TIZEN™ DEVELOPER CONFERENCE
MAY 7–9, 2012

Engaging Open Source Projects (with a corporate view)

Thiago Macieira, Software Architect, Intel OTC

Who am I?

- Open Source developer for over 15 years
- Working for Intel’s Open Source Technology Center
- Maintainer of modules in the Qt* open source project
- MBA and double degree in Engineering
- Collect way too many air miles presenting in events

Disclaimer

- I am not a lawyer and this is not legal advice
- This presentation is not made to conform to any specific company's policies regarding Open Source
- Other:
  - This presentation was not reviewed by Intel Legal
  - You should always check your company's current policies on the activities before executing them
  - You should consult your manager or your department's legal counsel if you are unsure
  - The suggestions made in this presentation may become obsolete, in which case you should rely at most on the general suggestions. That may be caused by:
    - Policies changing
    - Open Source way of working changing
    - Universal constants changing and altering reality. I hereby disclaim any foreknowledge of such an event and may not be held responsible for it. Additionally, should the Universe change, in no way should the contents of this presentation be held under scrutiny, notwithstanding the ability of current detectors to determine that universal constants have changed. If you can read this small print, smile. If you can't, you're not losing anything anyway.

Audience check

- Work for a medium- or large-sized corporation?
  - Already contributed to Open Source projects?
- Maintainers or contributors in Open Source projects?
- Both?

What “*uses*” are there?

- Compiling the code and using it in a product
- Modifying the code
- Reporting issues and bugs
- Addressing issues and bugs found in the product
- Requesting features
- Interacting on mailing lists and forums
- Contributing code, documentation, examples, unit tests, benchmarks, etc.
- Advocating, blogging

Uses and company policies

- Any and all uses may be prohibited or otherwise limited by company policies
- They might change depending on whether any external parties are involved
- Examples?

Engaging *uses*

- Reporting issues and bugs
- Requesting features
- Interacting on mailing lists and forums
- Contributing back to the project

Disclosing Open Source usage

- Company policies may require that you disclose if:
  - You are using Open Source software
  - Are modifying such software
  - Are including such software in products sold to the public or given to third parties (for example, contractors)
- When you do, make sure to include the licence terms
  - For example, BSD-licensed software often requires acknowledging its use in the documentation

Agenda

Basics

Contributing code

Protecting the secrets

The first things you need to know...

- Open Source developers are also people
  - Often employed by a company to work on that project

- The project comes before the company
  - Usually because there's more than one company *using* the project

Open Source projects aren’t hobbies

- When *using* an Open Source project, remember that
  - Many man-years (or man-millennia) have gone into development
  - You’re not doing the project a favour by using it
- But, do demand equal respect

Barriers, like a company

- Different geography, different cultures
- Different time zones
  - Real-time communication is usually difficult
- Different native languages!
- Different project assignments
  - The person you need may be busy with other tasks

Barriers, not so much like a company

- Developers sometimes bite
  - Sometimes, justifiably: netiquette, FAQs, BKMs
  - Other times, not so – *king of the hill*
- Part-time developers may reply on weekends only
- Very often, developers “don’t get” the need for secrecy

Principles of Open Source projects

- Usually, projects are founded on the these principles:
  - Transparency
  - Inclusion
  - Meritocracy

Welcoming new contributors

Newcomer

- Don’t ask for too much
- Don’t assume anyone will understand your urgency
- Do your “homework” first
- Explain in detail
- Be prepared to give more information

Existing OSS contributor

- Be receptive and professional
- Don’t snap back
- Point to useful resources the new contributor may have overlooked
- Understand that there may be limits to what they can tell

Communication channels

- Open Source projects like to use:
  - Mailing lists
  - IRC channels
- Use them wisely:
  - Mailing lists for long, detailed messages
  - Asynchronous communication
  - IRC for short, real-time communication

Netiquette

- Set of guidelines to facilitate communication:
  - Originally defined by RFC 1855
  - Be courteous, professional
  - Some simple things, like NOT SPEAKING ALL IN CAPS, giving the benefit of the doubt, etc.
- Mailing list communication:
  - Quote properly; avoid long signatures, top-posting, HTML emails, cross-posting, and long Cc: lines
  - Teach, but don’t complain about them

Finding your “local elders”

