Making your Project Successful and Less Stressfull

Thomas Neumann

Technische Universität München

April 23, 2016

Collaborative Coding

In your research you are probably building some piece of software

- other people (undergraduates, collegues etc.) might work on it, too
- often a cause of problems
- ideas differ, sometimes people derail it, etc.
- driven by ideas, requirements, papers
- many ideas are experimental, unclear if they will work
- have to find a balance between code quality and freedom
- ego can be a problem, too

Disclaimer: The approach I propose worked for us, not necessarily for everybody. But perhaps it can give ideas.

The Actors

Everybody is different, but here are some common perspectives

- **undergraduate students**
  - enthusiastic, but often inexperienced
  - code often needs some polishing to improve quality
  - might leave after some time
- **PhD students**
  - stay for much longer
  - know the system better, more experienced programmers
  - sometimes tempted to hack something together for a paper, though
- **you**
  - should have the long term success of the project as goal
  - have to keep the code quality up
  - make sure that boring but missing features are implemented

Not always easy. Also some people are more altruistic than others.

The Solution

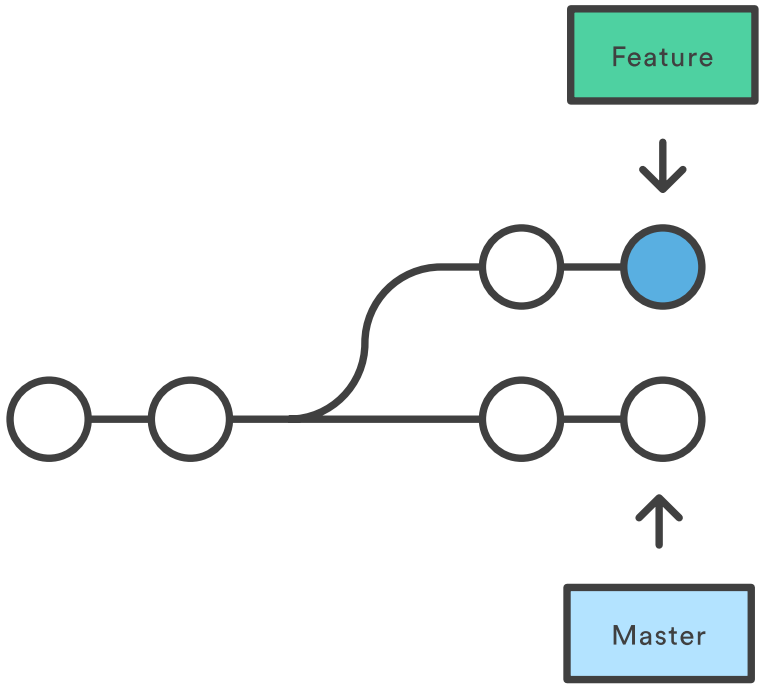
Use git, do not give anybody write access

The Solution

Use git, do not give anybody write access

- sounds a bit extreme, but is a good idea
- a key requirement for code quality is code review
- after the fact reviewing never works
- it must be physically impossible to write without reviewing
- but reviewing must not hinder productivity
- a DVCS with pull requests is a good system for that

Distributed Version Control



- in a DVCS like git everybody has its own repository
- the “mainline” only exists by convention, everybody is one a branch
- on the branch everybody can experiment with new features
- but mainline merge happens only after pull request + code review
- the most important task of the project coordinator
- can also happen hierarchical (undergraduates $\rightarrow$ PhD students)

Code Review

An essential step for long-term code quality, must be taken very serious

- a student pushes its own branch to some public place, sends a pull request
- you carefully go over the code together with the student
- give hints, check for logic errors, point out style issues, etc.
- usually (!) the initial pull request will be refined, perhaps multiple times
- quite time consuming, also for the reviewer, but very important

Everybody knowns: If I want my feature to get into mainline, I have to write high-quality code.

Benefits

- everybody can experiment on his branch just fine, not limit in productivity
- but mainline code is carefully reviewed
- mainline is always in good shape
- inexperienced programmers do not disrupt the code, learn from reviews
- prevents half-baked code (as in: "a hack for this paper")
- creates pressure to implement boring stuff

A lot of work for the reviewer, but essential for long term project success.

You should build systems, too!

I would like to encourage you to build some kind of system:

- much more work than just a micro-benchmark for a paper, of course
- but you should aim for the long term benefit
- incredible useful to have a running system
- you can try out new features easily
- experiments are much more meaningful in complete systems
- great for teaching PhD students

And use code reviews!