Profiting Even More from Open Source

MICHAEL WEISS, Carleton University

The patterns in this paper continue the description of open source business models started in Weiss [2010]. Many businesses now incorporate open source, either leveraging open source to develop new products or starting their own open source projects and building their products and services around their open source offerings. The patterns in this paper aim to provide entrepreneurs, managers and students of business models with a language for creating new business models around open source, or for incorporating open source into existing business models. The patterns in this paper capture more advanced patterns such as the use of an open source foundation.

Categories and Subject Descriptors: D.2.11 [**Software Architectures**] Patterns; K.6.0 [**Management of Computing and Information Systems**] Economics; D.2.9 [**Management**] Copyrights

General Terms: Design, Economics, Management

Additional Key Words and Phrases: Patterns, open source, business models

1. INTRODUCTION

This paper is part of a larger work on open source patterns [Weiss 2009; 2010]. The patterns in this paper continue the description of open source business models, and capture more advanced patterns such as the use of an open source foundation. How companies can make money from open source is a frequently debated topic. However, since the origins of open source, our understanding of open source business models has significantly evolved. Many businesses now incorporate open source, either leveraging open source to develop new products or starting their own open source projects and building their products and services around their open source offerings.

Traditionally, software companies made their money through software licenses and services that complement the software including customization, training, support, and maintenance, or the sale of hardware that runs the software (eg routers). Open source software lacks the strong intellectual property (IP) protection of traditional software. Hence, strategies for capturing value from open source software place more emphasis on complementing the software with other products or services, whereas licensing still plays an important, however different, role.

The audience for these patterns are entrepreneurs who seek opportunities to create new products based on open source, managers in established companies who need to compete with new market entrants that leverage open source, and students of business models. The patterns aim to provide them with a language for creating new business models around open source, or for incorporating open source into existing business models.

Author's address: M. Weiss, Institute of Technology Innovation Management and Commercialization, Carleton University, 1125 Colonel By Dr, Ottawa, ON K1S 5B6, Canada; email: michael_weiss@carleton.ca

EuroPLoP '11, July 13-17, 2011, Irsee Monastery, Bavaria, Germany

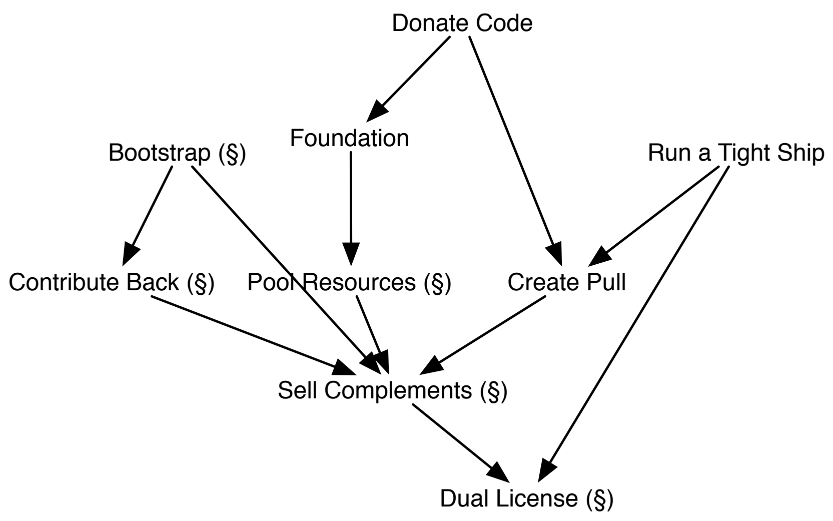

Fig. 1. Patterns for profiting from open source.

2. OVERVIEW OF THE PATTERNS

A map of the patterns showing their relationships is shown in Figure 1. Links between patterns X and Y should be interpreted as "after pattern X you may also use pattern Y". A paragraph symbol (§) after a pattern name means that this pattern is (will be) described elsewhere. Thumbnails of these patterns can be found in the Appendix.

