FLOSSing proprietary code

Enlisting the help of the FLOSS community to build a commercial product

David Kaltschmidt <david@inf.fu-berlin.de>

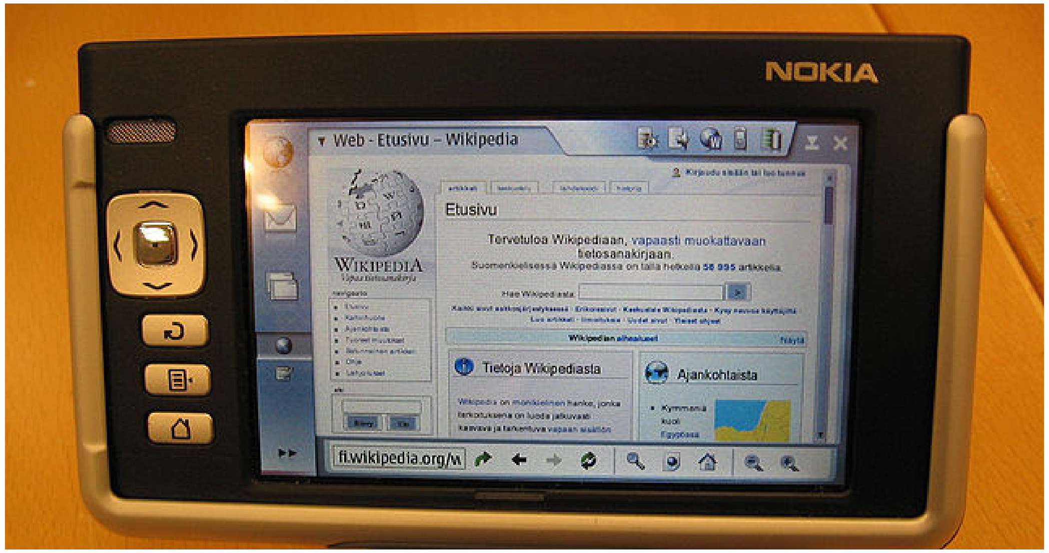
Contents

- Development models: private, public, private-collective
- Case study[1]: “Nokia Internet Tablet”
  - Benefits, hidden costs, strategies
- Survey[2] of Embedded Linux developers
  - Licenses
  - Reasons to reveal
  - Revealing behavior
- DOs and DON’Ts

1. M. Stuermer, S. Spaeth, and G. Von Krogh. “Extending private-collective innovation: a case study.” In: R&D Management 39.2 (2009), pp. 170–191.
2. J. Henkel. “Selective revealing in open innovation processes: The case of embedded Linux.” In: Research policy 35.7 (2006), pp. 953–969.

Case Study: “Nokia Internet Tablet”

- Development started as early as 2000, 1st release 2005
- Designed for Web-browsing and e-mail
- No built-in phone
- Operating system based OSS


NOKIA
Web - Etusivu – Wikipedia
Etusivu
Tervetuloa Wikipediaan, vapaasti muokattavaan tietosanakirjaan.
Tietoja Wikipediasta
Ajankohtaista
fi.wikipedia.org/w

Based on OSS

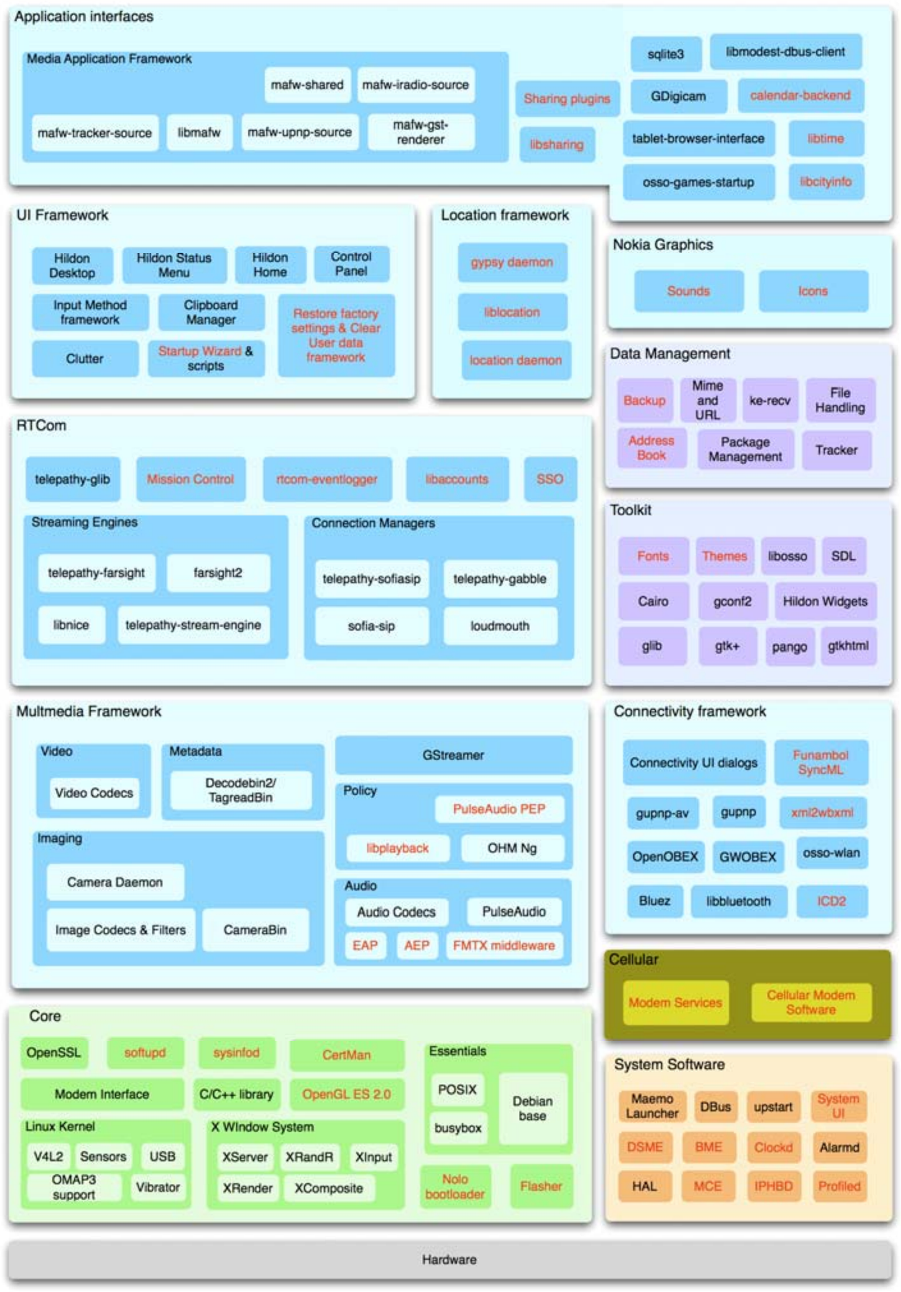
Proprietary components are marked in red.

