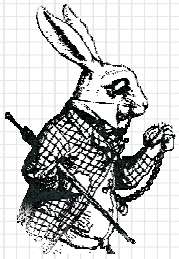

IT-DIRECTOR.COM

Agile development and change management: are they mutually exclusive?

Published: 18th July 2012

Further to my recent article about change management for business applications I want to discuss a similar issue with respect to agile development. First, let's be clear about what we mean by "agile development" on the one hand and "change management" on the other.

Agile development is all about speed, responsiveness to change and flexibility. In theory (not always achieved in practice of course) requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

Change management is all about ensuring that standardised methods and procedures are used to manage all the changes that occur to whatever it is you are building (a product or system) from the planning of that change through to its implementation. The objective is to ensure that no unnecessary changes are made, that any negative impacts are minimised, that resources are used efficiently, and to provide traceability.

So the question is: can you have the one with the other. One is fast and flexible and the other is perceived as not!

Every software project is a compromise: a balancing act between capability, speed and robustness. In an ideal world software would be delivered quickly, it would be function-rich, it would have been thoroughly tested and checked, and it would be provisioned appropriately. Unfortunately this is not an ideal world and we are forever looking for ways to get the product out to the client as quickly as possible.

But you shouldn't compromise functionality or quality simply for the sake of speed. And that's the potential danger with agile development. If not done properly by suitably qualified and competent people using understood and agreed control processes, you can easily compromise 'quality' in turn, causing more problems later on down the line, simply for the sake of speed right at this moment.

When you bear in mind that, according to *Capers Jones, Software quality 2011 - A Survey of the state of the art*, poor software quality costs $150+ billion per year in the US and over $500 billion worldwide then it is easy to see that this is not a minor issue.

So, how do you manage to have control and still be agile? I put it to you the other way around. How can you be responsive while simultaneously ensuring that solution evolution is going in the right direction and without compromising quality if you do not have effective change management procedures in place? Change Management does not have to be heavy weight and intrusive and nor should it be for an agile environment.

Finally, I would like to make one additional specific point with respect to agile development and version control. What versioning does is enable you to take keep track of your development. What it doesn't help you to do is either make design decisions or to manage collaboration. I mention this because SubVersion, the open source Apache project for version control appears to be wildly popular within the agile community. However, the question is whether it is enough? Change management vendors such as IntaSoft that have integrated their products with SubVersion would say no. Now you could argue that they have an axe to grind but I agree with them. Yes, you need to version your code, data models, test cases and so on but simply versioning these things imposes no control or discipline on the environment and if there is one thing that is important in an agile environment it is discipline. In a way, the real problem with SubVersion is that it is easy to think you have now imposed control on the environment when you have not and that can be extremely dangerous unless other processes, such as change management, are put in place around it: you can end up with a software project going nowhere very fast.

Philip Howard
Research Director - Data Management