Agile

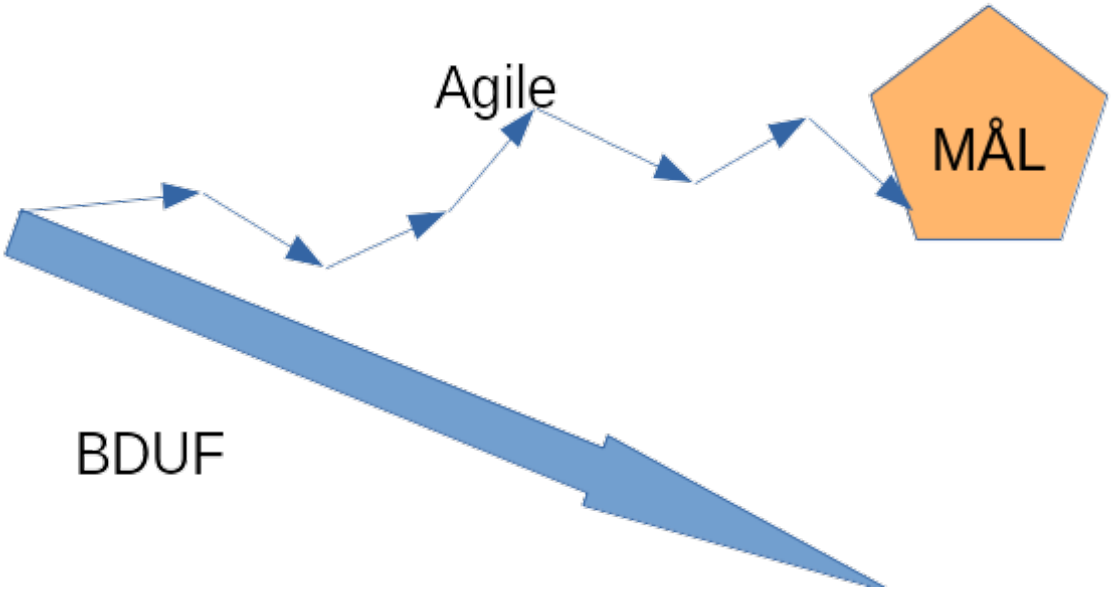
September 17, 2024

1 Agile methodologies

The agile programming methodologies (Extreme programming, SCRUM, etc.) come out of the turn of the century, when it became obvious that more than half of the large software projects in the world were total failures. *Ie. they failed to deliver any end user value.*

The ideas that people had for breaking the vicious circle were:

- Skip Big Design Up Front
  - You can't know everything about requirements from the start
  - Work in short iterative cycles
  - Deliver functionality as it becomes ready
- Work continously with your users
  - The user makes priorities between different features
  - The sooner they see the functionality, the sooner we can correct our course
  - Small corrections in each cycle
- By building trust, we can get all parties to work towards a common goal
  - BDUF requires very detaild contracts and course corrections are hard
  - In agile course corrections are part of the process
- Cross train your team
  - When the user is in charge, everyone needs to know a large part of the codebase
  - You get resilience in your team, sick people and people leaving don't derail things
  - You get trust within your team
  - Everyone gains new skills



The short cycles usually contain:

1. Review of backlog
2. Estimation
3. Selection of goals for the cycle
4. Preliminary division of work
5. Implementation, possibly with spikes and sprints
   - A spike is a short session to develop a user story
   - A sprint is a focus period when all other business is set aside
   - (A sprint in Scrum is different, others call that an iteration)
6. Review of goals
   - Check off accomplished ones
   - Send others back to the backlog
   - Anything we discovered for the backlog?
7. Retrospective
   - What went well, what not so well, what do we want to change

Cross training means a number of things in practice:

- Pair programming
- Code reviews
  - Keep code reviews short, it should maximize learning
- Let people discuss design and process
  - By being involved, you learn
  - Even a blind hen picks corn

1.0.1 Pair programming

Ensure undisturbed environment to get into flow. - No emails - No phones - No colleagues dropping by

Master-apprentice

- Apprentice at the keyboard
- Master needs to have a rough plan for what is to be implemented
- Master gives directions as necessary
- Apprentice can ask questions and make comments to verify understanding
- Master should test boundaries by making directions more and more general
- Very fast knowledge transfer
- Done in short sessions - no more than 2 hours at a time
- Master and apprentice can switch roles if moving between knowledge domains

Journeyman-Journeyman

- Peer programming
- Switch at keyboard
- Discuss implementation
- Back seat driver notices mistakes and thinks about the bigger picture
- Builds mutual understanding and trust

1.1 Mob programming

Also called **ensamble programming** is a mode of working where you bring many people in front of a computer and develop code. I haven't tried it myself, but I can see occasions where there is a point to it.

- When you want to get multiple people on the same page
- When you want to teach something to a larger group
- When you have a problem that requires multiple fields of expertise

[ ]: