Qt: Automotive Adoption and Governance of FOSS

Qt's Challenges and Experience

April 13, 2018

What is Qt?

Qt (/kjuːt/ "cute"[7][8][9]) is a cross-platform application framework. It is used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while still being a native application with native capabilities and speed.

https://en.wikipedia.org/wiki/Qt_(software)

Who Uses Qt

tableau

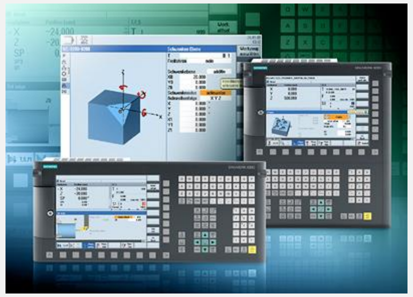
Wolfram Mathematica

AUTODESK® MAYA®

VLC MEDIA PLAYER

OneDrive

Qt in Devices



Siemens Sinumerik Series

Fanuc iHMI Platform

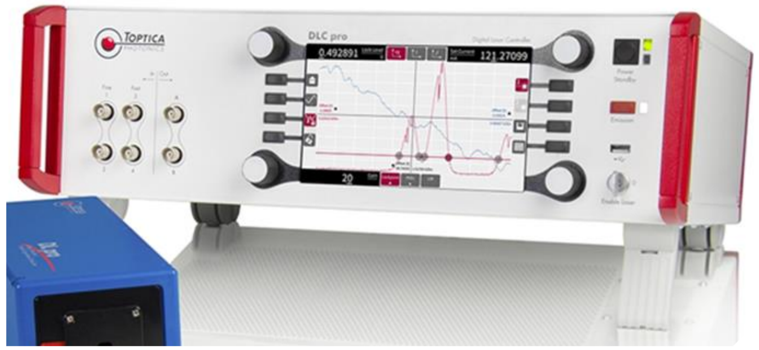
Panasonic Inflight Entertainment

DMG Mori Celos

N&W Coffee machine

Contex Wide scanner

Harman Connected Car Solutions



Zühlke Digital Laser Controller

Qt use in Automotive

All Screens in the car.

The *Digital cockpit.*

Not autonomous control systems.

Questions:

- How that **software is governed** (how it is maintained, repaired, inspected; who may change it how, and with what accountability; how vehicle owners and **users can exercise their rights in that software**, and what their rights ought to be), will determine the real social consequences hidden behind phrases like “self-driving,” “connected cars,” and “robot revolution.”

- How can innovations in software engineering help us to improve reliability and serviceability of embedded automotive software? How can user innovation be maximized, **allowing car owners to modify the operation of their vehicles**, while preserving safety and allowing manufacturers to limit their liability? How can vehicles be upgraded in service without also creating dangers of malware injection and privacy invasion?

Open Source in Automotive and the meaning of *free*

- Why do we care about open source in automotive?

- Freedom to inspect the code
- Freedom to change the code

- Not necessarily free of charge

- Why Open Source is good – or rather why proprietary can be bad
  - Real world example

The Economics of Open Source

› Lets explore….

1. People working in a basement?

- Sometimes
  - node.js started that way but rapidly became company sponsored.
- Lots of small projects and add-on modules (e.g. Python modules)
- Great learning and proving ground for individual developers
- OpenSSL

OpenSSL – Heartbleed, Code introduced December 2011 Reported April 2014 (https://en.wikipedia.org/wiki/Heartbleed)

Software engineer John Walsh commented*:*

> *Think about it, OpenSSL only has two [fulltime] people to write, maintain, test, and review 500,000 lines of business critical code.*

The OpenSSL foundation's president, Steve Marquess:

> *The mystery is not that a few overworked volunteers missed this bug; the mystery is why it hasn't happened more often.*

Paul Chiusano suggested Heartbleed may have resulted from **failed software economics.**

50% of servers relied on this? (https://www.troyhunt.com/everything-you-need-to-know-about3/)

2. Big Open Source Projects

https://www.linuxfoundation.org/blog/successful-open-source-projects-common/ (2016-2017)

What differentiates the most successful open source projects? One commonality is that ***most of them are backed by either one company or a group of companies*** *collaborating together.*

Chromium, Tensorflow , AngularJS by Google,
React by Facebook
Docker/Moby by Docker
VS Code and Office Developer by Microsoft
Ansible by Red Hat,
ElasticSearch by Elastic

Auth0 by Auth0
GitLab by GitLab
Ruby on Rails by Basecamp
Ionic by Ionic
Terraform by HashiCorp
Chef by Chef Software

In all cases, there are major contributions by developers from outside companies and independent developers, but many or most of the maintainers are employed by one company.

