Introduction to Agile Software Development

https://github.com/christopherchristensen

3. April 2019

Inhaltsverzeichnis

Summary **2**
Notes of Peter Erne's talk 2
Sources 3
Questions **4**

Summary

Agile Software Development is known as an approach for developing software by putting the collaborative efforts of the team and all other stakeholders of a project in the forefront. It was largely popularized by the Manifesto for Agile Software Development (Wikipedia 2019) that was defined by a group of authors in the software development community back in 2001 (Beck 2001C). The authors main values are as follows and quoted:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

They claim that, *[…] while there is value in the items on the right, we value the items on the left more* (Beck 2001A). The Agile Manifesto follows 12 principles that should lay the foundation for anyone who is interested in implementing the approach. The following list shows these 12 principles (Beck 2019B):

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity–the art of maximizing the amount of work not done–is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

It is important to note that these are principles and do not tell you how these principles must be implemented to work. This leaves room for interpretation and for adaptation to the individual use cases.

Notes of Peter Erne's talk

- The Art of SwissQ consists of Agile - Requirements - Testing
- Before and why Agile
  - Different kind of products
  - 1890 - 1925: Proprietary Capitalism (Early Consumer)
    - Higher costs in productions very high for additional requirements / goods
  - 1915 - 2005: Managerial Capitalism
    - Decaying costs for additional goods
  - 2005 - present: Distributed Capitalism

    - Faster innovation cycles
  - Value doesn't follow organizational silos
  - Management challenge: create value across silos
  - silos offer good communication in the individual silos but not across them
  - good for production of physical products
- Why Agile?
  - Ability to cope with changing priorities
  - Speed up Time-to-market
  - Better collaboration of Business and IT
  - Increase productivity
  - Simplify development processes
  - Minimise risks
  - Enhance team morale
  - Increase transparency
  - Reduce costs
- Classical alternative models
  - Waterfalls
    - divided into phases
    - plan
    - concept
    - dev
    - testing & QA
    - customer
  - Customer never involved (too much room for interpretation)
  - In the beginning only paper or small prototypes are created
- Do agile approaches cost more?
  - measure against high costs is automation (pipelines, tests, etc.)
  - the risk of delivering a wrong product through less feedbacks (i.e. as in waterfall approaches) can increase the costs
- Specialists of different teams join up and form guilds (communities of practice)
- Tools & Processes don't make a company agile, it is the mindset, the values and principles around the tools that are more powerful
- the mindset and values are less visible but are more powerful
- the tools & processes are more visible but less powerful
- shifting these mindsets and values can take a long time (up to 5 years!)
- the team and their performance should be the focus when it comes to giving bonuses to individual employees
- with agile approaches many middle managers (line managers) lose their "hirarchical position" and the migration can therefore be very difficult

Sources

- Wikipedia, Agile Software Development, 2019 wikipedia.org
- Kent Beck et al., Manifesto for Agile Software Development, 2001, agilemanifesto.org
- Kent Beck et al., Principles behind the Agile Manifesto, 2001, agilemanifesto.org
- Kent Beck et al., Authors: The Agile Manifesto, 2001, agilemanifesto.org

Questions

1. For newcomer developers like me who has barely scraped the surface of the different types of approaches, it often seems like Agile Software Development is praised as the ultimate approach. In what use cases would you **not** recommend this approach?
    - Deterministic project / problem, where you can make a plan in advance (stable requirements, well-known technology)
    - Exploratory projects (meeting requirements, where customers might not even know the requirements exactly or doesn't know the requirements until they see it)
2. Are there any big functioning software companies that use entirely different approaches?
3. Do you get customers that constantly reject results of iterations due to the unfinished status?
    - Banana principle (Microsoft)
    - Often different versions for the same part of software (netflix, etc.)
        - Only works if you have a good craftmanship
4. Does Agile Development ever suffer under the "lazy" one?
5. If working software is the primary measure of progress, isn't there an incentive to "fake" working software?
    - working software that adds value for the customer
    - it also depends on **how** working software is measured
    - it does exist
    - its important that everyone adds value also for the team performance. there is an incentive but it important that the measurements are set correctly to eliminate these incentives.
6. Are there any difficulties in adapting this approach?
    - easy to understand but difficult to implement
    - many real world components are missing (no finance, hr, workplaces, sourcing, supplying) → condensed
    - it can be very misleading, because of its simplicity
    - you have to add these missing components
    - often scrum master becomes team leader which is not optimal
    - responsibilities must be divided smartly or only separated disciplinary responsibilities
    - with agile approaches many middle managers (line managers) lose their "hirarchical position" and the migration can therefore be very difficult
    - It is difficult to find good product owners that can decide on things and can communicate with all the stakeholders

```
burtzoorg
```