Table of contents (../toc.ipynb)

# Workflows and beyond

This final theory notebook will be a short summary of different workflows and software development processes. The goal is to get a high level overview of the methods.

Add to this, two methods for collaborative software development are outlined.

# Keywords

You probably heard about keywords and concepts like

- Inner Source,
- Open Source,
- Waterfall,
- Agile?

The first two keywords are about collaboration models and the latter about project management (or its absence).

Let us take a short look at Open and Inner Source first.

# Collaboration

## Open Source

(https://de.wikipedia.org/wiki/Open_Source)


Nowadays, there is no need to tell much about open source (https://en.wikipedia.org/wiki/Open_source) because it is virtually everywhere and very successful. Especially if you are using Python, you build entirely on open source software.

The main strengths of open source are:

- reuse of software,
- often better quality than proprietary software,
- you can propose extensions,
- more trustworthy because you can read the code,
- close to customer, you can change the part of software you want,
- ...

This was just a short list of arguments.

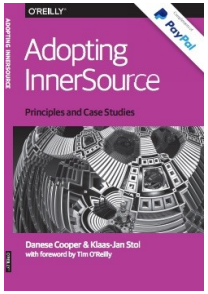
# Inner Source

(http://innersourcecommons.org/)

Inner Source (https://en.wikipedia.org/wiki/Inner_source) is a rather recent initiative to apply open source principles inside companies. Larger companies face basically the same challenges that lead to open source culture. These challenges are:

- Software development is highly distributed in space (countries, departments) and time (time locations).
- Redundant software development is waste of money, reuse of existing software makes more sense.
- It is hard to shape high quality software culture without sharing code.
- Organizational boundaries slow down development and do not contribute to quality.

(http://innersourcecommons.org/assets/img/AdoptingInnerSource.jpg)



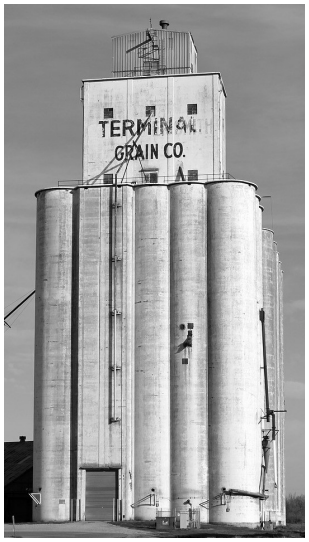

Add to these challenges, inner source benefits are:

- Flexible utilization of developers (you do not need to change the department to fix a bug in a project).
- Higher motivated developers.

A great source of company experience with inner source is [Cooper2018] (../references.bib).

# Silo thinking

Last but not least, probably the majority of software within companies is still written in silos. The common problems with silo thinking in general are:




- It is hard to find someone who is in charge.
- Much redundant is work is done.
- The quality is usually poorer than in shared projects because no one else can see the mess behind.
- Individual software parts within a company are not compatible because there is not need to work together.
- Specification, negotiation, internal contracting wastes many resources.
- Blaming more common than solving problems.
- ...

# Development paradigms

## Waterfall model

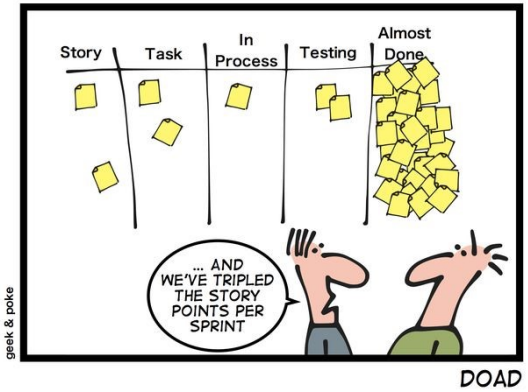
The [waterfall model (https://en.wikipedia.org/wiki/Waterfall_model)](https://en.wikipedia.org/wiki/Waterfall_model) is the oldest project management principle and is seen as "not so hot today".

The waterfall model is a sequential development where **product requirements** are converted into a **software design**, the **implementation** is made within this design, the software is **verified**, and **maintained**.

Main criticism is with respect to sequential development. **Usually, neither costumers nor product managers know all requirements at beginning** and in practice the **requirements change over time**. With waterfall, you need to wait until end of verification to change the requirements and start from scratch, which is very time consuming.

(http://geek-and-poke.com/geekandpoke/2012/10/1/doad.html)

## Agile




Compared with waterfall, the term Agile (https://en.wikipedia.org/wiki/Agile_software_development) is very hot and a lot of consulting agencies make money out of this fact.

