Single-Vendor Open Source Firms

Dirk Riehle
University of Erlangen / Bayave GmbH

2019
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

- Ph.D. from ETH Zurich, M.B.A. from Stanford GSB
Commercial Open Source by Intellectual Property

- **Service and support firms**
  - Simply service existing open source software
  - Don’t own any of the IP
  - Don’t attract venture capital

- **Open source distributor firms**
  - Provide a well working assembly of open source components
  - Own non-core-software IP (configuration data, regression test suites, …)
  - Can attract venture capital; can have outsize returns

- **Single-vendor open source firms**
  - Provide a traditional software product to enterprises
  - Exclusively own (key parts of) the software their business is based on
  - Can attract venture capital; can have outsize returns
Terminology 1 / 2: Business Strategies

- Dual licensing / multi-licensing
  - The practice of licensing a piece of software under two or more licenses

- Open core model (IP modularity)
  - The practice of splitting software into modules of different licenses
Terminology 2 / 2: Product Variants

- Community edition = pure open source source software (often open core)
- Commercial or enterprise edition = the commercially licensed product
Three Generations of Single-Vendor Open Source Firms

- The pioneers (199x-2002)
  - MySQL
  - Sleepycat Software (Makers of Berkeley DB)
  - Trolltech

- The second wave (2002-2008)
  - SugarCRM
  - MuleSoft
  - Jaspersoft

- The current breed (since 2008)
  - MongoDB
  - Confluent
  - Redis Labs
Why the Open Source Strategy?

• Purpose of open sourcing
  • To drive adoption (of product in market)
    – To build a large (not necessarily paying) user base from which benefits accrue

• What is not new
  • Revenue sources

• What is new
  • Everything else (changes)
Revenue Sources

- **Whole product**
  - **Basic product**
    - Usage rights
      - Core product (software)
      - Complementary materials
      - Self-help services
  - Guarantees ("insurance")
  - Support services

- Training
- Consulting
- Operations
Commercial Open Source Sales Process Illustrated

1. Market product
2. Provide open source
3. Track users
4. Connect users of organization
5. Generate leads from behavior
6. Identify champion
7. Make sales call
8. Engage in comp. sales
9. Win Sale!
Structure product and services so that you

1. Maximize conversion to paying customer
2. While benefiting from user community
3. And keeping the competition at bay
Intellectual Property Management

- **Intellectual property rights imperative** (of single-vendor 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.” [1]

- Use contributor agreement to maintain ownership
  - Almost all single-vendor open source firms require copyright transfer for any contributions to maintain full IP ownership [2]

- Use reciprocal license to keep competition away
  - By choice, almost all single-vendor-owned commercial open source is provided under a reciprocal license, typically the AGPLv3

[2] All you really need is a relicensing right though
Use contributor license agreement

**Problem:** A third party submitted a pull request, but you need to maintain copyright to your intellectual property.

**Context:** The component being submitted to counts as commercial IP. Accepting a pull request without a copyright transfer or at least a re-licensing rights agreement will dilute your ownership to this IP. Over time, you could lose your ability to change licenses, leading to loss of flexibility, and, ultimately, the ability to generate revenue from your product.

**Solution:** Accept pull request only after the submitter signed a contributor license agreement (CLA), sometimes also copyright assignment. You need usage and re-licensing rights from the submitter. Example CLAs are the ...
### Recent Licensing Changes

<table>
<thead>
<tr>
<th>Who?</th>
<th>What?</th>
<th>When?</th>
<th>From License</th>
<th>To License</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mongodb</strong></td>
<td>Core</td>
<td>2018</td>
<td>AGPLv3</td>
<td>SSPL</td>
</tr>
<tr>
<td></td>
<td>Extensions</td>
<td>2019</td>
<td>Apache 2.0</td>
<td>CCL</td>
</tr>
<tr>
<td><strong>confluent</strong></td>
<td>Extensions</td>
<td>2019</td>
<td>AGPLv3</td>
<td>RSAL</td>
</tr>
<tr>
<td><strong>redislabs</strong></td>
<td>Extensions</td>
<td>2019</td>
<td>Commercial</td>
<td>Apache 2.0</td>
</tr>
<tr>
<td><strong>yugabyteDB</strong></td>
<td>Extensions</td>
<td>2019</td>
<td>Commercial</td>
<td>Apache 2.0</td>
</tr>
</tbody>
</table>
Redis After Licensing Change (AGPLv3 to RSAL) [1]

Why The Licensing Change? Maximize Conversion

- **SugarCRM**, **Chef**, ...
- **MongoDB**, **Redis**, ...
- **Anaconda**, **Icinga**, ...
- **Gatsby**, **Rasa**, ...

End-user (LoB, IT) vs. Developer

- **Maturity**
- **Growth**

Market
<table>
<thead>
<tr>
<th>Component</th>
<th>From-License</th>
<th>To-License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Community server</td>
<td>AGPLv3 (and commercial)</td>
<td>SSPL (and commercial)</td>
</tr>
<tr>
<td>Connectors and drivers</td>
<td>Apache 2.0 (and commercial)</td>
<td>Apache 2.0 (and commercial)</td>
</tr>
<tr>
<td>Cloud management</td>
<td>Commercial (only)</td>
<td>Commercial (only)</td>
</tr>
</tbody>
</table>
The Single-Vendor Commercial Open Source Playbook

Define product architecture
- Problem
- Define conversion triggers
- Solution

Define product components
- Classify product components
- Define conversion path
- Define conversion triggers
- Use LGPL for core and MIT for entire project

Product management
- Define component features
- Define component license
- Review component dependencies
- Define market segments
- Define market
- Define product architecture
- Define component features
- Define component license
- Relate segments to features
- Classify product components
- Define product components
Thank you! Questions?

dirk.riehle@fau.de – http://osr.cs.fau.de

dirk@riehle.org – http://dirkriehle.com – @dirkriehle