Why academic software should be open source

Sébastien Paumier

FS/OSS

- two terms:
  - *free software* (ambiguous)
  - *open source*
- same principles, but different underlying ideas:
  - free software: generosity with mankind (knowledge belongs to all)
  - open source: efficiency of cooperative work (the one that knows is the one that does)

4 freedoms

- quoting Richard Stallman, father of the Free Software Foundation:
  - freedom to use software, no matter the usage
  - freedom to study and modify it
  - freedom to redistribute it
  - freedom to improve it and to distribute your modifications

Academic software

- we talk here about scientists working with public funding: academics, national research centers, ...
- we talk about all kind of software they may produce:
  - proof of concept
  - industrial product
  - internal tool
  - etc.

Compatibility with industry

- no incompatibility:
  - licenses as LGPL do allow integrating OSS code in proprietary software
- industrial development differs from scientific knowledge production:
  - design, ergonomy, maintenability, support of many data formats, plugins, interaction with other tools, etc.

An OSS has no unique master

- with OSS, you do not depends on any group or any person
- if you don't like a new policy, you can always prefer the old one or create your own custom branch

- example: symbolic calculus software changing the operator + into U

Bugs and patches

- in case of bug or missing feature, with OSS, non-programmers can ask someone to patch the code for them:
  - no awful tricks in Perl/Python (i.e. splitting a big file into small ones because of a small-sized constant)
  - no painful fights for neewbies with terrible things like UTF8 or library dependancies

One may not be a coder

- many software are written by non professional coders:
  - those codes are hard to maintain and to integrate into other software
- OSS can be studied and improved by skilled people:
  - code optimization by industrial coders
  - spellchecking of English comments by native speakers
  - etc.

Fearing being anonymous

- some people fear being anonymous because of OSS, thus being not cited in papers, because of the cooperative aspect
- OSS protects authors:
  - contributions can be signed by comments in the code
  - anonymous modification of someone else's code is not permitted

Need to control

- some people want to control the evolution of their creation, but everyone can modify and redistribute OSS
- solution: refering to the "official" version by a name, that can be protected

- example: Ubuntu and Debian are Linux distributions identified by their names

Compatibility

- most non free software are distributed as binaries that depend on a specific system, or even on a specific machine
- how many labs do own a computer only used to run version 2.7.1 of FooCalculator ?

- OSS can be adapted et compiled everywhere

Peer-reviewing

- "*Empirism is not a matter of faith*"

  Ted Pedersen

- how can one seriously review a paper using software X if X is not accessible ?
- violation of the experiment reproducibility principle

Room for improvement

- if software is non free, how can one know what can be improved ?
  - theoretical improvements: algorithm with a better complexity
  - coding improvements: a smart-sized constant may speed up the program, but how could you know ?

Room for improvement

- with non free software, to test an idea, you have to:
  - recode all, adding your modification
  - compare
  - pray for the existence of a statistically significant experimental difference:
    - yes: you win, but the original software was maybe just not optimized enough…
    - no: you lost *X* months, because no one likes to publish negative results

Changing is hard

- people don't like changes:
  - they won't use your new software, unless there is a *killer feature*
- instead of recoding the whole thing, maybe in vain, your modifications will benefit of being integrated into a software that is already used:
  - OSS replaces time wasting concurrency by constructive cooperation

Software birth and death

- many software are written by PhD students, and die when they leave the research world:
  - how many software people have written papers about in the last 10 years are still accessible and used today ?
- OSS can survive to their authors
- you can avoid programs that only the author can use

Secret is a fantasy

- there is no program that cannot be rewritten
  - secret is only a temporary protection
- if a protected program is really interesting, it will be rewritten sooner or later
- protection encourages concurrency, but:
  - when there is OSS concurrent, will you win against a motivated community ?

Academic prestige

- if you release your work as OSS, you allow others to contribute to it
- they can complete your work, which makes it more valuable

- from a "publish or perish" point of view, inheritance is much more profitable than concurrency

Speed of science

- if your work saves people's time from reinventing wheel, they can focus on real new things

- science toolbox grows faster

Quality of science

- if everyone provides free components in its own expertise domain, you avoid to implement bad answers to problems that are well-known by people of the domain

- improvements can come from everywhere:
  - code optimization, new features, documentation, debugging, etc

Many good reasons

- science shares many ideas with OSS:
  - knowledge sharing, cooperative work, transparency, ...
- OSS is constructive way of working that helps science to develop better and faster

Conclusion

- public-funded research should only produce OSS
- you should never publish or let publish papers about non free software

References (papers available online):

Why academic software should be open source, Sébastien Paumier

Empirism is not a matter of faith, Ted Pedersen