# Is "Agile" Distracting You?

by Jonathan Kohl

The term "agile" has become popular, and profitable. Early adopters of "agile" became in demand. Other groups jumped on the bandwagon, learning techniques such as Extreme Programming or Scrum, and adopting tools such as continuous integration, test-driven development, and refactoring. Still others wanted to mimic the success but didn't want to change the product, process, tool, or service they were offering. Instead, they discovered that a little "agile" whitewashing goes a long way. It's easy to prefix whatever tool, process, or service you're selling with the term "agile." The consumer is distracted from the real offering; it contains the right buzzword, it must be good. Consumers buy both the buzzword *and* the product or service.

The term "agile" has become abused and, since we don't have a standard dictionary definition, it is open to interpretation. (Agilists will tell you that being "Agile" is not the same as being "agile" in the dictionary sense.) That isn't to say that the Agile Manifesto that brought us the term "agile software development" was a bad thing. It wasn't. It was a wonderful reminder to return to a human-centered approach (we value people over process) and a product-centered approach (we believe in working software) that many had lost along the way. Just because something contains the word "agile" does not mean that the practitioners are not sincere, skilled, and working hard to create value for their customers and their teams. However, even when we have the best intentions, an ideal can distract us from our real goals.

"Agile" can distract from your message. I once started writing a book on agile testing and got stuck early on. My wife told me to look at Strunk and White's *The Elements of Style*, specifically the chapter on "Omit needless words." I realized my wife meant "agile" was a needless word. If I removed the agile prefix, I was left with content that was strong enough to stand on its own. It was also much easier just to write about exploratory testing or test planning without having to fit the ideas into an "agile" slant. I was really talking about applying testing ideas on agile projects, or being an agile-fluent tester. Many of the test ideas we've developed over the years plug right in to agile environments, or any other iterative, incremental lifecycle with just a bit of adaptation. It was a hard realization, but my writing (and particularly my thinking) profited from it. Now I just focus on testing in any context, and I pride myself on being able to test in all sorts of process environments, and my work is more meaningful.

> "I've seen far too many successful agile process implementations produce software that customers aren't interested in."

Agile can be a distraction from your mission to deliver software that your customers value, while supporting your team. I've seen far too many successful agile process implementations produce software that customers aren't interested in. I've also witnessed agile bullies deliver working software but grind up teams through their zealotry, bigotry, and elitism.

Agile can distract from your skill development. We can talk about agile ideals, but someone needs to code, test, document, market, and sell the software. One of my friends mentioned that for the past few years he was so worried about following agile practices that his programming skills suffered. He said he learned more in several months spending his time reviewing and practicing the programming techniques in *Structure and Interpretation of Computer Programs* [1] than he had in several years of trying to follow agile practices.

Take a moment to ask yourself these questions: *To what extent are agile practices helping you create value in your product and within your team? To what extent are they distracting you away from reaching your goals?*

I'm not against using the term "agile." However, since it has come to mean whatever the speaker wants it to, it has lost significance. Rather, tell us your project stories. Let's talk about what's really going on—what practices you are using, how they work for you, what you have tried, and what you have learned—agile or not. Tell us what your team learned when implementing the twelve practices of Extreme Programming, or how you successfully blended Scrum with existing practices on your project. I'd also like to hear about practices that don't get a lot of air time in agile circles: What are Cleanroom or RAD teams doing? How are they delivering successful software? What lessons can we learn from successful "waterfall" projects? What about that process that has no name but is working wonders with your team?

The Agile Manifesto was a welcome development in an industry that seemed mired in paperwork and process. Unfortunately, "agile" can be carried too far. I propose our rallying cry be not Agile, but *Value*—both to our customers and our teams. If we are delivering what our customers need, and we are building up our teams and those we interact with, does it matter what it is called? Let's stop worrying about whether what we do is "agile" or not, and go back to calling it software development. Let's worry about how we can do *that* to the best of our ability. **{end}**


References:

1] Abelson, Harold and Sussman, Gerald Jay. *Structure and Interpretation of Computer Programs.* The MIT Press, 1996.