- Who best to give you advice than those who have already been through this?
  - Try to find people experienced with Open Source in your company, department, or region
  - Encourage such a group if one doesn’t exist (and join them)
  - Keep a list of best-known practices
- Also true for existing OSS developers:
  - Advise you on how to deal with corporate developers

Agenda

Basics

Contributing code

Protecting the secrets

Benefits of contributing back

- Less maintenance:
  - No forks to be kept for long periods
  - Ability to benefit from upstream bug fixes
- Gain of merit:
  - Will allow you to have more influence on decision-making in the future

Gracefully accepting rejection

- Your contributions may not be accepted, then what?
  - Do request a reason, a fair one, don’t accept vague answers
  - Do improve what you were requested to improve
  - Check the timelines and contribution policies
  - Then re-submit
- Sometimes, the problem is the change itself
  - There are ways around it

Working around

- You can keep a local branch of your changes
- But you should keep it updated relative to your upstream
- If possible, you should make that branch public
  - Others might be interested in your solution
  - Which, in turn, might be enough to get the decision reversed
- If you make it public, take care not to portray it as a “fork”

Ways of developing

- Development culture, usually codified in a file or wiki:
  - How to format your commit messages
  - How to style the source code, to adhere to coding policies
  - How much to change in a commit
  - When changes are acceptable, when they’re not
  - What happens after you submit the change
- The closer you get to the expected way of working, the higher the chance of getting the change in

Coding style and policies

- Follow that of the project you're contributing to
  - Placement of braces, use of tabs, naming of functions
  - What you can or cannot use
  - Whether you need to include unit tests
  - Where you must test

- Don't argue, that leads to flame wars
  - "Bikeshedding"

Tooling

- Engineer’s motto: *the right tool for the right job*
- Especially for development, tooling is important:
  - Version control systems
  - Code reviewing procedures (formal or *ad-hoc*)
- Sometimes, tooling overlaps:
  - For example, for Linux kernel development, you really need a good email client and good email servers

Maintaining your submission

- You’re often required to maintain what you submit
  - Make sure you have the time!
  - You can count on users finding issues
  - And they’ll do it in architectures, platforms, and configurations you’ve not dreamed of
- Understand how long your commitment must be for
  - Some projects allow handing over the contribution

Release scheduling

- Usually set way in advance:
  - Don’t ask for changes to suit your deliverables
  - But do make sure that everyone knows what they are
- It’s unfair:
  - Hopefully, equally unfair to everyone
  - Your urgency is not shared by others
  - And vice-versa: your lack of urgency does not imply no one else has urgency

Testing at releases

- Combination of maintenance duty and release schedules
- Releases mean more people testing
  - More issues found, under more unexpected conditions

Agenda

Basics

Contributing code

Protecting the secrets

What secrets

- Companies have secrets to protect:
  - Trade secrets
    - Your upcoming product’s specs
  - Intellectual Property rights (usually, patents)
  - Information obtained through NDA from third-parties
  - Information private by law
    - Like the names of your co-workers
  - Other / misc
    - Codenames

When interacting with OSS projects

- Know what your company considers a secret
- Keep them in mind:
  - To provide enough information
  - To not provide too much information
- If in doubt, ask your colleagues and “local elders”
- Err on the side of caution: don’t disclose

Example of interactions

- Probably not ok:

  “I’m using project Xxxx and I get a compilation error! Help! I need this by the end of the week because we’re releasing next week!”

  - You’ve disclosed your release dates, were they public knowledge?

- Probably ok:

  “I’m using project Xxxx and I’m having problems building with these configuration options. Here’s the compilation error. Can anyone help?”

  - Make sure to not include codenames or secret info in the log

If contributing code

- Make sure you've got all the permissions
  - Especially if you didn't write it yourself
  - Or used any secret information in the development
- Make sure that the Legal department is aware of it
  - They usually have pre-approved licence texts
  - They'll know if patents may be an issue
- Check the timing: not too soon, not too late

Agenda

Conclusions

Checklist

- Know what a secret is
- Don’t bypass the lawyers — keep them in the loop
- Know who your “local elders” are
- Act professionally, demand equal respect
- Learn the ways of working and tooling
- Make sure you have the time required for the task
- Document the best-known practices
- Do make yourself known and gain merit by contributing
- Become an “elder” yourself!

Any questions?

- If you wish to contact me:

  thiago.macieira@intel.com

- This presentation will be available online on the conference's website later

TIZEN™ DEVELOPER CONFERENCE
MAY 7–9, 2012