've chosen to capture this conclusion has caught the information technology (IT) industry by storm. This same group later started the [Agile Software Development Alliance](#), often referred to simply as the Agile Alliance, after realizing they had started something that could prove to be of great importance to the industry.

The Agile Alliance's four values, referred to as their "[manifesto](#)", form the philosophical basis for agile software development. These values are:

1. **Individuals and interactions *over* processes and tools.** The most important factors that you need to consider are the people and how they work together because if you don't get that right the best tools and processes won't be of any use.
2. **Working software *over* comprehensive documentation.** The primary goal of software development is to create software, not documents — otherwise it would be called documentation development. Modeling and documentation have their place in agile software development, just visit www.agilemodeling.com to see that, but it is streamlined and made effective as possible. Properly written documentation is a valuable guide describing how and why a system is built and how to work with the system.
3. **Customer collaboration *over* contract negotiation.** Only your customer can tell you what they want. They likely do not have the skills to exactly specify the system, they likely won't get it right at first, and they'll likely change their minds. That's perfectly fine if you follow techniques that support this, not hinder this. Working together with your customers is hard, but that's the reality of the job. Having a contract with your customers is important, but, a contract isn't a substitute for communication.
4. **Responding to change *over* following a plan.** Change is a reality of software development, a reality that your software process must reflect. People change

their priorities for a variety of reasons, their understanding of the problem domain changes as they see your work, the business environment changes, as does technology. Although you need a project plan, it must be malleable and it can in fact be very simple (unlike many of the Gantt charts you may have seen in the past).

The important thing to understand is that while you should value the concepts on the right-hand side you should value the things on the left-hand side even more. A good way to think about the manifesto is that it defines preferences, not alternatives, encouraging a focus on certain areas but not eliminating others.

Although the four values are a good start, they clearly aren't sufficient to start an entire software process movement. In snowbird the Agile Alliance also defined an initial [collection of twelve principles](#), which they evolved in the following months, to help guide agilists. These principles are:

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 time scale.
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 and then tunes and adjusts its behavior accordingly.

In March of 2006 I ran an [agile adoption survey](#) for [Dr. Dobb's Journal](#) and it revealed several interesting statistics:

1. **A wide range of organizations are adopting agile.** The 4232 respondents worked for different sized organizations, from very small to Fortune 10, yet there was no statistical difference in adoption rate based on organization size.
2. **The majority of organizations are in the process of adoption agility approaches.** 41% of organizations had adopted at least one agile methodology (other surveys have shown an even higher adoption rate, although I suspect that those surveys weren't as broad as this one). More importantly, 65% of organizations had adopted one or more agile techniques. Because the first step in the adoption process is to try out a few techniques, I suspect that this is an indication that agile adoption is still increasing.

3. **Adoption agile is a low-risk decision.** Agile adoption appears to be very successful in practice. 60% of respondents indicated increases in productivity (34% report no change and 6% report a decrease), 66% report increases in quality (31% report no change and 3% report a decrease), and 58% of respondents report increases in business stakeholder satisfaction (39% report no change and 3% report a decrease).

So what is the implication for business analysts? First, you need to form a fair and accurate understanding of agility. This can be difficult considering all the rhetoric flying around out there, so my advice is to step back and think for yourself. Many people mistakenly focus solely on the left-hand side of the agile values and thereby form spectacularly incorrect conclusions about agility. Have you ever heard someone tell you that agilists don't model or write documentation? Reread value #2 again. Or they'll focus solely on the values and ignore the principles. Have you ever heard someone tell you that agilists do poor quality work? Reread principles number 1, 2, 9, 10, and 11. My experience is that agilists develop very high quality work, even compared to the work that I've seen on highly controlled CMM teams, and I suspect that within a few years the researchers will have gathered sufficient evidence to prove exactly that. However, I've also seen some non-agilists who claim to be agile do very poor quality work – the best rule of thumb that I can give you is that if they're not working very closely with their project stakeholders and taking a test-first approach to writing high-quality code then they're not agile. Also, don't let names like "Extreme Programming (XP)" and [Open Unified Process \(OpenUP\)](#) distract you. Although it sounds as if XP is something a bunch of marijuana-smoking snowboarders would do, and OpenUP something that you're forced to do during an off-site team building event, the fact is that both XP and OpenUP require incredible discipline. A more accurate name for these methods would have been "Highly Focused Programming Leading to High Quality Software on Time and on Budget (HFPLHQSTB)" but it just doesn't have the same cachet.

