FOSS Best Practices

PLI Open Source Software 2008

San Francisco, December 10, 2008

Richard E. Fontana

Open Source Licensing and Patent Counsel

Red Hat, Inc.

Agenda

- Understand FOSS
- Acquire realistic view of risk
- Develop FOSS policy and implement process
  - Distinguish use cases
  - Due diligence on **all** inbound third-party code
  - Compliance for distributed code; audit and remediation
- Learn to read source code
- Encourage developer participation in upstream communities

For many lawyers, FOSS remains exotic

- FOSS is:
  - A **culture** with distinct licensing/governance traditions, development practices and distribution norms
  - A community-led experimental **law reform effort** to build a new legal regime for software on top of (and manipulating) traditional software IP/licensing law
- Lawyers dealing with FOSS often lack:
  - familiarity with FOSS history and culture
  - adequate knowledge of the subject technology
  - appropriately non-formalist perspective

FOSS law as customary law

- Legislation, regulations, case law provide little guidance
  - **Custom** is de facto source of law for FOSS (like *lex mercatoria*)
- Challenges for lawyers:
  - No single source of legal authority
  - “Unwritten” tradition; no collection of legal sources
  - In absence of court decisions, conflicts between custom and underlying legal strata

Defining FOSS

- No standard definition
  - Rather, evolving norms of free software licensing
  - Prestigious community organizations have persuasive authority as articulators of licensing norms (FSF: free software definition, Debian: DFSG, OSI: Open Source Definition)
  - Licenses at outer boundaries: typically badly drafted, unpopular, unique to particular companies or projects, often have traditional binary commercial licensing restrictions; best to avoid
- Hundreds of legacy licenses, but standardization on a very small set

Defining FOSS

- FOSS code is (supposed to be)
  - made available to you free of royalty obligations
  - in intelligible source code form, or with source code readily available for no additional charge
  - free of field-of-use restrictions
  - for all practical purposes free of restrictions on internal/private execution, copying and modification
  - free of **undue (*non-customary*) burdens** on subsequent distribution and licensing
- For FOSS code legitimately incorporated into proprietary software products, conditions generally continue to run with the code

License categories (by copyleft criterion)

- Strong copyleft (GPLv2, GPLv3, Affero GPLv3)
  - Modified versions, **if distributed*,** cannot be under a more restrictive license (*AGPL: or network-facing)
  - Binary distribution requires source code disclosure
- Weak copyleft (LGPLv2.x, LGPLv3, MPL, EPL)
  - Copyleft scope cut off at file/module boundary
  - Can incorporate into proprietary-licensed binary; disclose source code for copyleft-covered part
- Permissive/non-copyleft (BSD variants, MIT, Apache 2.0)
  - Allow proprietary derivative works
  - No source code disclosure requirement

A few notes on the GPL

- Meaning is determined by **tradition**
- “BSD-equivalent” for internal use
- Binaries (even if unmodified) must be accompanied simultaneously by **complete corresponding** source code or can provide 3-year written offer for source
  - Skilled recipient able to recreate functionally equivalent binary
- Copyleft requirement understood to prevent circumvention by artful packaging
  - Program-centric: does not extend to “mere aggregation”; **no so-called “virality”**
- “No further restrictions” principle

FOSS as development model

- FOSS has also come to be associated with certain software development norms (not a necessary condition)
  - Upstream collaborative non-profit project formed by independent/cross-organizational developers to address particular technical need
  - Source code publicly available; development process largely conducted in the open
  - If successful, project attracts large community of contributors, testers, users; binaries get packaged for major OSes/distros
  - Collaboration uses various Internet media (net-accessible source code version control systems, mailing lists, IRC channels, bug trackers, wikis)

Risk analysis

- Why is there FOSS risk aversion among lawyers?
  - Anti-FOSS propaganda
  - Exoticness
  - Transparency of legal issues inherent in public source code and open development practices
- In reality, risk in use of third-party code is largely the same regardless of licensing
- Some reason to believe IP infringement risk (patent, copyright, trade secret) is lower where source code is publicly available