When you create your own open source project, a key decision is whether or not you should RUN A TIGHT SHIP by keeping control of the direction of the open source project. Typically, you would contribute most of the resources for developing the software in this case. However, there is a benefit: as owner of the code you can DUAL LICENSE [Link 2010] it, offering a free and a for-charge version, which is a special case of sell complements.

A company can also DONATE CODE to open source that was proprietary but no longer provides you with a competitive advantage. This can be used to extend the lifespan of your software by giving others a platform to use, and to create demand for complementary products and services that you offer.

Contributing to open source code is a way of reducing your marketing expenses. In a typical software-based product only a small portion of the cost is directly related to development. Allowing customers to download your software for free CREATES PULL for your product and its complements. Complements are products or services (such as training or hardware) that add value to the open source product, and which you can charge for.

To gain wide-spread adoption of code that you donate and get other companies (including your competitors) to contribute you need to give up ownership over the code by setting up a foundation and assigning ownership of the code to it. A FOUNDATION builds trust and facilitates collaboration among different companies. It allows companies to pool resources with other companies [Weiss 2010] to jointly create software that each company builds on in their own products. Every company that contributes to the pool can leverage the resources to SELL COMPLEMENTS.

3. RUN A TIGHT SHIP

"You can make sure you own the copyrights by either writing all your own code or obtaining third-party licenses. In fact at SleepyCat and MySQL, all code development is done in-house. External development can be done, but the licensor needs to ensure that copyrights are assigned from contributors before their contribution is added in." [Albert 2004]

3.1 Context

You are creating your own commercial open source project.

3.2 Problem

How do you keep control of the project's direction?

3.3 Forces

Opening up a project encourages external participants to join, and allows you to benefit from their contributions. However, once you allow others to contribute to your project you may lose control over its direction.

External contributors provide you with skills and resources that allow you to scale your project or to complete it sooner. However, they have their own goals and may want to influence the project to align with those goals.

3.4 Solution

Maintain full ownership of the code.

Most projects start out like that. The project initiator is often also the sole owner of the code. Rarely, projects are set up as shared projects from the get-go. The real decision is the one you face when the project grows.

As projects evolve, you need to decide whether to keep in control of the project by maintaining full ownership of the code, or to share control with others. When you decide to keep ownership of the code, you either need to write the code yourself or get others to ASSIGN COPYRIGHT [Link 2010] to the code that they contribute to you.

3.5 Consequences

You need to contribute most of the project resources, since there are less incentives for others to contribute.

Keeping tight control over a project limits the growth of the project.

3.6 Examples

OpenOffice is an example of a project that changed its status from a tightly-controlled project to an community-based project multiple times. It started as a closed source product (StarOffice), which was bought by Sun and subsequently released as open source (OpenOffice). When Sun was acquired by Oracle, the project was again put under tight control with Oracle aiming to sell a commercial version. Recently, Oracle, once again, released control over the project, but not before the project was forked by the community in the form of the LibreOffice project.

3.7 Related patterns

This pattern is an alternative to DONATE CODE.

Since you own the code, you can license it to different groups of users under different licenses as in DUAL LICENSE [Link 2010]. Thus, you can give away a restricted, open source version of your code, and charge for a commercial version that has no restrictions.

Making your software available as open source creates pull for your product and any complements that you sell.

3.8 Sources

Literature [West and Gallagher 2006] and the author's observation.

4. DONATE CODE

IBM is donating half a million lines of code from its Cloudscape database to the open source community, care of the Apache Software Foundation. IBM said the move would spur the development of Java applications, [...]. The move also suggests IBM is not taking the open source street cred it has gained from its fight with SCO for granted. [Scherriff 2004]

4.1 Context

You have proprietary code that does not (or no longer) give you a competitive advantage, and you consider open sourcing it.

4.2 Problem

How do you extend the lifespan of your proprietary code?

4.3 Forces

Continuing to maintain the code is difficult to justify in light of decreasing revenue, and one solution is to terminate the project. However, the project might have value for somebody else, and even create new revenue for you.

Opening up a project encourages external participants to join, and allows you to benefit from their contributions.

4.4 Solution

Relinquish control over the code by releasing it under a flexible license that allows others to build on it easily.

