The project that brings you Qt

Thiago Macieira, Qt Core Maintainer
Software Architect, Intel OTC
Berlin, Nov 13-14, 2012
Santa Clara, Dec 6-7, 2012
Executive summary

• Qt Project is how Qt gets developed
• Open Source and commercial licenses based on it
• Participation is open to all
Who am I?

• Open Source developer for 15 years

• Software Architect at Intel’s Open Source Technology Center (OTC) since last year
  – Living in Portland, Oregon

• Maintainer of two modules in the Qt Project
  – QtCore and QtDBus

• MBA and double degree in Engineering

• Previously, led the “Qt Open Governance” project
  – Ended with the creation of the Qt Project
Who am I?

- Open Source developer for 15 years
- Software Architect at Intel’s Open Source Technology Center (OTC) since last year
  - Living in Portland, Oregon
- Maintainer of two modules in the Qt Project
  - QtCore and QtDBus
- MBA and double degree in Engineering
- Previously, led the “Qt Open Governance” project
  - Ended with the creation of the Qt Project
The Qt Project celebrated 1 year on October 22

http://qt-project.org
Some numbers

In the one year completed on Oct. 22:

• 25,224 commits reviewed & accepted
  – ~38000 submitted, 41000 approving comments

• From 451 different people (and two bots)

• Employed by 81 distinct companies or entities

• Top contributors: Nokia*, Digia*, KDAB*, Intel, RIM*

* Other names and brands may be claimed as property of others
Contribution graph

Week

Commits

others
collabora.com
moji-mobile.com
woboq.com
mabrand.nl
QNX (by kdab.com)
intel.com
kdab.com
(individuals)
digia.com
nokia.com
We want your participation too
Reasons to participate

• Learn what’s going on
• Influence decisions
• Gain merit in the project
• Add features & fix bugs you need
  – You / your company keeps the copyright
• Help Qt go really everywhere
Before

2009

Open repository
Contribution model

2010

Qt Open Governance
discussions

2011

Qt Project launches

Daily snapshots
Contribution via
copyright assignment

© 2012 Intel
Before the Qt Project

Contribution Model
- Clumsy
- Different for Qt engineers
- Not the model we wanted for ourselves

Open Governance project
- Investigated other projects
  - Linux kernel, WebKit, KDE
- Talked to a lot of stakeholders
- Discussed in the open
- Modelled ideal infra
Principles of an Open Source project

- Fair
- Transparent
- Inclusive
- Meritocratic
Meritocracy: Project roles

- **Chief Maintainer**: Leads maintainers with ultimate decision making power
- **Maintainers**: Final responsible for a given module or port
- **Approvers**: Approves / rejects contributions
- **Contributors**: Contributes code to Qt
Meritocracy: Project roles

- Chief Maintainer: Leads maintainers with ultimate decision making power
- Maintainers: Final responsible for a given module or port
- Approvers: Approves/rejects contributions
- Contributors: Contributes code to Qt
Meritocracy: Maintainers

They have a duty

- Must ensure all contributions are reviewed
- Must ensure the quality is good
- Must ensure the rules are followed

They get some rights

- Right to set direction
- Right to override approvals and denials
Fairness: same rules for everyone

Technical fit
• Does it build?
• Is it documented?
• Is the API reviewed?
• Does it follow the coding style?
• Does it work as intended?
  - (at least on reference platforms?)
• ...

Spirit fit
• Is it the right implementation?
• Is it the right time?
• Does it take Qt in the right direction?
• ...

© 2012 Intel
Transparency: decisions in the mailing list

Face to face discussions are permitted

All decisions must be posted to the mailing list
  • Others can review / offer new insights

Lorem ipsum?
Ok!
To: ML
Lorem ipsum
Inclusive: contributor summits
Very simple to contribute

Review and accept collaboration agreement

Create JIRA account & configure Gerrit account → Create personal clone → Configure personal clone

Only once

Early warning System (EWS) → Develop & push to refs/for/<branch> on "gerrit"

Success

Add reviewers to review request (once per patch)

Developers review patch

Success → Further iteration required → Failure

Rejected

Failure

You or Approver stage commit

CI system builds and runs autotests

Success → Patch integrated into Qt project

Accepted
No, really, very simple to contribute

1. Create contribution
2. Review in Gerrit
3. Update & improve
4. CI tests & commits

* Other names and brands may be claimed as property of others
Live contribution demo!
Release process

Hybrid time-based & quality based

Driven by a release team
Other activities

Marketing
• Going to events together, as Qt Project
• Social media

Infrastructure
• Running the bug tracker, continuous integration, mailing lists, ...
A success story?

• Survived Nokia changes
• Developed Qt 5 with contributions from many
• Released several Qt versions
  – 4.8.1, 4.8.2, 4.8.3, 4.8.4, 5.0 alpha 1, 5.0 beta 1, 5.0 beta 2
• Good cooperation between partners
• Developer Days 2012
Success case: maintainers

Four base libraries, four different companies

QtCore  QtGui  QtNetwork  QtWidgets

intel  digia  accenture  AKDAB
Questions?

Thiago Macieira

thiago.macieira@intel.com

Links:
Website: http://qt-project.org
Mailing lists: http://lists.qt-project.org
IRC: #qt and #qt-labs on Freenode