Article: Ten Simple Rules for the Open Development of Scientific Software

Brynjar Smári Bjarnason

KTH – School of Computer Science and Communication

link to article

March 11, 2013

Introduction

Open source software

- Among the most cited publication
- High impact
- Wide user base

Pojects such as

- Galaxy, Bio[Python |Perl |. . .], EMBOSS, MetaVelvet, Taverna, Python, R

Ten rules

1. Don't reinvent the wheel
2. Code well
3. Be your own user
4. Be transparent
5. Be simple
6. Don't be a perfectionist
7. Nurture and grow your community
8. Promote your project
9. Find sponsors
10. Science counts

Don't reinvent the wheel

- Many algorithms and methods already implemented
  - Check if your problem has been solved
  - Break down your problem and check if parts have been solved
- If an open source project could be fitted to your problem, it might be a good idea to collaborate rather than re-implement.
- You can estimate if collaboration is appropriate by evaluating the source

Code well

- Know the basics of software development
- Read code
- Improve open source code and send patches

Code well

- Know the basics of software development
- Read code
- Improve open source code and send patches
- READ CODE
- You learn a lot of tips and tricks by reading code, BioPython nice example

Be your own user

- Show your software can answer important and relevant questions in your field
- Should be simple for developers to use and integrate in their own research
- By doing this during development you avoid ending up with software that doesn't even meet your requirements

Be transparent

- Fear of getting scooped and bugs in pre-releases are valid concerns
- However being transparent can lead to
  - Founding or contribution to stake claim. Easier than redoing the whole project
  - Pre-release users know the code is not complete. Should review the code to make sure it does what they want. Leads to more eyes searching for bugs
- Therefor, better final product.
- Probably best known ways to be transparent: GitHub and SourceForge

Be simple

- Science is HARD! :)

Be simple

- Science is HARD! :)
- Using complex software for complex science is HARD!

Be simple

- Science is HARD! :)
- Using complex software for complex science is HARD!
- Make sure your software is as easy to set up as possible
- Make sure your software is as portable as possible
- Use standard way of packaging software.
- Stick to standard file formats.
- Everybody loves standards!
- Software which is simple to install is more likely to be tried out
- Documentation, sample code, sample data, test cases, video demonstrations. Make it simple, both for your potential users and for your sake

Don't be a perfectionist

- Release early, release often (attributed to Linus)
- Release when new features done. Users quickly identify bugs and request new features.
- It's better to have software now that solves some of your problems than to wait months for software that solves some more of your problems
- Think about using Agile development (SCRUM) for your work
- Trello is very nice online Agile board

Nurture and grow your community

- Give credit to the open source tools you use for your research
- Give credit for contributions to your open source software
- Get others involved in acting on feedback and contributions
- Try not to make changes to key aspects of your code such as APIs, file formats or command line options, people get annoyed

Promote your project

- If you want users, you need to advertise
  - Name your project, stick with it
  - Well organized, simple web page
  - Promote where potential users might lurk (LinkedIn ResearchGate)
- Conferences are important, give presentations
- Ad-hoc?? meet-ups, hackathrons

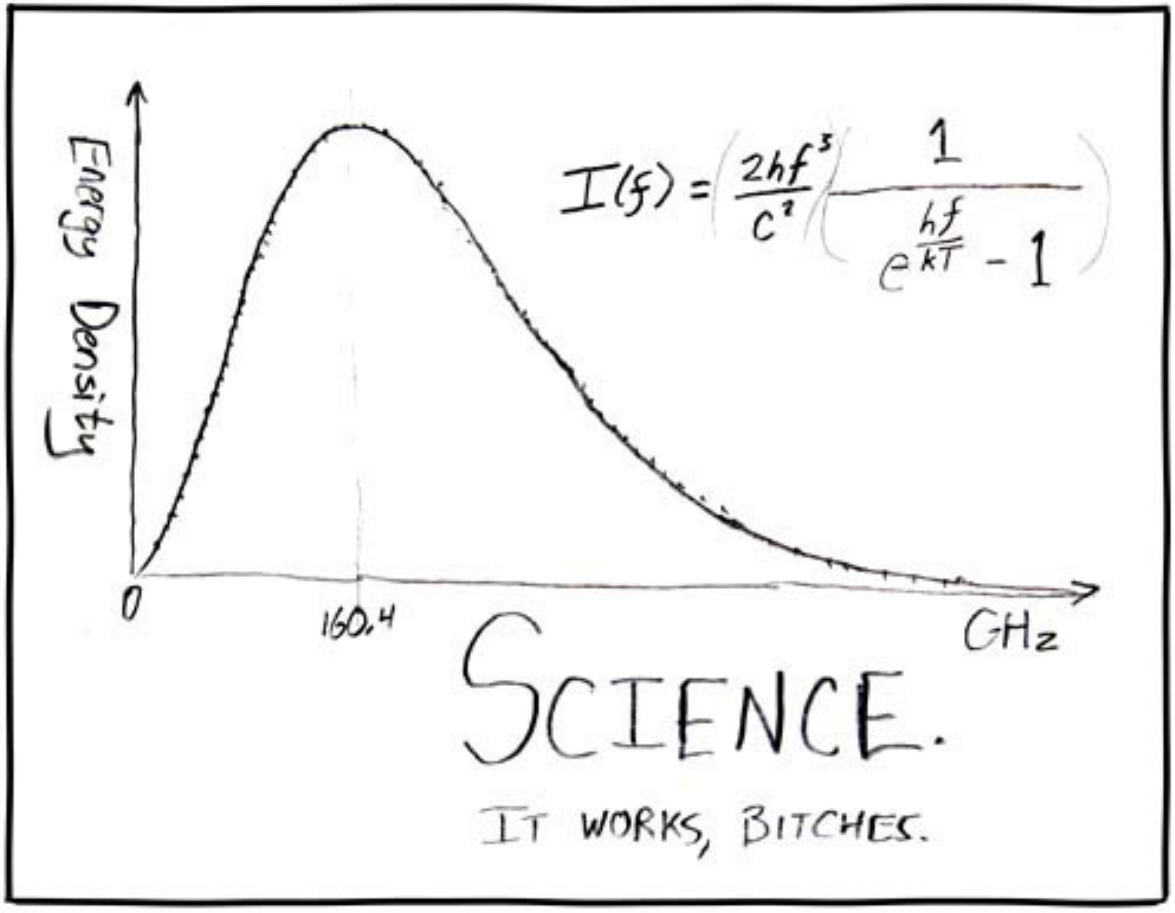
Find sponsors

- We can't work for free (for too long)
- Open development directly addresses the sustainability clause in many grant applications
- Open source not enough though, the community around the project needs to be the focus for long term success
- Any input on other ways to get sponsors??

Science counts

- Software to achieve scientific goals
- Becomes time sink when research over
- If done right, other people interested in furthering your process can take over
- Allows you to attack new frontiers knowing your software is in good hands

Science. It works



XKCD