Creating an Open Source Project

Thiago Macieira

Who am I?

- **Open Source developer for 15 years**
- **Software Architect at Intel’s Open Source Technology Center (OTC) and Tizen Platform Community Manager**
- **Maintainer of two modules in the Qt Project**
  - QtCore and QtDBus
- **Platform Community Manager for Tizen**
- **MBA and double degree in Engineering**
- **Previously, led the “Qt Open Governance” project**

Who is this presentation for?

- **Developers and decision makers working with open source code**
  - Or in the process of getting open-sourced
  - Or thinking about it
- **People interested in recently-opened code**
- **Everyone who is trying to answer the question**

“We’ve published the code, what now?”

(That is, you've just watched Ibrahim Haddad's presentation)

When should you create an Open Source project?

- **When you’re facing the following situation:**
  - “We’ve developed some code in our company because we had to”
  - “This code might be useful to other people and companies”
  - “Obviously, we want some benefit for our effort”
- **The code is getting released under an Open Source licence:**
  - GPL, LGPL, BSD, MIT, MPL, AFL, “Apache License”, etc.

Why should you create an Open Source project?

- **Benefit to the company**
- **Solution to problems others had and didn’t even know**
- **Continuity for the project**
  - Should your company decide to stop developing it (“*bus factor*”)
- **In certain cases, improvement to competitors’ products**

What benefits can I expect?

- **Developer community:**
  - Code development
  - Discussions about the code and about improving it
  - Bug fixing
  - Documentation
  - Attracting new developers

- **Non-developer community:**
  - New requirements, improving ideas, insights
  - Bug reporting and new test cases
  - Promotion, marketing
  - Support for infra structure

What does my company get from that?

- **Better code**
  - *Given enough eyeballs, all bugs are shallow* – Linus Torvalds
  - More supported functionality or use-cases
- **Larger ecosystem**
  - Recruiting, consulting, etc.
- **Recognition of the company as innovative and supporter of Open Source**

Creation of the Community

Maintaining the community

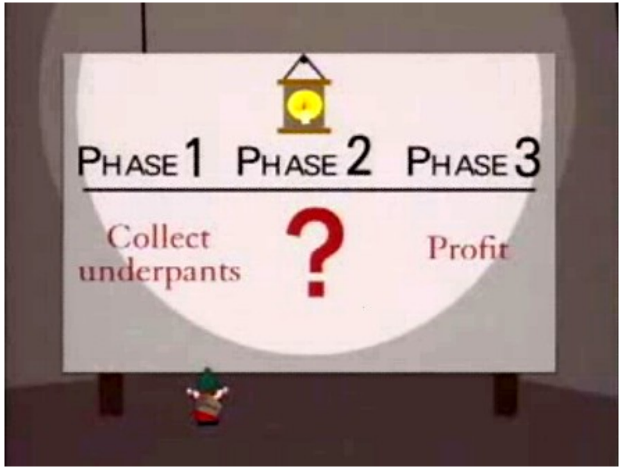
and participation

Some points are illustrated with the experience making Qt an Open Source Project

"We've published the code, what now?"

- **If we were to ask on http://slashdot.org, the answer would be:**

  1) *Publish* code
  2) *?*
  3) *Profit!*



What does an Open Source project consist of (1/2)?

- **An Open Source project contains, at least:**
  - Open source code, under an approved licence
  - Developer community
  - Communication channels
  - Source code management system (often)
  - Releases (usually)

What does an Open Source project consist of (2/2)?

- **Optionally:**
  - Bug tracking system (“bugtracker”)
  - Quality assurance (QA) team
  - Non-developer community (artists, translators, technical writers, marketing, community management, etc.)
  - A lot more

Create the infrastructure

- **Servers, sites and services**
- **Mark what's optional and what is mandatory**
  - The minimum necessary to work is mandatory

For Qt, the following was mandatory:
- Own domain, web site and Wiki (qt-project.org)
- Source code and contribution management tooling
- Bug tracking

Publish, communicate and generate interest

- **After all, if no one knows the code exists...**
- **Communication channels:**
  - Your company's website
  - The developers' blogs
  - Forums and relevant mailing lists
- **Create the project's website**

For Qt, we created a temporary site, a wiki, and mailing lists.

The announcement was done in my blog; we created interest by talking directly to people we identified as potentially interested.

Define a strategy

- **Know what you want, know what you have**
- **Know the advantages of the code:**
  - What it does
  - What it doesn’t do (yet)
  - What it will never do (delimiting the scope)
- **Identify an audience**
  - Who would use this code?
  - Who would be interested in participating?