An example of a flexible license is the Eclipse Public License (EPL). It allows anyone to include existing components licensed under the EPL in their products.

4.5 Consequences

Donating code extends the lifespan of your software.

It helps you establish the software as a standard and create demand for complementary products or upgrades that you can sell.

4.6 Examples

In 2001, IBM donated the initial code for the Eclipse project. Eclipse is an open source open source development platform. One way IBM benefits from donating this codebase is that its platform gained more wide-spread adoption. The other is that is can sell products built on top of the Eclipse platform, such as WebSphere.

In 2004, IBM also donated the code from its Cloudscape database to the Apache Derby project. It continues to offer commercial support (e.g., additional features) for the Derby project under the Cloudscape name.

4.7 Related patterns

Making your software available as open source CREATES PULL for your product and any complements you sell.

To get other companies to contribute you need to give up ownership over the code by setting up a foundation and assigning ownership of the code to it. Creating a FOUNDATION builds trust and facilitates collaboration among different companies.

4.8 Sources

Literature [West and Gallagher 2006] and the author's observation.

5. CREATE PULL

Since Compiere is freely available for download, anyone can install the software, try it, and see if they want to use it in their environment. [...] Having an open source business model can generate a level of awareness that might otherwise cost a substantial amount of money to achieve through trade shows, advertising, etc. [Cingari 2007]

5.1 Context

You RUN A TIGHT SHIP or DONATE CODE by offering open source.

5.2 Problem

How do you market your open source product?

5.3 Forces

In a typical software-based product only a small portion of the cost is directly related to development. Marketing can account for up to 70% of the cost of the product [Watson et al. 2005]. Companies largely depend on their sales staff to convince customers of the benefits of their products.

If your software is available under a DUAL LICENSE, companies want to evaluate your software before buying a commercial license.

5.4 Solution

Make it easy to access your product and leverage distributors, partners, as well as community members to market your product.

Open source software can be downloaded for free. Customers can try the software on their own. Some of these customers will convert into paying customers (eg paying for a commercial license or demanding customizations to the product). The larger the number of users of the product, the more contributors and customers.

Your product will also be marketed by companies who include your product in their distributions, partners who tailor your product by adding custom features or integrating with other products, or who SELL COMPLEMENTS to your product (such as special hardware), and community members (such as bloggers) who are excited about the product and promote it without being on your payroll.

Marketing an open source product cannot be achieved by pull only. It is also important that you create an attractive and informative community forum where potential developers and customers can learn more about the project and get their questions answered.

5.5 Consequences

Offering your product as open source is a way of reducing your marketing expenses. It enables its viral distribution.

Companies who want to integrate your software with their products, can trial the software without the usual inconvenience of a time-limited evaluation license before buying a license.

5.6 Examples

JBOSS used this pattern extensively by making their code of their application server available for download. Potential users of the server could evaluate the server without having to contact JBOSS or request a special evaluation license (sometimes a costly process on its own and detractor for using the software), as well as examine the implementation. Users could make a thorough evaluation of the software without investing more than their time.

The Compiere example (quote) illustrates that the fact even of making code available as open source makes potential users aware of the code. The product benefits from the open source ÒbrandÓ.

5.7 Related patterns

By creating demand for your product you can SELL COMPLEMENTS.

5.8 Sources

Literature [Watson et al. 2005] and the author's observation.

6. FOUNDATION

Vendors can add value on top of this stack, build tools on top of Eclipse and the value of the Eclipse ecosystem will grow. [Milinkovich 2005]

6.1 Context

You have started an open source project and want to get other companies (including your competitors) to contribute.

6.2 Problem

How do you attract other companies to contribute to an open source project that you created?

6.3 Forces

If the project creator controls the project, it is difficult to motivate other companies (especially competitors) to contribute.

Other contributors will be suspicious of your motivations, if you ask them to contribute, but keep in charge of the project.

Developing an open source project on your own is costly and time-consuming. You want other companies to help.

Less than 20% of the code in a software system is really unique.

6.4 Solution

Transfer ownership of the code to an independent foundation.

