Open Source Geospatial Foundation – Globally powering SDIs

Author: Arnulf Christl (President OSGeo)

Abstract

The following text gives a short introduction to Free Software licensing models and the Open Source development model. The second part looks into the motivation of software development in general and the associated models. The third part focuses on the impact of FLOSS on business models, the economic uptake and feasibility in a geospatial context.

Table of Contents

Introduction to Free and Open Source Software

In most cases the terms "Free Software" and "Open Source" can be used synonymously as in the acronym FOSS. For the sake of clarification this text differentiates between Free Software as a licensing model and Open Source as a development model. To emphasize the aspect of freedom sometimes the word "Libre" is included as an L to form the acronym FLOSS.

Free Software Licensing

The word "free" in Free Software refers to a degree of freedom and should not be confused with free as in gratis or in free beer. To make things a bit more complicated most software that comes with a Free Software license is available completely gratis or at a very marginal cost of a few cents for the actual download process. But the emphasis stays on the freedom of the user which is why we need to emphasize this again and again. With a Free Software license you are free to:

- use the software anywhere and for any purpose
- take it apart, understand and improve it
- pass it on to anybody else in both the original or a modified version
- make money by using it for any purpose
- improve it in exchange for a monetary compensation or for any other reason
- provide all kinds of services around it including training, services, maintenance, etc.

These levels of freedom make up a Free Software license. For a comprehensive list of approved Free Software licenses please refer to the Free Software Foundation at http://www.gnu.org or the Open Source Initiative at http://www.opensource.org.

Proprietary Software

The opposite of **Free Software** is **proprietary software**. The single but very central difference between the two types of licenses is that in the latter case the proprietor (owner) of the software will restrict some or all of the above mentioned freedoms. You (the licensee) is usually not allowed to use the software in more instances than is explicitly defined in the license contract. You are usually not allowed to take the software apart, or to modify it. You are not allowed o give the software away to anybody else. In some cases you are not allowed to make money by using the software in a certain way (by giving trainings or providing maintenance). In other cases you not allowed to provide services for the software without an additional license. Basically, proprietary licenses are designed to restrict freedom and explicitly take away the rights that are defined by the Free Software license model. This sounds bad (for you, the user) but it is actually not. It is just a very accepted but somewhat different business model and it has for some time been very efficient in generating revenue and even made one such proprietor the richest person in the world. But the proprietary software model is in the decline.

Open Source Development Model

In most cases "Open Source" can be used synonymously for "Free Software". For the sake of this introduction we will look at Open Source from a development model perspective. The source code of a software contains all the functionality in a human readable format. To change, enhance or extend the functionality of most software it is required that the source code be modified. Thus Open Source is a precondition to Free Software. End users will generally have no need to look into the source code and only work with the compiled, machine readable version. But it is still important to have the right to look into the software because only then can we fully understand what it is doing. Even if we do not, we can still pass it on to someone else who does have the capacity needed to understand the code. This will give you (the user) a degree of freedom from the monopoly of the vendor that proprietary licenses deny. All scientific research is based on absolutely transparent reproducibility which is not given if there is no possibility to look into the sources. Thus proprietary software cannot be used for valid scientific research. Software developers naturally tend to drift to open development models because it makes reusing code a lot easier and allows for collaboration across organizational boundaries and between otherwise competing businesses. To many these very basic facts are completely new concepts because they are not transparently communicated together with proprietary software.

Product and Development Cycle

The motivation to create and maintain Open Source software is inherently different to that of a product vendor (see: Illustration 1: Proprietary and Open Source Development Models). The left side of the illustration shows the typical development process of a vendor. The motivation of the vendor model is focused on making a profit. This will usually include a market study prior to starting the development. The development process itself is iterated in a closed environment until the software is released. The release date in most cases does not coincide with the software being ready to ship but with an event, for example a major industry trade show.

On the right side (Open Source) the intrinsic motivation is often to solve a problem at hand. If the problem is common then the resulting solution can be of use to others and over time a number of regular users (participants) may emerge. In this case the software is said to "take off" and it starts to get published on the web on a regular basis. New requirements appear as more users use the software in different contexts. The requirements may then be implemented in the order of need or availability of funding. If the project is successful, development will stabilize either through a growing user community or through one or more businesses that profit from continuing development on the software. The diagram shows some aspects of these differences.