Talk to your ~~competitors~~ collaborators

- **Collaborative projects usually involve companies in competition**
  - Seek to include your competitors
  - Increases the value of the project
- **Examples: Linux kernel, Yocto Project**

Minimally define processes

- **Do this with your prospective community!**
- **Answer this question:**
  - How does a contribution go from idea to released code?
- **Don’t dwell on details, because there will be variables you’re not aware of yet**

Define the decision-making structure

- **It's equally important to decide "how" decisions are made as "who" makes them:**
  - Who makes decisions, which decisions?
  - Who can reverse decisions of others?
  - In case of conflict, who to ask for help?
- **Recommendation: analyse other communities**

We chose four principles that guided us in our decisions:

Meritocratic, inclusive, open, fair

Example: Qt Project's structure

- **Based on analyses of Linux, KDE, and WebKit**
- **3+1 participation levels:**
  - Contributor: everyone who wants to
  - Approver: can make decisions on inclusion or rejection of code
  - Maintainer: responsible for the quality and direction
    - Chief Maintainer
- **Simple and/or automated processes**

Allow the discussions to go on…

- **It's not necessary to have all the answers**
- **In fact, it's better *not* to have them:**
  - The community will feel more involved if it helps in finding the answers

The first step was to create a project (creatively) called "*Open Governance*", for which we had an objective: create the rules.
We spent months discussing the rules with the community, for the community.

… But keep the mind on the ball

- **Be very clear on the objectives that need to be reached**
- **It’s ok to have a “cheat sheet”:**
  - The community will not have answers for everything, or it might get stuck and lose sight of the objective
  - Give directions only, don’t impose solutions

Before we started the public discussion, we discussed internally what we wanted and what we didn’t want (we had a product to release).

We also had a mental model of what we wanted to have.

Deal with legal issues

- **Choose the licence carefully**
  - Avoid writing your own licence text
  - Use one of the existing and known licences
- **Verify the risks with the Legal Dept.**
  - Protect your company and others against unnecessary risks

The product already had a licence: LGPL version 2.1 and GPL version 3.

One important risk we knew of was about software patents.

If there’s interest, the community will come

- **It doesn’t require a lot of effort**

- **But don’t fool yourself: few projects will be as big as Linux**

Creation of the Community

Maintaining the community and participation

Two sides of the same coin

- **Internal transformation**
- **Maintenance of the external community**

Internal transformation

- **Possibly the hardest part**
- **The team must now operate as an Open Source project**
- **Change of the way of thinking**
  - “Our project” *versus* “The project”

In Qt’s case, we had 250 full-time professions working on the code and 15 years of history.

It was necessary to prepare trainings on the new tooling and on “how to interact with the community”

Internal contributors

- **Your company's professionals are now "project contributors"**
- **The same rules must apply to everyone:**
  - Requirements imposed on the external contributors to gain privileges now apply to internal contributors too
- **Many people will have to work with externals**
  - Be careful about confidential information

There's help

- **Other people who have been through this process**
- **Consulting company specialised in Open Source trainings**
  - For example, the Linux Foundation offers courses on how to deal with Open Source
- **Be open with the community, don't hide information**

External community

- **Passive maintenance:**
  - It should be part of the internal contributors’ day to day
  - Keep the quality in the discussions
- **It shouldn’t be hard, it should simply be the work you already do**

Active maintenance

- **Special attention required and might have cost associated**
  - Stimulating external contributions
  - Helping new contributors
  - Conflict resolution
- **Necessary to avoid deterioration and emptying of the community**

Meetings and conferences

- **Meet contributors face to face:**
  - Great way to resolve conflicts, with a beer glass
  - Helps preventing future conflicts
- **Improves the project's image and that of the sponsors**

Hackfests

- **One objective:**
  - Develop a functionality, solve NN bugs, rewrite documentation, update the website, etc.
- **One location:**
  - For example, your office
- **Some people:**
  - Include external people and “new blood”
  - Make them feel like part of the project
- **Low cost**

Long term…

- **Some activities become routine**
  - Contributors know each other and how to behave
  - Community grows and becomes more attractive
  - Certain “boring” tasks get done by volunteers (who don’t find it boring)
- **And some people will go to conferences to talk about the experience they gained**

Any questions?

Recommended further reading:

– Open Advice book, http://open-advice.org

Thiago Macieira

thiago.macieira@intel.com

http://google.com/+ThiagoMacieira