The foundation manages a commons that all contributors share.

Contributors need to transfer ownership of their contributions to the foundation as in MAINTAIN A CONTRIBUTOR AGREEMENT [Link 2010]. The foundation is the legal owner of the open source project. It facilitates the interaction of the companies that contribute to the project. Typically, this includes member acquisition, project coordination, evangelism, ensuring DUE DILIGENCE [Link 2010], and the organization of events such as developer conferences.

6.5 Consequences

Creates an arms-length relationship between the creator of an open source project and the project itself.

Builds trust and facilitates collaboration among the contributors.

Reduces your resource commitment to the project, while giving you access to all assets developed by the project.

You end up sharing the cost of developing the code that does not differentiate you from other companies with them. In fact, everyone will avoid creating their own solutions to common problems, and can spend their development dollars more wisely.

6.6 Examples

When Eclipse was spun out of IBM, a foundation was put in place to manage the project. While IBM still contributes many of the developers, IBM does not own the project, and many of its competitors, including Oracle and SAP contribute to and build many of their products on top of the Eclipse platform.

6.7 Related patterns

Creating a neutral foundation allows companies to POOL RESOURCES with each other, and benefit from the jointly created software.

6.8 Sources

Literature [West and Gallagher 2006] and the author's observation [Weiss 2011].

7. ACKNOWLEDGEMENTS

I thank Aliaksandr Birukou for offering to shepherd this paper, and all the participants at the EuroPLoP 2011 workshop for their comments.

In formatting these patterns I owe a tremendous amount to the format Allan Kelly has used in his own papers, which I tried to emulate.

APPENDIX

Here are short forms of the patterns not described in this paper:

BOOTSTRAP	Use existing open source components in your products to get something working quickly and keep costs low [Weiss 2010].
CONTRIBUTE BACK	Contribute back to the project in order to keep aligned with its evolution and to keep it healthy [Weiss 2010].
SELL COMPLEMENTS	Monetize an open source product by selling products or services that complement the open source product [Weiss 2010].
DUAL LICENSE	Use two licenses for your project, one which enables the use in other open source projects, and one which simplifies commercial usage in exchange for a fee [Link 2011].
POOL RESOURCES	Pool resources with other companies to develop a common stack of open source assets that the can all build on [Weiss 2010].

REFERENCES

ALBERT, P. 2004. Dual licensing: Having your cake and eating it too. www.linuxinsider.com/story/38172.html.

BIGBLUEBUTTON 2010. code.google.com/p/bigbluebutton.

CINGARI, J. 2007. Open source as a marketing strategy. blog.compiere.com/2007/01/open_source_as_.html

FITZGERALD, M. 2006. Bootstrapping 101: Use cheap web tools. *Inc.*, www.inc.com/magazine/20060701/bootstrapping-l5.html.

KELLY, A. 2008. Business patterns for product development. *EuroPLoP*.

LINK, C. 2010. Patterns for the commercial use of open source. *EuroPLoP*.

LINK, C. 2011. Patterns for the commercial use of open source: License patterns. *EuroPLoP*.

MILINKOVICH, M. 2005. quoted in: Rooney P. (2005), “Eclipse effect” will drive channel business. *CRN*, August 31, 2005.

SCHERRIFF, L. 2004. IBM gives code to Apache open sourcerers. *The Register*, http://www.theregister.co.uk/2004/08/03/ibm_open_source_gift.

WATSON, R., WYNN, D., AND BOURDEAU M.C. 2005. JBOSS: The evolution of professional open source software. *MIT Quarterly Executive*, *4(3)*, 329-341.

WEISS, M. 2009. Performance of Open Source Projects *EuroPLoP*.

WEISS, M. 2010. Profiting from Open Source *EuroPLoP*, ACM.

WEISS, M. 2011. Economics of collectives. *International Workshop on Quantitative Methods in Software Product Line Engineering* at the *International Software Product Line Conference*, 39, ACM.

WEST, J., and GALLAGHER, S. 2006. Challenges of open innovation: The paradox of firm investment in open-source software. *R&D Management*, *36(3)*, 319-331.