The Innovations of Open Source

Dirk Riehle
Friedrich-Alexander University Erlangen-Nürnberg

2019-01-28 – INF-KOLL – Universität Hamburg
Professorship of Open Source Software

- **Professor of Computer Science**
  - For software engineering and open source software
  - At the computer science department of the engineering faculty

- **Previously held research positions at ...**
  - SAP Labs (Silicon Valley) leading the open source research group
  - UBS (Swiss Bank, Zurich) leading the software engineering group

- **Previously worked in development at ...**
  - Skyva Inc. (supply chain software, Boston) as software architect
  - Bayave GmbH (on-demand business software, Berlin) as CTO
Professorship of Open Source Software

- At the computer science department
  - Also teaches in information systems at FAU
  - Led by Prof. Dr. Dirk Riehle, M.B.A.

- Core research and teaching areas
  - Open source software
    - Governance and license compliance
    - Open source strategies
    - Open source business models
  - Inner source software development
    - Program management, project management
    - Quality assurance and security
    - Transfer pricing and intellectual property
  - Artificial intelligence techniques in applications
  - One-off projects leading to startups
The Innovations of Open Source

- Content
  - Legal innovation
  - Process innovation
  - Tool innovation
  - Business model innovation

- Structure
  - Innovation
  - Industry problems
  - Research challenges
I. Legal Innovation
Open Source Software (Legal Definition)

- Open source software is software whose license fulfills
  - these ten requirements [https://opensource.org/osd](https://opensource.org/osd); it must
    - not restrict redistribution
    - require inclusion of source code
    - allow modifications and derived works
    - preserve the integrity of the author's source code
    - not discriminate against persons or groups
    - not discriminate against fields of endeavor
    - not require additional license execution
    - not be specific to a product
    - not restrict other software distributed alongside
    - be technology neutral

- and has been approved by the Open Source Initiative
Copyleft ("Reciprocal") License

- Copyleft is a copyright-based strategy to ensure original rights cannot be curtailed when passed from licensor to licensee.
- The licensee of copyleft-licensed software will have to use the same copyleft license (or later) when passing on the software.
Primary Benefits of Using Open Source

- **Better**
  - Open source components can be of high quality

- **Faster**
  - Open source components are immediately available

- **Cheaper**
  - Open source components are free (no license fee)
Secondary Benefits

- Open source is open to inspect, modify
  - Faster: Users can help themselves, fix bugs
  - Faster: Users can extend the software, develop new features

- Most open source has no or little vendor lock-in
  - Faster: Innovation cannot be blocked by one company
  - Cheaper: Competition keeps service prices low

- Open source components are compatible
  - Faster, better: With standards (as reference implementations)
  - Faster, better: With platforms (as de-facto implementations)
The MIT License (Template)

Copyright <YEAR> <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
The Software Supply Chain

- Software vendor
  - Supplier tier 1
    - closed source comp #24
    - open source proj #5332
    - open source proj #152
  - Supplier tier n
    - closed source comp #897
    - open source proj #832
    - open source proj #21632

- license from supplier
- include as library
- copy code from web
- whatever they are up to
- who knows
- who knows
The Iceberg Under the Water Line
Creating the Bill of Materials

- Information retrieval challenge
  - License scanning
  - Source code analysis

- Complications to challenge
  - Copy and paste
  - Lost history

- Industry-wide effort
  - To add meta-data
  - Standardize
Maintaining a Product (Architecture) Model

- A novel model of a product’s code architecture to
  - Derive license conflicts, resulting license
  - Map and assess security vulnerabilities
  - Remain aware of export restrictions
II. Process Innovation
Open source enables a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.” [O18]
Quality Assurance in Open Source

- **Committer**
  - Formally: Has commit (write) rights
  - Performs bulk of work; quality assurance

- **Contributor**
  - Provides small features, bug fixes
  - Submits patches (no commit rights)
  - Knows and uses software
  - Helps with comments, feedback

- **User**
  - explicit promotion
  - implicit promotion
The Three Principles of Open Collaboration

1. **Egalitarian** participation
2. **Meritocratic** decision-making
3. **Self-organizing** processes
Principles in Comparison [R+09]

- **Traditional Work**
  - Hierarchical project assignment
    - Closed and hidden silos
    - Assigned to project
  - Status-oriented decision making
    - Public + private discussions
    - Hierarchical status decides
  - Predefined processes
    - Prescribed process
    - Prescribed jobs

- **Open Collaboration**
  - Egalitarian participation
    - Open for contribution
    - Everyone can contribute
  - Meritocratic decision-making
    - Public discussion process
    - Decisions based on merit
  - Self-organizing processes
    - People find their process
    - People find their project

The Four Practices of Open Communication [R15]

1. Public
2. Written
3. Complete
4. Archived

Open Source is Scale-free (Hypothesis)
Open Source Process Research Challenges

- Understanding open source communities
- Understanding open source engineering
Contributing to Open Source Components

• Benefits
  • Reducing / relieving a maintenance burden
  • Managing your open source dependencies

• Downsides
  • Potential loss of intellectual property
  • Revealing of important information
Definition of Inner Source

Inner source refers to the application of the open source approach and benefits to developers within the corporate environment [...]”

What Does That Mean?

