# Preserving the Spirit of Agile

Let's start at the beginning – why did your enterprise convert from Waterfall to Agile? Was it peer pressure, or a matter of keeping up with the Joneses, or a little of both? In our experience, many people have confessed they switched to Agile because it had the reputation of being the hot trend, and most companies were going that way anyway. However, if your organization genuinely saw the benefits of Agile, then you probably reasoned that your goal was to leave behind all the pain and suffering of Waterfall development. You can be fairly confident you were not alone if you have endured some of the following key challenges relating to Waterfall:

- Your organization spent months compiling requirements;
- Your staff played a seemingly endless telephone game refining those requirements;
- What your firm finally implemented is at least 35% off from what you really needed;
- Your requirements changed by the time you completed development anyway;
- Your organization was terrible at responding to changes in the market because of the above.

Those who created Agile intuited that there had to be a better way, and went about the business of dissecting these inherent problems within the Waterfall methodology. Agile methods have contributed to great improvements in productivity, solution deployment, and team collaboration.

However, what is required to be truly Agile is to accept and apply the fundamental tenets of Agile development. Let the spirit of Agile guide you during implementation. If you have already implemented it, but are still suffering from some of Waterfall's inflexibility, revisit your implementation. Keep it focused and keep it in accordance with why Agile was created in the first place:

**The Agile Manifesto for 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."

*Source: Wikipedia*

The strength of Agile is in its simplicity and purity, as articulated in the above declaration. Sadly, the above principles are often ignored. To understand this point, it would be helpful to walk through the Agile principles as they relate to process development. We'll leave it up to you to make your own honest assessment as to whether or not your organization is ignoring some of these tenets. If so, be bold and refactor your Agile implementation. Refactoring is one of the most important tenets of Agile, so refactoring your Agile implementation will preserve its integrity.

Our experience at many organizations that have tried to make the transition to Agile involves plenty of lip service to Agile, but without the real focus on Agile methodology to back it up. If you implement Agile and still hang on to old, lingering Waterfall concepts, you'll discover the worst of both worlds. Old habits die hard,

but it's crucial to make the commitment to the Agile tenets, or you won't reap the rewards.

First, we value **Individuals and interactions** over processes and tools. This is a big concept. You'll have a lot of vendors bombarding you with their wares in the process, technology, and tool categories. Right out of the blocks your organization will be asking, "How do we go Agile?" and you'll confront an army of tools vendors besieging you with their answers. Irony aside, why would you bring in a tool if the whole point is to make personal interactions a priority over tools, and why would you want someone pushing you toward a process when Agile stresses deemphasizing it?

We understand this principle can be a very touchy area to address in practice. As the Agile Manifesto states, there is value on the right, there is value to processes, and there is value to tools—nobody is saying these elements are worthless. But if your intention is truly to reap the benefits of Agile, start first with creating an atmosphere of human interaction. Use that as your guide, and err on the side of fewer tools as you start your transition. If you're already using Agile, ask yourself if your tool or process has become the work in and of itself. If so, you may have created the very situation Agile warns against, and will not be as productive as focusing more on **Individuals and interactions**.

Second, we value **Working software** over comprehensive documentation. This one is my personal favorite part of the manifesto, embracing a central point. What is the main benefit of Agile? It replaces those things that masquerade as work with actual work. It's a development methodology for developing. Period. Development is what Agile considers work. Meeting about work is not work. Taking notes about work is not work. Going over status about work is not work. Tracking your work in a tool is not work.

In that vein, **Working software** is an apt name. If you keep it as your focus, you'll be able to resist the following: compiling gigantic requirements documents; compiling gigantic technical specifications; and wasting valuable time by continually updating all your documentation as you go through the development process. Instead, you'll focus on doing the real work, which involves the actual development.

Third, fully engage in **Customer collaboration** over contract negotiation. This tenet may not be as wide-reaching as the others, as it pertains more directly to software vendors. However, if you extrapolate this tenet to include your internal customers and change contract negotiation to requirements negotiation, then it becomes useful for general development purposes.

The driver here is a bit more elusive, due to the reality that users can't get requirements right the first time—it's virtually impossible. Waterfall tried to remedy this issue through the process of getting sign-off from users. But, despite any rules governing sign-off, unmet requirements always remain. What Agile really strives to improve over Waterfall is the distance in time, money, and resources between receiving the requirement and its actual implementation. A belief in user transparency is what Agile promotes, and it's key to getting quick software views to customers, receiving rapid customer feedback, and making swift adaptations based on their responses.

What is the spirit of Agile when it comes to **Customer collaboration**? It's about building and showing customers the software, not dictating to them what the software will look like or how it will perform. It's about working so closely with the groups who are specifying the software that communication breakdowns get minimized.