3. Consulting, Training and Support

What you see in the market are companies, for example in the Hadoop space, that are open source and essentially have the same go to market model, which is selling consulting, training and support. That's their main revenue stream. They do ***well, because Hadoop is so complicated that there is a lot of demand for training, consulting and support****.*

From <https://diginomica.com/2015/11/06/a-look-at-how-mongodb-plans-to-make-open-source-profitable/>

Does this only work for complex hard to use software?

The goal of Qt is to make an easy to use well documented framework that's fun to use.

4. Open Source moral obligations

- Give-a-penny-take-a-penny (Ben Balter – opensource.com)
- Take a lot, give a little (Jim Zemlin - some Linux Collaboration Summit)
- Copy left licenses (GPL)

-> Fair amount of corporate avoidance

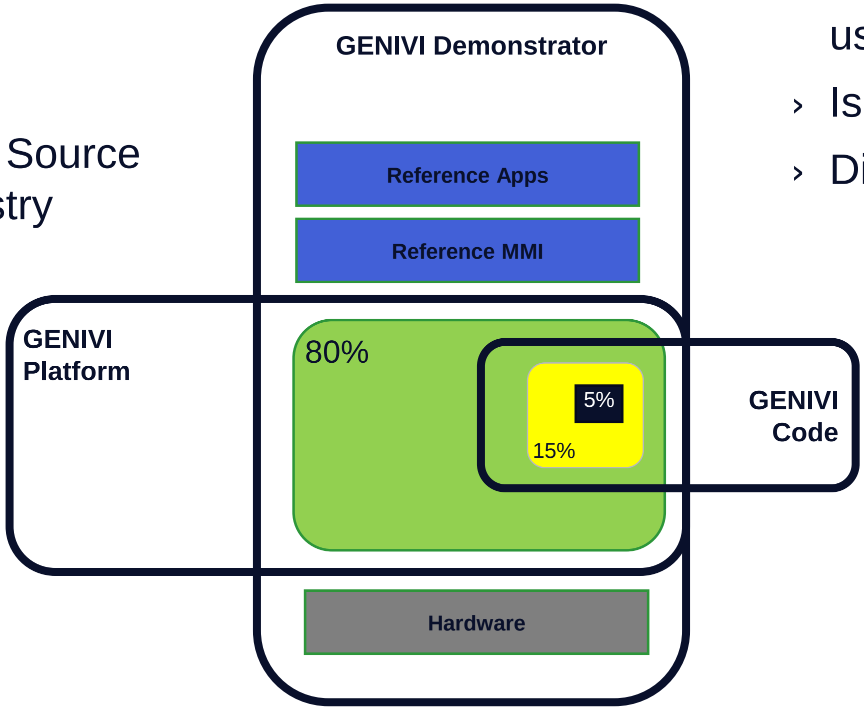
5. Consortiums - Automotive Open Source Initiatives

GENIVI

- **Ideal**: Lots of IVI software is commodity. Everyone has to build it. Lets do it once and share.
- **Reality**: Tier1s afraid. Everyone has their secret sauce.
- Specification based
- Biggest success: raising Open Source awareness in automotive industry

AGL

- Code is the spec.
- Membership fees to fund development (~$2M/year)
- Are the funding OEMs really serious about using it?
- Is it moving fast enough?
- Discarding a lot of GENIVI work



Qt Business Models – Four Phases

1. Trolltech (1993-2008)
   - Commercial & GPL.
   - Make money using Qt -> You should pay

2. Nokia (2009-2012)
   - Commercial & LGPLv2.1 Please use for free
   - Qt commercial rounding error on balance sheet
   - Killed Gtk
   - Number of paying customers halved

3. Digia / The Qt Company (2012-2016)
   - Commercial & LGPLv2.1
   - Please pay us.
   - Limited funding to advancing Qt

   - Ever met an Automotive purchasing manager?
     - Their job is to drive the cost to **zero**

Qt Business Model – The Fourth Phase

4. The Qt Company (2016...)

- (L)GPLv2.1 -> (LGPLv3) + Commercial + GPLv2
- Tivoization fix - if you're building a locked down device you pretty much have to pay
- Feedback…

- Additional parts of the business & licensing model
  - Commercial only add-ons.
  - KDE Free Qt Foundation
    - Limits what we can do
    - Which licenses we use
    - Requires regular releases

Qt Governance

- Qt Project (2011) created by Nokia
- Documented https://wiki.qt.io/The_Qt_Governance_Model
- All code in open repository (git)
- Code commits are reviewed and approved (gerrit, list of Approvers)
- Bugs are in the open (unless commercially sensitive) https://bugreports.qt.io
- https://wiki.qt.io/Maintainers
- Public development mailing list http://lists.qt-project.org/pipermail/development/

Conclusions

- Open Source has been a fantastic benefit
- Good software doesn't come for free
- Big corporations will give away great software for their own motivations
- Business models vary by type of software
- Some corporations take without contributing back
- Qt has chosen a dual commercial (L)GPL strategy – it works for us