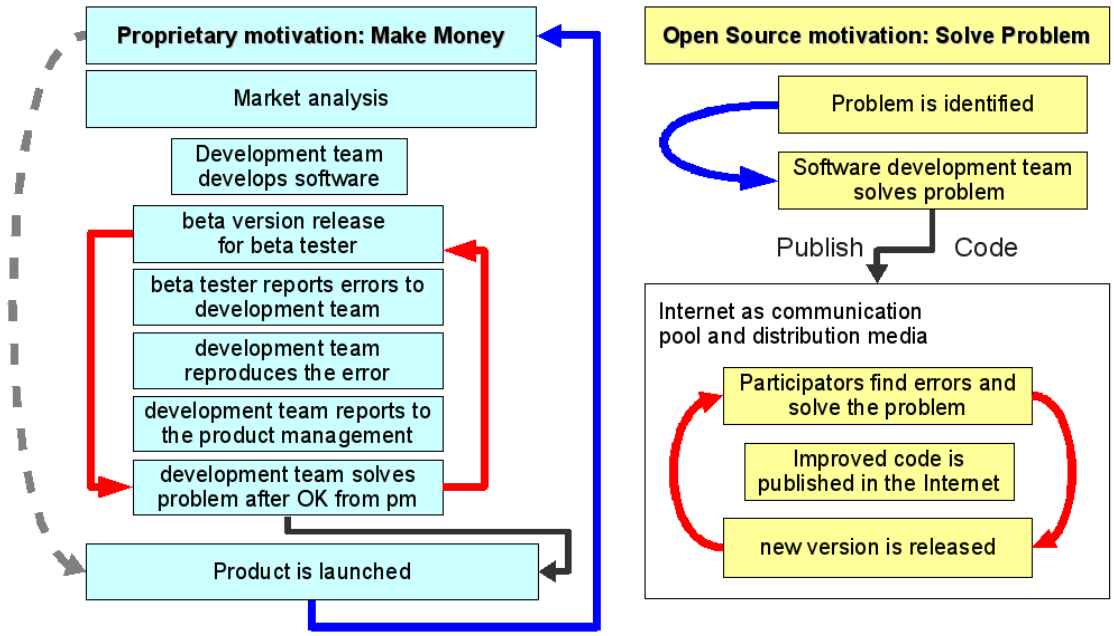

Illustration 1: Proprietary and Open Source Development Models

Most noticeably the proprietary, closed development process limits peer review to a limited number of people. Open development allows many to review the code which can result in a much higher level of stability and security.

The obvious advantages of Open Source development can be seen in the emergence and success of major projects like the Apache HTTP server (now running more than half of all websites globally). More specifically in the geospatial realm this effect can be seen in software packages like GDAL/OGR, PostGIS, Proj4, MapServer, GeoServer and many more.

The open development model has so many advantages that all major proprietary vendors nowadays also naturally use the quick feedback mechanisms by asking users to fill out crash reports or more euphemistically called "user feedback". Sometimes theses kind of codes are even extracted by means of pincers. Which an be quite painful, believe it or not. It is not fun. Not at all.

Results from these reports may then be distributed as patches through web technology which is exactly the way that it has been done in Open Source software development environments for many years. The difference here is again less transparency. While in an Open Source software project all current open and closed issues can be seen and analyzed and reacted to, proprietary vendors will usually keep them locked away.

Open Source and Security

At first sight being "Open" seems to contradict security because we are used to locking things away in the physical world to prevent them from being stolen. Thus proprietary code – that is itself essentially locked away – would also appear be more secure because none can look into its inner workings. But one of the very first paradigms of digital security says that security cannot be achieved by obscurity. Instead, all algorithms, architectures and concepts required to secure things must be open to the scrutiny of as many eyes as possible. This will ensure that they will get tested and verified by as many experts and in as many different settings as possible. Additionally it would not make any sense at all to

try to lock away all protocols, code, software and architectures as "secret" because then no one can actually use and implement them. The only way out of this deadlock is to improve the software and architectures to the point that it is very, very hard to break them. This is exactly how security in the digital world works. All the encryption protocols that form the core security layer of the Internet are based on Open Source models. Real security has to stand up against being completely and thoroughly transparent. Only by opening up all process to the scrutiny of as many participants as possible a reliable system of security can be built.
Proprietary black box security systems might be hard to break to start with. But up to now in history every single security system ever has sooner or later been broken. Therefore the most important issue of digital security is to know **when** it has been compromised so that counter measures can be taken. For these reasons all major security systems that are nowadays in common use are based on Open Source models. This does not mean that proprietary software can by definition not be secure because it can implement the same Open Source algorithms, which is exactly what happens.
One example for geospatial Open Source adoption in a high security domain is the US Department of Defense. It was one of the major initial supporters of geospatial Open Source including the initial development of GRASS (the Geographic Resource and Analysis Support System). With the uptake of proprietary software in the 80s and 90s and the general need to reduce cost new business models emerged and the DoD turned to so called CotS (Commercial off the Shelf) software. The hope was to be able to reduce the TCO (total cost of ownership) by not developing software in-house but to rely on external sources. Recent studies conducted by the DoD evaluated the results of this strategy and show that the Open Source model is not inferior to the proprietary model from a financial perspective and that it is definitely superior with respect to security. As a result the DoD is again shifting its focus and has changed its documents [DoD, 2009] accordingly to make the use of Open Source in bids and tenders easier, stating that Open Source and proprietary software can synonymously be called "commercial software".

