389COM: Becoming an Open Source Developer

Dr Carey Pridgeon, DR Nazaraf Shah

Coventry University
http://www.coventry.ac.uk

2016-08-06 Tue

Benefits

- Jobs all need experience.
- You can't get experience without a job.
- A Good Open Source Product can give you that same experience.
- You get to control *everything* in your product
- You can learn new technologies/practices without an employers deadlines.
- It doesn't matter if you fail.

Managing Expectations

- Making money from your project *cannot* be you motivation
- Making money through the experience you gain can be.
- Your project (if it's a lone project) isn't likely to be a worldwide success.
- Having a few people appreciate your work can feel very good.

Starting your own Project

- Identify an interesting subject that will teach you something new.
- Choose an online repository (don't host it yourself).
- Write a User Story (why is this product interesting, even if only to you).
- Pick a licence and start. (getting this aspect right is *really* important.

Joining another Project

- Identify an existing project that interests you.
- What value can you add to that project?
- Write something and submit it to the project owner.

Develop Good Practices

- The moment you apply for a programming job, someone will go look at your code.
- Write code, then write a test for that code, and document it *as you go along*.
- Use Continuous Integration from day one.

What open source developers do wrong

- The biggest complaint is that documentation is rubbish.
- Thinking that users will not notice issues your code has (bugs or design).
- Not responding to requests or recommendations.
- Choosing the wrong type of licence.
- Assuming your project will get additional developers as part of the initial concept.

Forking an existing project

- Is it worth doing? Can your fork add value.
- Forking has been used as a protest action (X-Org being a major example)
- Fork vs submodule, can you simply extend/utilise a project?

Research your field

- Has anyone else done what you want to do?
- What can you learn from their efforts?
- If someone has done something, it doesn't mean you can't, but maybe collaboration to extend is an option?
- Is there a demand? (if their isn't it needn't matter, but it might).

Walk first, Run later

- Don't set goals too far ahead.
- Design, and putting that design online with your code is important, since people *will* look if you apply for a job.
- Perfection is not the initial aim, a design that sucks is better then no design at all, since a design can be critiqued.
- If you are enjoying the project, you are more likely to succeed.

Obligatory XKCD