Finally, what comes along with that? It's **Responding to change** over following a plan. As you collaborate with your customer on the software, what if your customer points out what you've built is different from what they had envisioned? Often, that will mean you built it correctly but it's still wrong, and doesn't reflect what

the software should be doing. That requires a change. It's vastly more important to respond to this issue rather than dig in your heels and stick to what was originally discussed, planned, and agreed to with the customer.

*Embrace change*—this tenet is the soul of Agile, and is arguably the biggest driver that allows Agile to be successful in your organization. The better you become at responding to change, the better your entire software development process will be. While it is possible to rely heavily on processes and tools and still be pretty successful, if you don't have the capacity for **Responding to change**, your organization will never be truly Agile.

---

It's well worth starting a dialogue to get your enterprise thinking about your own processes by highlighting some common mistakes organizations make in not sticking with the spirit of Agile. Let's discuss tools first, by evaluating a hypothetical project that's using AgiTrue as its Agile project management software. At many organizations, the project leader's relentless mantra on an Agile project is, "Did you update AgiTrue?" Project priorities have now shifted, and the process of updating AgiTrue threatens to become the focus of the work output, rather than the actual development itself.

Development does not have to be that way. If the focus is on **Individuals and interactions**, then the reliance on a tool, in this case AgiTrue, can be progressively minimized. The dreaded, "Is AgiTrue up to date?" will become a quaint refrain of the past. It's important to stress that there's nothing wrong with using AgiTrue, but if meaningful interactions are, in fact, occurring, dependence on this tool goes the way of the proverbial Waterfall dinosaur.

What about scrum planning meetings? If you've ever lived through one of those two-hour torture sessions, "Interaction" with "individuals" does not have to mean shoving everyone in a room together. It is painful and wastes everyone's valuable time. For the sake of everyone's sanity, it's time for a new rule. Find new ways to plan that are successful, but not unbearable. There are many creative ways to do this, but if the team leads and project managers are truly interacting on a regular basis, there should be no need for torture sessions. How you communicate with others is a great area to refactor as you adjust your behaviors to work more effectively.

I would be remiss if I didn't mention that sacred cow we all know and adore as 'documentation'. The perceived versus real value of documentation can induce panic in the hearts of many a poor soul. For some, this topic feels uncomfortably like an intervention, and let's be honest, we're all addicted to documentation to some degree. Relax—the first step in fixing this addiction is to admit it.

In our work at CBIG, we've encountered many organizations that persist in their unnecessary documentation habits even as they transition to Agile. A typical documentation obsession may go something like this: "Where is the technical spec? Where is the Source-to-Target mapping? Where is the business functional requirements doc? We can't even start without that!" We believe this occurs as a result of misplaced 'spinning in circles' momentum combined with a fear of all things new and different, specifically with regard to changing entrenched processes.

Let the truth set you free—you can actually develop working software without having every requirement documented, signed off on, reviewed by the team, agreed to by project management, budgeted, approved by finance, and signed off on by legal. Not only does it work, it is in fact a central tenet of Agile.

---

I'd like to finish with a point covered earlier, and a bit of some hard-earned advice. What you want to do when

you go Agile is focus on work. Don't focus on things that are like work. Don't focus on things that are quasi-work. Don't focus on those things that masquerade as work. Meetings can be work. Meetings can also certainly masquerade as work.

Many organizations are so dominated by meetings that there's often little time to do any actual work. Meetings can be successful interactions by individuals, but always keep your focus on refactoring how your projects are addressing meetings. Always keep refactoring how your organization views real work versus fake work. Listen very closely to those doing the work. Continually ask them what is getting in the way of their accomplishing real work.

Do you have a robust testing process with quality assurance analysts and testing cycles and their own tools? It's in your enterprise's best interest to make sure they're focused on producing working software. Make sure part of your developer's job does not become creating documentation for testers. Make sure there's plenty of interaction between testers of all kinds of groups, business users who define the requirements, business analysts who help drive development, developers who build the software, in short, all the appropriate stakeholders on the project. But also make sure that developing documentation for testers receives its appropriate prioritization, which is to say, very low if at all.

Do you have DBA or data architects managing data modeling for your project? Make sure their work stays within the realm of developing data models, not documenting them. Make sure their work is real work, developing working software rather than lingering in meetings discussing data models for hours on end. Make sure they're part of the development process right alongside the developers. Make sure they're interacting. Make sure the data modeling is not being done alone in a closet somewhere. Make sure the only interaction taking place is not a set of project resources reviewing a data model on paper. Let the working software prove out the data model. When the model needs to get refactored, refactor it.

Everyone on the project needs to buy into the spirit of Agile. Anyone who just goes through the motions of Agile is performing a disservice to their project and their organization. Keep the spirit of Agile as your guiding beacon at all times. When things look like they are slipping, reevaluate them against the Agile manifesto and ask yourself, WWAD? And then don't just sit there talking about it, do it.

*Author: Chris Ford, CBIG Consulting*

**For more information, call CBIG at (800) 334-2078 or visit our Web site at www.cbigconsulting.com.**