Source Code Licensing as an Essential Aspect of Modern Software Development

Technical Briefing

Max Di Penta and Daniel German

“The way software is built is changing”

--Scott Peterson
Senior Counsel HP
Open Office dependency graph in Debian 5

"Anybody can fix code bugs, but only a few can deal with licensing bugs."
--Aaron Siego, previous president KDE Foundation
Software and Intellectual Property (IP)

- IP: “the commercial application of imaginative thought to solving a technical or artistic challenge” [WIPO 2005]
- Since 1980 software has the same protection as any other literary work (US)

An micro-intro to Copyright

- It is a monopolistic right to:
  - Make copies
  - “Perform” the work
  - Prepare derivative works
- Software is like Music and Writing
  - Owners want to control how it is used and reused
- The owner is:
  - In the absence of contract, its author
  - Employer usually owns copyright of employees
Joint Works

• a work is **joint** if the authors collaborated with each other,

• with the knowledge and intention that it would be merged with the contributions of other authors as **inseparable or interdependent parts of a unitary whole**.

• Each author can **independently license** the work
  • But should compensate the co-authors

Derivative Works

• A key concept in software development

• A derivative work is “**a work based upon one or more preexisting works**, such as a translation.... or any other form in which work may be recast, transformed or adapted” (17 USC)
Collective Works/Compilations

- A “collective work”:
  - A number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole.

- A “compilation”:
  - collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship.
  - Includes collective works.

Chain of Title

- Ownership can be transferred via a contract
  - One sells copyright to another

- Derivative works:
  - The creator of a derivative work is added to the chain of title of the original work

- Compilations:
  - Each work maintains its original ownership
  - The compilation receives its own ownership
Licenses

- Owner of copyright can grant licenses to third parties
  - They grant permissions that otherwise would be forbidden by law
    - Creating derivative works
  - Example:
    - EULAs
    - Software contracts
    - Free and Open Source Licenses
- Typically, a licensor would negotiate a license from the licensee under a contract

Open Source Software

- Software licensed under an Open Source License
- Open Source Licenses
  - Created to simplify software reuse
  - Examples:
    - BSD family
    - GPL family
    - Even Microsoft has open source licenses
      - Microsoft Public License and
      - Microsoft Reciprocal License
F/OSS Licenses

- Like any other license, it grants some rights on the work to the licensee
  - In exchange, licensee should abide to some requirements
- Usually licensor does not know its licensees
- Many different licenses:
  - Open Source Initiative approved licenses (OSI)
  - Free Software Foundation Licenses
  - Many others

Types of F/OSS Licenses

- Academic or Permissive:
  - they allow the licensee a wide range of rights, including creating proprietary derivative works:
  - BSD, MIT, Apache
- Reciprocal:
  - in return you should release any derivative works under the same license:
  - GPL, MPL
Lex Hacketoria*

- Like Lex Mercatoria, it goes beyond the current law
- Hackers have created a code of conduct:
  - Those who don't abide are chided
  - Lawsuits are not always practical
  - Law might not support this code
- Examples:
  - Is linking a derivative work?
    - **Lex Hacketoria** says yes!
    - Many high profile lawyers (e.g. Lawrence Rosen) argue that dynamic linking should not

*concept developed by Richard Fontana, OSS Counsel at RedHat

Daniel M. German (dmg)  http://turingmachine.org
Research in Licensing Issues

- Lawyers are risk managers
- Software engineers are builders
- Licensing (and other IP related issues) has become core to the practice of Soft Eng.
- Open source has emphasized the need for this are
  - Legal Compliance in Software Development

The Big Goal

- The long term goal is to incorporate software licensing concerns of reusing licensed components into the software development processes.
  - From the point of view of:
    - The Licensor: how to license the software
    - The licensee: how does licensing affect the reuse of software components?
What does it mean?

- Understanding the problem
  - Doing Empirical Studies
  - Creating models
  - Building tools
- Ultimately helping the lawyers and the software engineers manage more easily the licensing concerns associated with component reuse.
  - **Legal Compliance**

Architectural Problems

- What components are used in this system?
- How do they interconnect?
  - Do they create a compilation?
  - Do they create a derivative work?
- What constraints do they impose onto the system?
- Can the system be re-architected to change its licensing constraints?
License Incompatibility Problem

- Can I combine components under two different licenses?
  - If yes, what should I do to comply to all the licenses requirements?
- Example:
  - GPLv2 does not allow imposition of any further restrictions.
    - You can't combined BSD-4 clauses
  - If not, are there work arounds?

License Integration Patterns: Dealing with Licenses Mismatches in Component-Based Development (German et al ICSE'09)

- A formalization of licenses in the context of component reuse
- The specific grant to be able to integrate a component C in a software system S depends on two factors:
  1. Whether S is a derivative work of S,
     - depends on the type of interconnection of C with the rest of S;
  2. the rights required to use, and potentially distribute C

