# Why Agile works

By Armin K. Roth

The other day I found a company desperately looking for a "senior Rubyist", and they were describing the way they worked like this:

*"This is a fast-paced agile environment where code you write today will be live on our site tomorrow (Continuous Deployment FTW!). We need the best and the brightest to help us build better, more robust applications."*

This is about a process that has taken web-development in a hurricane-like way and destroyed many age-old traditions. Take for example the way code was written years ago (hey, and I remember the good old COBOL days, my friends!). No way we can afford to wait. Our times are not the ones that alow for half a year contemplating on a hill of thought, thinking like a philosopher about why a web-page exists, and what are the essential things to consider. We cannot allow the team to sit around and write specifications instead of code!

It is like the bicycle racer that was pushing his bike all the way and when asked by a bystander why he did not jump on it he answered: "no time!". Folks, this is the age of get stuff done the right way. And if it is not 100% right, we make it perfect the next sprint – but don't, ever, under no circumstances, let me wait!

So, the cool code you write today is in production tomorrow! Wow, we could now say: Isn't this is cool stuff other people are doing? But is it? The truth is, that everybody can actually get going this way. Just set out to try and you will see if it works. Of course, in order to live up to expectations and not to loose the connection with reality, there are checks and controls, but, oh, are they different from before!

Just a mere decade ago, you would sit in an office, and your project manager would call you in (as a matter of fact, the whole team!) to discuss the progress. You would then prepare powerpoints, with neat "progress bars", red, green and yellow lights on it just to make sure nobody realized that there wasn't any progress at all for a week. And they wouldn't find out until a week before the delivery was due and then a huge storm would break loose, resulting in a change request thrown at the client's desk, who then was in a dead-lock situation between available budget and available product... Everybody bogged down by having to write Word-documents and Powerpoints instead of gleaming, shining and shimmering  perfect code!

This is, what a progress bar should look like, folks:

(credit: Xing.com)

And, of course, it should be reserved for the weekend or very late nights.

The day should begin with a 15 minute chat with friends, in which everybody has a say on what she did yesterday (before going to that particular bar serving the progress stuff), what she will be putting her thoughts at today and if anything (apart from the hangover of last night) is stopping her to be creative. Then in the evening, if everything had worked out, she could put up a post-it note on the whiteboard and walk away in bliss.

In total, if the whole 4-7 heads team is doing it like her, they should be going at high velocity into the real world, delivering something visible, tangible and to be proud of in 4-8 weeks time that can actually be shoveled up into the cloud, on the server or on a mobile device and water the eyes of any client: They see that there is progress, instead of gawking at a powerpoint – and that, folks, makes all the difference.

Now, let's imagine, we have several of these sets of work (you may wish to call them sprints, as the velocity is like you were on a race track of creativity), you might see that everybody at any given moment of time can meet objectives and still keep some time to get acquainted with these new cool tricks (hey, I don't need to invent everything UI from scratch, I can use the twitter-bootstrap and look pretty professional!)

The funny fact is, that we are looking at a positive self-fulfilling prophecy (if you wonder about this thing, do yourself a favour and read Sir Karl Poppers book published 1976, "Unended Quest: An Intellectual Autobiography." LaSalle, Illinois: Open Court. ISBN 978-0-87548-343-6).

The more fun we experience in doing what we do, the better we get, and the more inspired we are about it. This is, what the all-time-guru of Agile, Jeff Sutherland, calls "hyperproductivity": The more satisfying the coding is, the more work we can get done and still have change in the pocket for creative procrastination. Seriously, and Agile development process allows for at least 10% of time for self-enlightning about new coding technologies, cool new languages (hey, have you heard about this amazing language for young start-ups, clojure?) and getting the newest technology in your brains.

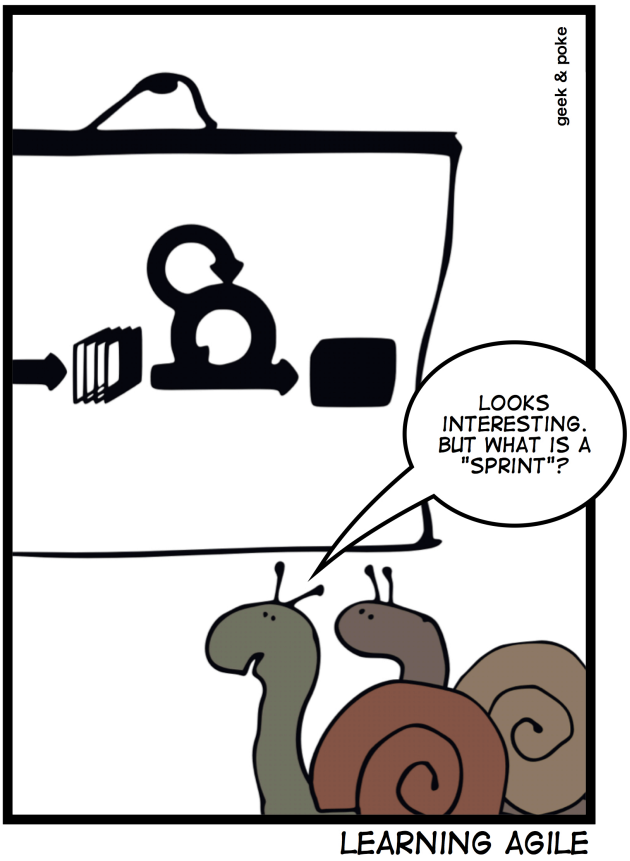
A while ago, I was participating in a coaching process to a large global company with its headquarters in Germany. Analyzing their development process, we found four versions that were cranked out every year, each of these being worked on for roughly 9 months time, so at least three of them in parallel. The coding for either version was not more than 2 months!

So more than half a year for each version was spent on getting it aligned with all sorts of perceived requirements. Every error ocurring caused even more gates and controls and procedures to pass. Yikes! Why not get stuff done in much less time and evolve from document-filling slaves to upright-walking achievers?

So why does Agile work? The reason is not in slavishly following procedures, or obeying a strict hierarchy. It is not in having long development cycles, in which the most productive and creative heads are working on "interdependence-management-documents" or turning a specification into an IT concept. It works, because the business owners and the developers sit at the same side of the table. Both share the same vision of what the product should look like, and both are okay with things that don't work out quite right in the first instance, as surely it can be remedied.

It is about getting stuff out to the clients as quick as ferasible. It is about getting the use cases and test scenarios while the ideas are still hot. It is about getting hooked into not doing anything twice or unnecessary. It is about reaching the ultimate level of efficiency, about not doing anything that doesn't serve the ultimate vision or the client's pipe-dreams, respectively.

This, my friends, is why Agile works.




(credits: geek & poke)