

Development Workflows

How a list of goals get worked on

- Past: Waterfall
- Recent: Agile

Waterfall model

Each step done in turn before the next starts

- If a hole is found in a previous step, back up
- Common key: All requirements in advance
- Fantastic for "known" tasks
- Great for hard requirements:
 - features
 - resources (incl people)
 - deadlines
- Generally terrible for most software
 - slow
 - inaccurate

Agile development

<https://agilemanifesto.org/>

Often struggle to actually adopt

- Companies want hard accuracy (it is a lie)

Common ideas:

- produce working code very frequently (no guess)
- cannot "make up", if behind, change one of:
 - resources
 - features
 - deadlines

Sprints

Two common styles:

- "Sprints"
 - 1wk/2wk/1month runs
 - work assigned at start
 - expected to be done at end
- "Kanban"
 - constant flow of work
 - each step should be quick (~1-2 days)

Both involve small, defined, concrete tasks

Tasks / Stories

- something worth tracking if done
 - depends on company
- has some meaningful result
- not too large (2 days or less)
 - break up bigger tasks

Estimation

We are terrible at estimating software dev times

Minimize impacts

- Small tasks
- Regular check-ins to confirm status
- Keep repo in usable state

Establish "velocity"

- Consistently bad estimation

Agile or Waterfall

- Both act like coders are fungible (we aren't)

Code Review

- NOT judgment of you
- Ideally not a rubber-stamp
- chance to improve
- consider longer-term impacts
- consider wider issues
 - consistency
 - compatibility
- should be done promptly
 - don't block others

Standups

Intended to be a fast meeting

- Identify problems
- Share status

Dos and Don'ts:

- Don't feel you need to justify your time
- Don't slow down standup without reason
- Do listen for things to learn
- Do listen for issues others might not see
- Do identify risks and problems