False risk dichotomy: FOSS vs. proprietary

- Cannot assume without investigation that a given software product does not incorporate third-party code covered by undisclosed licensing requirements (FOSS/non-FOSS)
- All major GPL enforcement cases (FSF actions, BusyBox, gpl-violations.org) have involved closed-source code that vendors apparently assumed to be legitimately proprietary
  - Suggests that FOSS copyright infringement risk is more a risk associated with **apparently-proprietary** closed-source software than with apparently-FOSS

Myth of the litigious hacker

- **FOSS licenses largely go unenforced/underenforced**; developers lack resources or inclination to sue
- Traditional enforcement: voluntary compliance (self-enforced) or community pressure
- Recent assertiveness by small number of GPL licensors is exception proving the rule
  - FSF Compliance Lab (2001) (no litigation); Harald Welte/gpl-violations.org (2004) (Germany); BusyBox litigation (SFLC) (2007)
  - Cases involve most material GPL violations (source code nondisclosure in shipment of embedded devices containing **apparently-proprietary** binaries)

Myth of the litigious hacker

- Non-material license violations are largely ignored or worked out at upstream community level (good example: FOSS license incompatibility)
- Whole provisions in GPLv2 and LGPLv2.x appear to have been widely read out by licensors
- License compliance == **do the best you can** (good faith)
- Commercial dual-license companies present more litigation risk than upstream GPL community projects

Best practices: policy and process

- Develop corporate FOSS **policy** that is realistic, practical and flexible, and specific to particular business units
  - Should address common use cases
  - Revise if it requires too many exceptions
- Implement policy with **process**: should avoid bureaucracy and impediments to sysadmin/developer productivity;
- Include non-lawyers in process
- Good internal information efforts and procedures are necessary (use FAQs, wikis, mailing lists)
- **Inbound code due diligence** (FOSS/non-FOSS)

Distinguish FOSS use cases

- Internal non-development use (e.g. IT infrastructure): presents no risk particular to FOSS
- Use of FOSS tools in in-house development: very unlikely to affect product licensing
  - Output not affected by program license
  - See, e.g., gcc runtime library exceptions
- Non-distributed services interacting with network clients:
  - Be aware of still-rare licenses requiring source code disclosure
- Distribution of products incorporating FOSS code
  - Determine compliance with license conditions

Case 1: Opaque supplier use of FOSS

- Typical real-world case of enforced-against GPL violation: outsourcing of development and system integration to third-party suppliers making undisclosed use of GPL'd code
- Require supplier to:
  - disclose all FOSS use
  - disclose all its upstream vendors
  - detail supplier's FOSS compliance procedures
- Consider contractual risk-shifting to supplier

Case 2: Informal transparent use of FOSS code

- Importance of documenting use of code in product development (without impeding developer efficiency)
  - Require developers to use RCS
  - Require developers to document build instructions

Case 3: M&A

- Apply similar procedures as with suppliers
- Use appropriate reps & warranties and indemnities
- Audit source code prior to closing
- Interview target personnel if necessary
- View simple answers with skepticism
- Need for remediation will not justify delay in closing

License compliance for distributed code

- Upstream licensor expectation is “do the best you can”
- Review as early as possible in the product development process
- Audit source code trees corresponding to shipped binaries
- Be prepared to conduct close copyright analysis
- Remediate where necessary
  - Remove/replace problematic code
  - Release sources
  - Rearchitect closely-interacting components
  - Contact upstream licensors for clarification or negotiate more favorable terms

Reading source code

- Any lawyer dealing with FOSS should learn how to:
  - Unpack a tarball
  - Analyze a source code tree for licensing information
    - `$ grep -ri -C4   license\|copyright   *`
    - Conventions:
      - Top-level metadata (COPYING, LICENSE, AUTHORS)
      - License headers in source code files
  - Browse an RCS via a visual interface

Participate in FOSS communities

- For FOSS projects used by your company, encourage developers to participate actively in the upstream community (has positive business and legal effects)
  - Do not fork
  - File bug reports, submit patches, request new features
  - Keep up with upstream bug fixes, new releases
- Develop good relationships with other authoritative organizations (e.g. FSF, SFLC)

Thank you!

rfontana@redhat.com