Note of new methodology

Jiyang Chen 2013.10.30

1. methodology was introduced to make software predictable and efficient
2. Engineering methodology: planning inspired by other disciplines. Failed because so many stuffs need to do.
3. Agile methodology: just enough process. Code oriented. Adaptive and people oriented.
4. So the approach for software engineering methodologies looks like this: we want a predictable schedule that can use people with lower skills. It can look very good on paper, yet be seriously flawed when you actually have to program the thing.
5. In software the amount of time spent in coding is much, much less.
6. In building business software requirements changes are the norm. After all software is supposed to be soft. If you cannot get stable requirements you cannot get a predictable plan. Instead you need a process that can give you control over an unpredictability. That's what adaptivity is all about.
7. The key to iterative development is to frequently produce working versions of the final system that have a subset of the required features.
8. A usable, although minimal, system can go into production early on. The customer can then change its capabilities according to changes in the business, and also from learning from how the system is used in reality.
9. In an agile project there is a constant reworking of the plan with every iteration.
10. A project that's on-time and on-cost is considered to be a success. For agilists the question is business value - did the customer get software that's more valuable to them than the cost put into it.
11. If you expect all your developers to be plug-compatible programming units, you don't try to treat them as individuals. But for the highly creative and professional work, which I believe software development to be, this does not hold.
12. Ex-developers need to recognize that their technical skills will rapidly disappear and they need to trust and rely on current developers.
13. When measuring performance you have to get all the important factors under measurement. Anything that's missing has the inevitable result that the doers will alter what they do to produce the best measures, even if that clearly reduces the true effectiveness of what they do. This measurement dysfunction is the Achilles heel of measurement-based management.
14. Austin's conclusion is that you have to choose between measurement-base management and delegatory management (where the doers decide how to do the work). Measurement-based management is best suited to repetitive simple work, with low knowledge requirements and easily measured outputs - exactly the opposite of software development.
15. Technical people need guidance on the business needs. This leads to another important aspect of adaptive processes: they need very close contact with business expertise.So it's important to ensure that there is good quality business expertise that is both available to the developer and is of sufficient quality that the developer can trust them.
16. The first part of self-adaptivity is regular reviews of the process. Usually you do these with every iteration. At the end of each iteration, have a short meeting and ask yourself the following questions (culled from Norm Kerth)

What did we do well?

What have we learned?

What can we do better?

What puzzles us?

1. XP (Extreme Programming)

Extending their ideas of a software development approach that was both adaptive and people-oriented.

1. XP begins with five values (Communication, Feedback, Simplicity, Courage, and Respect). It then elaborates these into fourteen principles and again into twenty-four practices.