

Open innovation and Open source

Open innovation is a great way to acquire and share innovations with others. Yet, turning an innovation into a product requires a lot of additional functionality, as well as development and integration effort. Most of this functionality is being implemented in software. As a result the amount of software in products has risen dramatically in the past years. It is becoming harder and harder for companies to develop all this software in-house, or even to buy this software from third-party software firms. Open source software fills the gap between innovation and product.

What is Open Innovation?

The idea behind Open Innovation is that commercialization of technological breakthroughs increasingly requires close cooperation and sharing of innovations between companies, universities and other research institutions.

There are two parts to the idea. First, in this new environment of diffused knowledge, in which workers migrate from company to company (or start companies of their own), and in which information flows cheaply and fluidly over the Internet, companies' attempts to hoard intellectual property are increasingly counterproductive. Rather than seeing themselves as fortresses of knowledge, companies should swiftly inject their discoveries into the market, either through their own business units or externally via third parties. If they are slow in marketing their intellectual property, the knowledge is likely to seep out, leaving them with nothing to show for their efforts.

Secondly, with marketable knowledge available from so many sources outside the company, it makes no sense to reinvent the wheel. Innovators need to keep their eyes open, looking not only inside but also outside the company for the next revolutionary idea.

Closed innovation	Open innovation
The smart people in the field work for us.	We need to work with smart people inside and outside the company.
To profit from R&D, we must discover it, develop it, and ship it ourselves.	External R&D can create significant value: internal R&D is needed to claim some portion of that value.
If we discover it ourselves, we will get it to the market first.	We don't have to originate the research to profit from it.
The company that gets an innovation to the market first will win.	Building a better business model is better than getting to the market first.
If we create the most and the best ideas in the industry, we will win.	If we make the best use of internal and external ideas, we will win.
We should control our IP, so that our competitors don't profit from our ideas.	We should profit from others' use of our IP, and we should buy others' IP whenever it advances our business model.

Source: Henry Chesbrough, *Open Innovation*, Harvard Business School Press, 2003

Open Innovation and Open Source

Open innovation is a great way to acquire and share innovations with others. Yet, turning an innovation into a product requires a lot of additional functionality, as well as development and integration effort. Most of this functionality is being

“Open source fills the gap between innovation and product”

implemented in software. As a result the amount of software in products has risen dramatically in the past years. It is becoming harder and harder for companies to develop all this software in-house, or even to buy this software from third-party software firms. Open source software fills the gap between innovation and product.

Open source is most well-known as the model behind the successful Linux operating system and the Firefox web browser. Under the open source development model the source code to a computer program is made publicly available under a license that gives users the right to modify and redistribute the program. This right often comes with an obligation for licensees to make their modifications and improvements freely available in turn to others.

This “share and share alike” aspect of open source at first glance may seem a risk. In fact it is one of the greatest benefits of open source. The license ensures that all participants can benefit equally from the development. It creates a level playing field in which multiple companies, groups and individuals can work together to create the best software platforms for their respective innovations. There is no risk that a single company unfairly profits from the work of others. The GPL thus is a joint development agreement – royalty-free and open to all.

Those peculiar Open Source Licenses

The license conditions applicable to open source can be quite peculiar. For example, some licenses require one to release one's own software as open source software itself, if that software incorporates the open source software. If open source software under such a license is used throughout the software stack in a product, all software for that product may have to be published as open source. Other licenses require the publication of legal notices or the offering of source code to anyone who asks, which can get complicated when a company produces many products with large amounts of software.

There are over 40 different open source licenses, each with their own conditions and implications. Roughly they can be classified into these three categories:

- **Free-for-all licenses:** these licenses only require licensees to give credit to the original authors. Derivative works can be kept proprietary. Sometimes these licenses are referred to as "academic licenses". Examples are the so-called BSD and MIT licenses as well as the license used for the Apache Web server.
- **Keep-open licenses:** modifications to software under these licenses have to be made available as open source as well. Larger works incorporating such software can be kept proprietary. The GNU Lesser GPL (used for Linux

system libraries) and the Mozilla Public License (used for the Firefox Web browser) are Keep-open licenses.

- **Share-alike licenses:** when software under such a license is modified or extended, the result as a whole has to be made available as open source. The term ‘copyleft’ is sometimes used to characterize this kind of license. The most famous example is the GNU GPL, which applies e.g. to the Linux operating system. Another example is the Open Software License (OSL).

Because of the perceived risks, a company may be tempted to avoid open source software altogether. Formal open source policies often treat requests to use open source as a rare exception: “not allowed, unless”. This however is not a realistic option from a business point of view. Such a company locks itself out from all the available high-quality software and does not benefit from the reduced costs and time to market that open source software implies. In some cases it is simply not feasible for a commercial product to keep up feature-wise with open source alternatives.