Second, recognize that business analysis activities are still important but will be performed in different ways within an agile environment. In traditional environments business analysts act as a bridge between the business community and developers, acting as translators and facilitators between those two groups. Agile project teams require you to:

1. **Act as a mentor, not a bridge.** This can be the hardest concept of all, but your goal should be to put yourself out of work. Many traditional developers struggle when it comes to interacting with stakeholders, thus the need for business analysts. Agile developers, on the other hand, actively strive to work closely with their stakeholders although may not yet have the skills to do so effectively. Agile business analysts facilitate effective, face-to-face interactions between developers and stakeholders, mentoring both in modeling and information gathering techniques.
2. **Focus on modeling, not documentation.** If you discover that you're writing documentation and then providing it to someone you're very likely working ineffectively. Agile business analysts use [inclusive modeling techniques](#) which use simple tools such as whiteboards and paper. Inclusive techniques can be easily learned by both stakeholders and developers, enabling both to communicate

- with one another effectively. Some documentation will always be needed, of course, just not as much as with traditional software development projects because agile projects have a much higher level of communication amongst the people involved. Consider creating [agile documents](#) and thereby reduce your documentation burden.
3. **Expand your skills.** Effective IT professionals seek to become what I call [generalizing specialists](#), people who have one or more specialties (such as business analysis, project management, programming, testing, ...), a general knowledge of software development, and hopefully a good understanding of the business domain in which they work. Just as you should transfer some of your business analysis skills to developers, you should also strive to pick up new skills from them. The more skills you have, the more employable you are.
 4. **Work in an evolutionary manner.** Modern software development, and in particular agile software development, works in an iterative and incremental (evolutionary) manner. We know that our stakeholders change their minds, that they often don't know what they want at first, that their business environment changes over time and thus motivates them to rethink their requirements. We can both deny these basic facts and try to put a change management process in place (really a change prevention process when you think about it) or we can embrace change and find ways to deal with it properly. Luckily the Agile Modeling method includes [practices](#) such as *Model in Small Increments*, *Iterate to Another Artifact*, and *Create Several Models in Parallel* which enable you to work in an evolutionary manner.

Agile software development methods such as Extreme Programming (XP) and the Open Unified Process (OpenUP) are real, they work, they're here to stay, and they are in your future if you're not actively working with them now. The good news is that they focus on collaboration between stakeholders and developers, something that is clearly a strength of any good business analyst. Are you prepared to leverage your skills and take your career to the next level by becoming agile?

Source material:

This material was modified from *The Object Primer 3rd Edition: Agile Model Driven Development with UML 2* by Scott W. Ambler (Cambridge University Press, 2004) and from *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process* (Wiley Publishing 2002).

Byline:

[Scott W. Ambler](#) is the Practice Leader Agile Development in IBM's Methods Group. He is the founder of the Agile Modeling (AM), Agile Data (AD), Agile Unified Process (AUP), and Enterprise Unified Process (EUP) methodologies. Scott is also the (co-) author of several books, including *Refactoring Databases*, *Agile Modeling*, *Agile Database Techniques*, *The Object Primer 3rd Edition*, and *The Enterprise Unified*

Process. Scott is a columnist with *Dr. Dobb's Journal* and an expert reviewer of the IIBA's Business Analysis Book of Knowledge (BABoK).