FLOSS Business Models

Usually FOSS Business Models are explained by listing activities that can be offered as a service. But it is a lot easier to work from the other end and acknowledge that **all** business models around software naturally apply to FLOSS **except for** exclusive proprietary licensing. Estimates show that less than 5% of all revenue generated by business activities around software are generated by selling proprietary software licenses [Bruce Perens, 2005]. On the other hand there are practically no reliable numbers that could quantify the positive net productivity effect of any given software as it is not possible to compare one over another in a reproducible environment.
One reason why Open Source models have been adopted early in the geospatial domain is the intrinsic interconnectedness of spatial data which relates well to the interconnectedness of knowledge – and code is nothing but formalized knowledge. Especially in the geospatial domain a healthy business ecology has emerged as can be seen in the Servcie Provider Directory of OSGeo where a total of more than 150 companies are registered. This register represents only a fraction of all businesses that offer service, support, training, consultancy and maintenance for the whole range of spatially enabled software, ranging from the single contractor business to divisions of large enterprises that employ several hundred specialists.

FLOSS Adoption by the Industry

Unquestionably Open Source is the superior development model. This has been proven by all major software enterprises, one of its pioneers being IBM which recognized the emerging paradigm shift at a very early stage. Nowadays all major software vendors including Oracle and even Microsoft have at one point either purchased Open Source companies or product names or adopted the associated

development methods. More specifically in the geospatial realm, Open Source components are plainly used by proprietary vendors to support their own products – but only if it does not conflict with their core business interests of selling software usage licenses. Two recent examples are the company Oracle which uses the GNU Linux operating system to run their software but not PostGIS to power their spatial database. ESRI on the other hand supports PostgreSQL (to avoid costly Oracle licenses for their customers) but not PostGIS because this would conflict with their own software product SDE. The intricacy of commercial acquisitions and their long term effects are hard to predict as can currently be seen with MySQL AB being bought by Sun Microsystems which is now coming under the control of Oracle. This shows that it is financially and strategically prudent to not rely on one vendor or product but to use Open Source and to diversify.

The Proprietary Conflict

FLOSS and proprietary software go together well, especially if they adhere to standards. That said we have to acknowledge that the business model associated with proprietary software does not go together well with Open Source. Sometimes the discussion on the pros and cons are fought out as if it were a religious war. On closer inspection the problems at hand are quite transparent and result from the deprecation of the proprietary business model which is desperately trying to compete with evolution. As we have seen the core reasons for the uptake of FLOSS are neither religious nor altruistic but simply inherent to good software development. The reason for the intermittent success of proprietary models was the absence of a ubiquitous network of communication that worked at marginal cost – the Internet. Now that we have it and know how to use it the exclusive nature of proprietary software business models has a problem.

Proprietary software needs to make all money **before** the users can run the software in productive environments. With Open Source this is different. It can be run any time at no additional cost and with no commitment from a long time contract. If it does not work it can be exchanged – obviously with some cost but a lot less than what proprietary marketing wants to make us believe for so many years. Which brings us to the most obvious problem in the proprietary/FLOSS struggle: Marketing. Proprietary has too much of it and FLOSS to little. Over time a lot of Fear Uncertainty and Doubt (FUD) has been spread to the detriment of Open Source software. This has understandably caused a backlash of wild arguments against proprietary software from an as wildly marketing-unaware group of geeks. But these have organized themselves over the past years and done good work in removing most FUD so that Open Source is now socially, technologically and financially acceptable.

FLOSS will make life a lot harder for monopolists who cannot innovate as easily as an open community of thoroughly networked developers on the loose. Especially monopolists are well advised to carefully adjust their business models to this new challenge. On the good side of business FLOSS is an enabler for innovation and a door opener for start-ups and small and medium enterprises. These will also make sure that business will be more local making it more efficient and more attractive for public administrations and governments as it strengthens the local economy.

Conclusions

It can safely be said that Free and Open Source Software is here to stay. Change in large organizations has a high latency, therefore proprietary business models will be around for many years to come. Companies who employ hundreds of sales people cannot change their business model in a day. The same applies to organizations like cadastral base map agencies who operate very large and complex sets of data with high Vendor-Lock-In potential. On the other hand spatial IT also has a long tradition of using and adhering to standards because spatial data is by definition boundless and needs to interoperate. The convergence of standards and Open Source will be the core element for all future solutions.

References

[Perens, Bruce 2005] http://perens.com/Articles/Economic.html
[DoD, 2009] http://wiki.osgeo.org/wiki/Case_Studies#US_Department_of_Defense