“Avoiding open source is no longer an option”

Hence an open source policy should be about where and how, not if, the different types of open source can be used in a product.

Mixing open and closed components

Companies that use open source often make a purely black or white decision: either a product is open source or it is not. Much greater benefits can be obtained by using a more subtle approach. A company can use open source for certain features and use closed, in-house developed or commercially licensed software for other features if the above-mentioned license implications are properly managed.

The ultimate goal must be to ensure the product or service offers the most value. Value can be obtained through different strategies. A feature can serve to differentiate the product over competing products. Alternatively, the feature can be licensed to others for a royalty fee, for example as a software library or a chip with embedded software. Patents related to the feature can be licensed royalty-bearing as well. Open sourcing the feature increases industry adoption and reduces maintenance cost. Using existing open source for a feature means no more single-vendor lock-in.

So, which features to open source and which to keep closed? The answer comes from an old rule in economics: make the complement of your own product a commodity. A *complement* is a technology or feature that is needed to make your own product useful. For example, digital music complements portable music players. Commoditizing your complement actually stimulates demand for your own product. If digital music becomes easier to purchase online, demand for music players goes up. It is therefore in the interest of digital music player manufacturers

“Commoditize your complement”

to ensure that the widest range of digital music can be purchased online from as many shops as possible.

This rule not only applies to complete products, but also to components or features of a single product. An advanced digital television may have the ability to automatically identify and skip commercials in television programs. The ability to decode and play digital video streams is essential to make this feature useful, and so video decoding is a complement to commercial skip. If the television is positioned as differentiating because of its commercial skip, then digital video decoding software should be (or become) a commodity.

Open source software by definition is a commodity. Its license permits anyone to use, extend and distribute the software. This makes any software component that complements the differentiating parts of the product a candidate for open source. The component can be published as open source, or be replaced with an existing open source alternative. Commoditize your complement through open source.

Make or buy – or open source

To put the general rule “open source your complement” into practice, a slightly more detailed product feature classification is needed:

- **Differentiator:** these features provide added value to product. They give an ‘edge’ over the competition and provide a reason why customers want to buy the product.
- **Baseline:** these features are necessary and expected by customers. They provide value, but no *added* value to the product. For example, today no one buys a portable music player because it supports the MP3 format. On the other hand, no one would buy such a player if it does *not* support MP3.
- **Commodity:** these are hidden, uninteresting features. They are needed to make the product work but the customer doesn’t care about them.

Traditionally, the question for any product feature is “make or buy”, allowing a company to make only its unique features and to buy the remaining components from elsewhere. This classic question now becomes “make, buy or open source?” To answer this question for a particular feature, the first step is to classify it as a differentiator, a baseline feature or a commodity feature. Second, for each feature the impact of each option (make, buy or open source) should be made. The diagram in the figure to the right shows the impact of each.

	Make	Buy	Open
Differentiator			
Baseline			
Commodity			

Legal software design

While the above may seem straightforward enough in theory, the practice can be difficult. Building a software-based product is not a matter of putting components

together like LEGO® blocks. Source code can be copied and pasted, libraries can be linked or accessed through remote procedure calls. With object-oriented languages like Java and Web-accessible services the possibilities grow even larger. This may create complex interactions and dependencies between the software components. Assessing the legal implications of open source code in such a complex software stack can be a difficult task.

To ensure these implications can be properly managed, IP professionals should get involved in the product design at an early stage. When an IP assessment has to be carried out on a fully completed software stack, making changes based on IP or legal considerations can be very difficult or costly. IP and legal design goals should be formulated at the beginning of the project, so that these can be taken into account during the design and implementation of the software.

Policies regarding open source should not be presented as a prohibition but as part of a strategy to make effective use of open source. A positive attitude from the IP and legal professionals with good knowledge of software development issues is very important to get such a strategy off the ground.

Conclusion

Open Innovation is the way to go in this new environment of diffused knowledge. Companies should look for opportunities to acquire innovations from and share innovations with others. Commoditizing certain technologies, in particular by releasing them as open source, can be a good supporting strategy to ensure that new innovations become feasible. This requires a proper analysis of the value each feature brings to the product, so that the right features are open source or kept proprietary. By creating a large commons of open source, every business in the market can build upon this commons and create their own unique products or services.

And that, in the end, is what Open Innovation is all about: enlarging the very foundations of the market, creating new opportunities for every company in the industry.

About the author

Arnoud Engelfriet is a European patent attorney working at Philips Intellectual Property & Standards. As secretary of Philips' Open Source Advisory Board he coordinates the IP aspects of the use of open source software by Philips. He can be contacted at Arnoud.Engelfriet@Philips.com