Agile originates from the agile manifesto (https://agilemanifesto.org/) which makes the following suggestions:

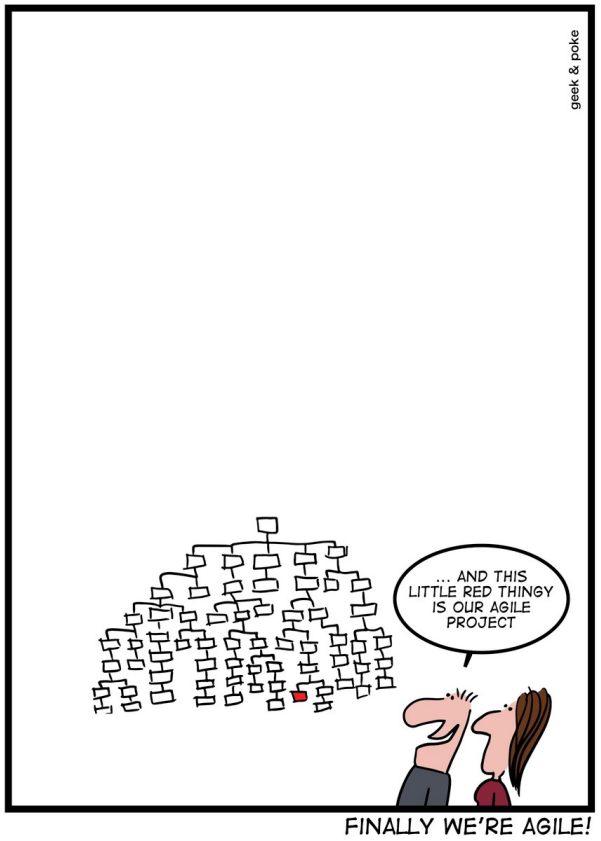
- **Individuals and Interactions** over processes and tools
- **Working Software** over comprehensive documentation
- **Customer Collaboration** over contract negotiation
- **Responding to Change** over following a plan

(http://geek-and-poke.com/geekandpoke/2016/4/26/finally-agile)

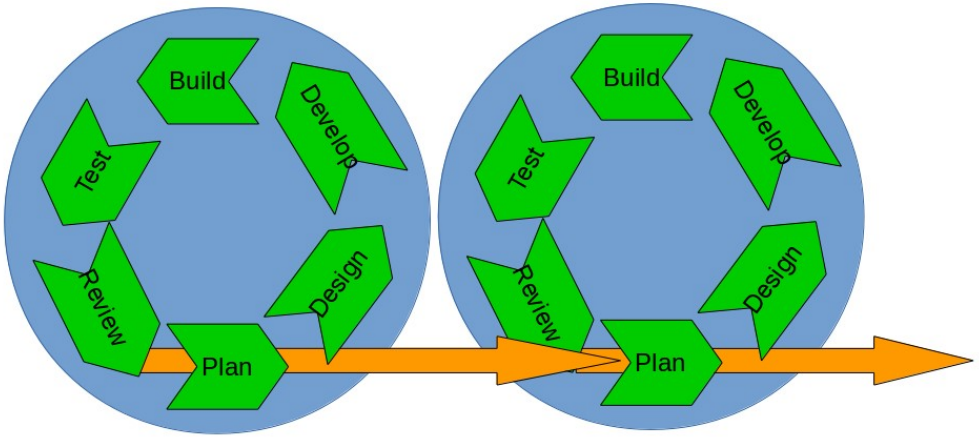
Therefore from its roots, **agile is quite the opposite to what large organizations are used to work with**.

This is why a lot of consulting agencies bend agile into a form which is convenient for large companies so that they can say they are working agile but stick to their old culture.

Agile is often used in combination or as substitute for terms like **Scrum** and **Kanban**, which complicates things further. Instead of a lengthly explanation what Agile is, I prefer to provide a very short explanation and a warning and links to expert resources.



# Agile in a nutshell




- Agile is iterative development.
- Working software is delivered frequently (weeks no months).
- Simplicity is what counts! MVP minimum viable product (bare minimum for the client).
- Regular meetings: daily standup (what did I, what I am working on, problems), sprint planning (slice features into stories, estimate effort in story points), sprint review (team presents what is delivered), retrospectives (what went wrong or well, continuous improvement).

### Sad story: some experts recommend to abandon Agile

Experts who developed the agile manifesto actually recommend developers to abandon agile (https://ronjeffries.com/articles/018-01ff/abandon-1/) because the process people flood all agile conferences with a very process driven view on agile.

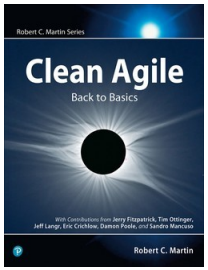
As result, many agile implementations in large companies are process driven and already the first principle "**Individuals and Interactions** over processes and tools" is not realized. This process driven implementations are known as dark scrum (https://ronjeffries.com/articles/016-09ff/defense/).

More serious warnings are stated in these conference video casts:

- The death of Agile - Allen Holub (https://www.youtube.com/watch?v=vSnCeJEka_s)
- GOTO 2015 • Agile is Dead • Pragmatic Dave Thomas (https://www.youtube.com/watch?v=a-BOSpxYJ9M)
- Agile in 2018 (https://www.youtube.com/watch?v=G_y2pNj0zZg)

## Books on Agile development

(https://www.oreilly.com/library/view/clean-agile-back/9780135782002/)




As there are so many authors and consulting agencies make money out of the agile hype, it is not so easy to find the original concept behind agile. Here are two trustworthy sources from my point of view.

- Agile Software Guide blog on martinfowler.com (https://martinfowler.com/agile.html)
- [Martin2019] (../references.bib)

# Congrats

These were the basics in software development I wanted to share. The topic is much larger, but with the given outline you will be able to understand and implement more advanced concepts.

The final part of this course is a collection of mini projects.