Modules dominated by proprietary software:

- System software
- Nokia graphics
- Location framework

Single strategic components:

- Boot-loader/Flasher
- Restore/Backup
- Fonts/Themes



Investment models

- **Private**: invest and create ideas internally, commercialize and protect IP
- **Collective**: public goods, give and take, public subsidy, non-exclusive
- **Private-collective**: private resources to create public goods, non-exclusive, IP rights forfeited
  - Nokia paid open source developers, or hired them
  - Nokia created a software platform: Maemo

Benefits 1/2

- Shared development cost
  - Reuse invites further contributions
  - Best case scenario: Higher quality at reduced fixed cost
- Faster time-to-market
  - Nokia developed a whole operating system in a short time
  - System integrator of loosely coupled component providers
- Learning from others’ contributions
  - Learn while creating something worth sharing
  - Learn by looking at other developers’ work

Benefits 2/2

- Reputation gain
  - Commitment to open source
  - Volunteers contribute to platform and later build apps
  - Recruitment opportunity for Nokia
- Widespread adoption/dominant design
  - Low entry barrier to participate
  - Network effects
  - Consolidation, e.g. GNOME Embedded Platform

Costs/Dangers 1/2

- Obvious: paying developers, forfeiting IP rights

- Lack of differentiation
  - Clone with same architecture, same OSS software
  - Nokia kept parts of UI and hardware layer closed source
- Losing business secrets
  - Company-specific code can be minimized
  - Outgoing code must be checked, NDAs

Costs/Dangers 2/2

- Giving up control
  - Increasing dependency on external technology
  - Direction of development of the OSS
- Organizational inertia
  - Clear platform releases by legal department
  - Bureaucratic process vs. dynamic community

Generalizing

- Was Nokia unique?
- Survey of embedded Linux developers:
  What are conditions under which openness is feasible?

Issues:

- Voluntary vs. forced openness (GPL)
- Selective revealing
- Proportion open vs. closed source
- Type of code
- Reasons to reveal
- Revealing behavior

Licenses

GPL:
Recipients of software have right to see source code, Copyleft, no cross-license linking

LGPL:
Copyleft only applies to the library/software itself, can be linked, GPL-compatible

APL:
Permissive, attribution, GPL-compatible, track modifications

MIT/BSD:
Permissive, attribution, no copyleft, GPL-compatible

Ways to protect software

- Release sources only upon request
  - If no one asks, source code remains “secret”
- Lead time
  - Distribution of code only when released/sold to customers (GPL)
- Software architecture
  - Consider all licenses of the reused OSS
  - Break up system into subsystems (a program in GPL terms)

Survey results

- Percentage of shared code differs strongly for commercial firms: on average 49% of all code is shared,
  BUT σ = 35%, min = 1%, max = 100%

- 49% share more than 5 years ago (2000)

- Generic code is shared by 63% of HW firms, 85% of SW firms
- Product specific code is shared by a third of firms

Reasons to reveal

For HW companies:

1. GPL requires it
2. Appear as good OSS player
3. Bug fixes by others
4. Advance development
5. Reduced maintenance effort
6. Revealing good code improves technical reputation

For SW companies same result, except “revealing good code improves technical reputation‟ ranked 3rd.

Revealing behavior

Other things being equal…

- Small firms reveal more code
- Policies encouraging to reveal code do not lead to more code being shared
- SW firms reveal more than HW firms
- Longer experience with embedded Linux leads to higher share of code being revealed
- Expecting development support and reputation gain lead to higher share
- Sharing as Marketing does not lead to more code being shared
- GPL is not a motivator

DO and DONTs

DO read the licenses

DO identify generic parts

DO improve your architecture for reuse

DO trust the community

DO respect its meritocracy

DO lead by example

DON’T focus on protection

DON’T switch back to closed

DON’T try to control the community

DON’T take a project hostage

DON’T fork and expect to keep benefiting

DO keep sharing.

DO learn from others.

THANK YOU. QUESTIONS?

Sources:

1. M. Stuermer, S. Spaeth, and G. Von Krogh. “Extending private-collective innovation: a case study.” In: R&D Management 39.2 (2009), pp. 170–191.
2. J. Henkel. “Selective revealing in open innovation processes: The case of embedded Linux.” In: Research policy 35.7 (2006), pp. 953–969.
3. Wikipedia: Comparison_of_free_software_licenses, Licence_compatibility, Maemo_(operating_system)