Daniel M. German (dmg)  http://turingmachine.org
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>License</td>
<td>Exception</td>
<td>To allow a particular use by expanding the terms of the license in an</td>
</tr>
<tr>
<td></td>
<td></td>
<td>addendum, without modifying the text of the license itself.</td>
</tr>
<tr>
<td></td>
<td>Disjunctive</td>
<td>To give the option to the licensor to choose one of several licenses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that will best suit her purpose.</td>
</tr>
<tr>
<td></td>
<td>Clarification</td>
<td>Give an interpretation of contentious or ambiguous parts of the license.</td>
</tr>
<tr>
<td></td>
<td>Permit Relicensing</td>
<td>Allow the derivative work to be licensed under a different license than</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the one under which the product is made available.</td>
</tr>
<tr>
<td></td>
<td>Add-on</td>
<td>Allow modules under a non-compatible license to extend the functionality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the product via a well-defined API.</td>
</tr>
<tr>
<td></td>
<td>Indirect License</td>
<td>A product indicates that its license will be the same as another one</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and does not explicitly states one.</td>
</tr>
<tr>
<td></td>
<td>Different parts, different</td>
<td>Provide different parts or features of the system under different</td>
</tr>
<tr>
<td></td>
<td>licenses</td>
<td>licenses.</td>
</tr>
<tr>
<td>License</td>
<td>Patch</td>
<td>Issue a patch and let the user create the derivative work by applying</td>
</tr>
<tr>
<td></td>
<td></td>
<td>it to a given product.</td>
</tr>
<tr>
<td></td>
<td>Component with Compatible</td>
<td>Find a component that can be licensed in a manner that is compatible</td>
</tr>
<tr>
<td></td>
<td>License</td>
<td>with the intended use.</td>
</tr>
<tr>
<td></td>
<td>Create Compilation</td>
<td>Make sure the product is considered a compilation of the component.</td>
</tr>
<tr>
<td></td>
<td>Ask for exception</td>
<td>Request the licensor to give you an exception to one or more conditions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>imposed by the license. Results in the Exception Pattern, above.</td>
</tr>
<tr>
<td></td>
<td>Ask for clarification</td>
<td>Request the licensor to clarify her interpretation of any ambiguous or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contentious parts of the license. Results in the Clarification Pattern,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>above.</td>
</tr>
</tbody>
</table>

Table 2. System of Patterns used to address the license-mismatch problem.

What is the license of a Component?

- File is the minimal licensing unit
- Software components are collections of files
  - Not always all files in a “source” archive are used to create the binary
  - In Fedora 15, in only 25% packages contained files under one license (di Penta et al ICPC, 2010)
- Sometimes they are many
  - Linux contains files under than 30 licenses
- Question: what is the license of a component?
License of a File

- License identification problem
  - Given a file, under what license is it?
  - Not trivial (German et al, ASE 2010)

<table>
<thead>
<tr>
<th>Type of Issue</th>
<th>Challenge</th>
</tr>
</thead>
</table>
| Finding the license statement | F1. License statements are usually mixed with other text  
F2. Files might reference another file where the license is located  
F3. Files might contain multiple licenses |
| Language related | L1. Licensing statements contain spelling errors  
L2. A given license is referred in different ways  
L3. Licensor change the spelling/grammar of the license statement |
| License customization | C1. Several licenses must be customized when used  
C2. Licensor modify, add or remove conditions to well known licenses  
C3. Licensor modify licenses for various intents |

Table 2: Major challenges of license identification.

Ninka (German et al ASE‘10)

- Lightweight license identification tool
  - Only applicable to source code files
  - Capable of identifying 200 different licenses
  - Its goal is to minimize errors in the detection
    - In experiments, capable of identifying 85-80% licenses in files
    - Most likely error: not able to identify all licenses in a file
1 Copyright (C) 2006 Apple Computer, Inc. All rights reserved.
2 Copyright (C) 2006 Michael Emmel mike.emmel@gmail.com Copyright[...]
3 Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
4 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
5 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
6 THIS SOFTWARE IS PROVIDED BY APPLE COMPUTER, INC. "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL APPLE COMPUTER, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
7 IN NO EVENT SHALL APPLE COMPUTER, INC. OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Daniel M. German (dmg) http://turingmachine.org
.sentences are matched

1 <AllRights, <Copyright (C) 2006 Apple Computer, Inc.>>
2 <BSDpre, <<>>
3 <BSDcondSource, <above>>
4 <BSDcondBinary, <>>
5 <BSDasIs, <APPLE COMPUTER, INC.>>
6 <BSDWarr, <APPLE COMPUTER, INC. OR CONTRIBUTORS>>

which translates to a BSD-2 license.

Provenance Discovery

- Reuse is not always easy to identify
- Cut&Paste reuse
  - At the file level
  - At the component level
  - At statements level
- Malicious vs Legally allowed copying
  - Copyright violations
  - License violations
Provenance Discovery

- Copyright violations: *Who are you?*
  - Is this code derived from another?
- License violations: *Does your mother know you are here?*
  - Do we have a license to reuse the code?

Binary Provenance

- Frequently dependencies are managed by embedding its binary
  - Very common in Java
- Two problems:
  - What is the license of the binary?
    - Di Penta et al MSR'09
  - What is the source of the binary?
    - Davis et al MSR'10
Software Bertillionage

- Finding the source of a binary is a hard problem
- Corpus needs to be large to be useful:
  - Maven2 is almost 300GB of compressed Java
- Bertillionage:
  - Narrow the search space using a fast method
  - Then other more expensive methods can be used (including manual analysis)
- Examples:
  - Java: using type signatures of Java methods (Davis et al, MSR’11)
  - C/C++: using strings present in binaries (Hemel et al MSR’11)

Software Package Data Exchange (SPDX)

- Standardizes the exchange of licensing information across the supply chain:
  - What components are used to create a given product?
  - Where do they come from?
  - What are their licenses?
  - Ideally, every F/OSS component should have its attached SPDX description
  - Version 1.0 was just released (Aug 2011)
Summary

- Copyright and software licensing are affecting the way software is created
- Areas of research
  - Impact of IP on Software Architectures
  - License identification
  - Component identification/Provenance discovery
  - Many others (wait for second part of TechBrief.)

Daniel M. German (dmg)  http://turingmachine.org