Open all artifacts!
Welcome visitors!
Reported Benefits of Inner Source [CR17]

- **Higher quality (code) components**
  - More users, more eyes, more perspectives: Less bugs faster
    
    → Ultimately, higher product quality

- **Improved knowledge sharing**
  - Collaboration across org. boundaries: More weak network ties
    
    → More innovation

- **Higher employee satisfaction**
  - More self-determination, broader recognition
    
    → More attractive employer, better employee retention

- **Faster, better, cheaper software development**

Example Practitioners of Inner Source
Example Scenarios of Inner Source Collaboration

- Bug fix
- Refactoring
- New component
The separation of product units as profit centers from a platform organization as a cost center, leads to delayed deliveries, increased defect rate, and redundant software components.

Five Years Later, Revisiting One Case … [C+18]

Platform of reusable assets – but no dedicated platform organization
Research Challenges for Inner Source

- Psychological resistance
- Programs and metrics
- Transfer pricing
III. Tool Innovation
Distributed Version Control

• Born out of the needs of the open source community
  • Examples: git, hg

• Efficiency of distributed vs. centralized version control
  • Developers voted with their feet
Software Forges

• Software Forge
  • A web-based tools platform for managing and performing projects
  • With the added key purpose of matchmaking between seekers and suppliers
  • Pioneered social networking, skills-based discovery, and lucky chance on the web
IV. Business Model Innovation
Evolution of Open Source Projects

User consortia

Single vendor (“commercial”) open source firms

Developer foundations

Distributor firms

Service and support firms

Community projects

<table>
<thead>
<tr>
<th>Year</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>1984</td>
<td>GNU Emacs</td>
</tr>
<tr>
<td>1987</td>
<td>GCC</td>
</tr>
<tr>
<td>1989</td>
<td>Cygnus Solutions</td>
</tr>
<tr>
<td>1991</td>
<td>Debian</td>
</tr>
<tr>
<td>1992</td>
<td>Suse</td>
</tr>
<tr>
<td>1993</td>
<td>Linux kernel</td>
</tr>
<tr>
<td>1994</td>
<td>Red Hat</td>
</tr>
<tr>
<td>1995</td>
<td>MySQL</td>
</tr>
<tr>
<td>1996</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>1999</td>
<td>Apache Software Foundation</td>
</tr>
<tr>
<td>2002</td>
<td>Univention</td>
</tr>
<tr>
<td>2004</td>
<td>SugarCRM, Jaspersoft, Hyperic, ...</td>
</tr>
<tr>
<td>2005</td>
<td>Kuali Foundation</td>
</tr>
<tr>
<td>2007</td>
<td>Linux Foundation</td>
</tr>
<tr>
<td>2009</td>
<td>GenIVI Alliance</td>
</tr>
<tr>
<td>2011</td>
<td>Hortonworks</td>
</tr>
</tbody>
</table>

The Innovations of Open Source
© 2019 Dirk Riehle - All Rights Reserved
<table>
<thead>
<tr>
<th>Usage rights</th>
<th>Complementary materials</th>
<th>Self-help services</th>
</tr>
</thead>
</table>
| **Software (core product)**
  - Core software
  - Additional software (extensions + plug-ins, tools and utilities, integrations)
| **Documentaion**
| **Training materials**
| **Forums, mailing lists**
| **Help and chat agents**
| **On-line tutorials**

**Pricing of usage rights**
- **Quantity:** User, machine, time, ...
- **Duration:** Perpetual, time-limited, ...
- **Structured:** Initial license fee, regular maintenance fee

**Guarantees (“insurance”)**
- **Fitness for use, certification**
- **Indemnification**

**Pricing of guarantees**
- **By damage:** Loss of business, fines received
- **Structured:** Levels / bands, formula

**Support services**
- **Hot-line support**
- **On-site servicing**

**Pricing of support services (SLAs)**
- **By availability:** Incident-based, 9x5, 24x7
- **By quality:** First-level, second-level, third-level

**Training**
- **In-house training**
- **Off-site training**

**Pricing of training**
- **Fixed fee**
- **Per participating person**

**Consulting**
- **Technical implementation services**
- **Strategic solution consulting**

**Pricing of consulting**
- **Fixed fee**
- **Time and materials**

**Operations**
- **Provision of SaaS (managed service)**

**Pricing of operations**
- **Quantity:** Users, resources, ...
- **Duration:** Always time-limited
- **Structured:** Set-up, subscription
The Intellectual Property Rights Imperative of Single-Vendor Commercial Open Source

- Always act in such a way that you, and only you, possess the right to provide the open source project under a license of your choice.

Incentive for non-paying users to upgrade to paying customers

- Multi-licensing and an aggressive open source license (AGPLv3)

Current challenge: The cloud providers like AWS

- Quiet consensus that AGPLv3 does not apply to cloud providers

How and Why Open Source Foundations Work

Without open source

(Cost of) rest of product components

(Cost of) closed source component

With open source

Potential added revenues

(Cost of) open source component

Money spent on solution
“Open source software is strategic to Google, and naturally we hire a great number of open source developers. Someone who demonstrates their ability by contributing to open source projects shows that they are able to code in the real world in ways other developers can not readily match. It’s the ultimate referral.”

Chris DiBona [R15]

Benefits of Being an Open Source Developer

- Verifiable technical skills
- Peer-confirmed competencies
- Position of power and influence
Thank you! Questions?

dirk.riehle@fau.de – http://osr.cs.fau.de
dirk@riehle.org – http://dirkriehle.com – @dirkriehle
Credits and License

• Original version
  • © 2012-2019 Dirk Riehle, all rights reserved

• Contributions
  • ...

The Innovations of Open Source
© 2019 Dirk Riehle - All Rights Reserved