A Better Definition of

Agile Software Engineering

Gebhard Greiter, 2011

Never in the history of software engineering did a concept provoke more comments (without much progress) than what we call ***Agile Software Development***.

It is – without any doubt – true that the classical waterfall models for the software development process are too rigid for the fast moving world of today. However, I doubt whether the Agile Manifesto is the right answer to this problem. Let me explain this:

In the Agile Manifesto 5 statements and 12 principles speak about ***how*** to achieve the goal of delivering better software. However, ***what*** better software actually should be is left undefined resp. is mentioned in one sentence only:

> Our highest priority is to satisfy the customer
> through early and continuous delivery of valuable software.

Just to say that our goal is *early and continuous* delivery of *valuable* software is simply not enough: What exactly is valuable software?

In my point of view ***maintainability and quality under various aspects*** are needed to make software valuable. But can value in this sense really be achieved by following the rules set by the Agile Manifesto? I don't think so.

What I feel is: We need a better, more abstract definition of what it means to become agile – a definition focusing on the goals (what exactly defines valuable software) and then asking the community to start a search for better and better ways to produce value in this sense.

To approach the problem this way is important because the first way suggested – the way the Agile Manifesto sketches – is definitely not the best one: It is not stressing the fact that when searching for more agile ways to produce software we must not forget sound software engineering principles (especially not the principle of abstraction discussed so deeply in the years between 1975 and 1995 [but then only survived in programming languages]).

Another problem with the development process suggested by the Agile Manifesto is that this process may be useful for maintaining software but is certainly problematic where we need to develop software for a predefined fixed price in a quite specific time frame.

If, e.g. software is to be developed to support a specific event (e.g. the Olympic Games in a specific year soon to come) the definition of “valuable software” must certainly not be the same as it is when we have to develop a product that we hope will be on the market for a very long time.

So we see: The best way to achieve “valuable software” cannot be same in every project.

Another fact is that Agility in the view of a developer is certainly not the same as Agility in the view of an – always necessary – project manager.

This is why I suggest two definitions of Agile:

Software Developer's Definition of Agile:

Agility means to have a process in place that will allow us (and urge us)
to react on changing business requirements as soon as possible
— accept that the goal is a moving target —

versus

Project Manager's Definition of Agile:

Agile Project Management means to have a process in place
that is to maximize team efficiency

Accept only these (abstract) definitions of Agile, analyze the quite serious shortcomings of the development process suggested by the Agile Manifesto, and then find better and better ways to become agile in a way that really is progress and does not forget lessons learned in past: Let us ensure that one step forward (Agility) is not also one step backwards.

Author’s homepage: http://greiterweb.de/spw/

Latest version at: http://greiterweb.de/spw/dox/A_Better_Definition_of_Agile_Software_Engineering.pdf

By the way, I am not alone when I think we need a more abstract definition of Agile. In 2006 already John Rusk wrote:

“Agile development is hard to define, because most people define it by giving examples. For instance they give a specific description of Extreme Programming (XP), instead of defining agile development in general. We’ve ended up with a widespread misconception that agility is about XP techniques like Pair Programming and Test-Driven Development (TDD). …

“Defining agility by describing XP is like defining democracy by describing America. Such a “definition” obscures the underlying concept with details of the chosen example.”