

## Avoiding the pitfalls of Open source

IN THE LAST FEW YEARS THE OPEN SOURCE SOFTWARE MOVEMENT HAS GROWN SIGNIFICANT FROM A TRAIL BLAZING GROUP OF VISIONARIES TO A MAINSTREAM MOVEMENT THAT IS TOUCHING ALMOST ALL AREAS OF THE DESK, SERVER AND MOBILE SOFTWARE WORLDS. IN THIS ARTICLE WE ATTEMPT TO SHARE MANY OF THE KEY LEARNING'S WE HAVE MADE OVER THE PAST DECADE.



## Avoiding the pitfalls of Open source

In the last few years the open source software movement has grown significant from a trail blazing group of visionaries to a main stream movement that is touching almost all areas of the desk, server and mobile software worlds. Over the past decade Teleca has been engaged in a wide range of open source mobile software projects and communities such as Android, Maemo, OpenMoko etc. This has enabled us to develop a strong understanding of the potential pitfalls that can derail what should be a successful development project leveraging open source software. In this article we attempt to share many of the key learning's we have developed over the past decade with the aspiration of help you to reduce learning curves and to gain a greater benefit from your engagement with the open source software community.

*Andrew Till*  
*November 2009*

### **Why? Rather than why not ?**

One of the common causes of frustration in using open source software is the lack of clear understanding as to why a company or development team has opted to use open source. Simply saying why not is not a valid reason for making this change due to the fact that leveraging open source is about much more than gaining access to software. As detailed in the points below it impacts many parts of the business not just the development team.

Having a clear understanding as to why you want to use open source and how you plan to engage with the wider open source community is critical to ensuring success.

### **Open does not mean FREE!**

One of the misconceptions about open source is that it is free. While you can in fact gain access to millions of lines of code without cost, this typically does not enable you to deliver a finished product. The software development activities still need to be performed such as integration, regression testing or development of feature enhancements. Indeed given that everyone potentially has access to the same code it is challenging to build a high value differentiated product with pure open source code alone. Typically what companies or development teams learn is that leveraging open source allows them to re-factor where they spend their valuable development dollars but it does not reduce the development spend. The key issue is that leveraging open source should enable a greater investment in innovation and differentiation built on top of the open source code and a lower level of investment in core plumbing or none differentiating software.

### **It's a different development model**

Leveraging open source software is not just about finding free code on the web and integrating it in to your development project but rather it has implications for your overall development model. Typically open source projects are ran using Agile or Scrum methodologies as opposed to traditional waterfall development models widely found inside large companies. It's also important to define a community engagement strategy and contribution policy. For example do you actively plan to give something back to the open source community, if so at what stage of your development, how and will you provide on-going support. Will your developers be able to seek support from members of community projects and if so how do you ensure that do not compromise the details of your development projects. These and many more issues need to be carefully considered before setting developers free with open source code.

### **Define a contribution strategy**

Building on the previous point I believe it is very important to develop a contribution strategy as early as possible. This will help you to maximise your overall engagement with the open source community but will also ensure that you provide the clarity and direction to your development teams. For example will you contribute complete components or bug fixes, when will you contribute (during development, when you launch your product or service, post launch?) Will you support your contributions making people available to respond to questions or provide maintenance and integration support especially if you're seeking to have your code integrated within future baseline releases. It is also important not to be viewed as a "free rider" simply taking and never giving back.

Failure to define a contribution strategy can also have significant hidden future costs. For example if you are building a Linux based platform and developing additional components that you know will be standard at some point in the future, for example a DLNA stack, then you run the risk that if you don't contribute a competitor may. This can result in the core baseline software containing components that are less functional or core APIs that are not longer fully compatible with the components you have developed. This then leaves you needing to make modifications or having to re-integrate on to each new baseline release thus stealing away valuable development hours from creating the next big new thing. In addition defining what you will contribute back will also help you to clarify the licensing strategy you should embrace.

### **Select your License with care**

Perhaps one of the most daunting challenges in leveraging open source is to understand the licensing models and the implications they have for your development activities. This is perhaps the greatest source of confusion, fear and misunderstanding within the entire open source space. It would not be appropriate to try and give a fully detailed appraisal of all licensing issues in this article but it is certainly worthwhile investing money and time to understand which licenses you should brace and which ones are not appropriate for your development objectives.

Typically when coming to open source for the first time the focus is on GPL (General Public Licence) also known as copyleft licenses. Such licenses typically require that derivative works or modified versions are licensed under the same terms and conditions and that the source code is made freely available, along with a number of other key requirements, to users of the software. (Note that this does not apply for private use.)

However not all licenses in open source are GPL with two other main categories existing known as License contracts and Permissive license notices. These are typically viewed as being friendlier for those wishing to develop commercial products.

For example Android, the Linux based mobile device platform, developed by the Open Handset Alliance uses an Apache 2 license (License Contract) which allows companies to use the source code in devices and to make modifications without the need to contribute or open source them provided they include the license text (on an Android based device this is usually found in the settings/about screen). As a comment on licensing it is also important to fully understand which licenses can be used in combination and which cannot, as not all licenses can co-exist within a common software block. Some good sources for more information on open source licensing models are the Open Source Initiative ([www.opensource.org](http://www.opensource.org)) and the Free Software Foundation – FSF ([www.gnu.org](http://www.gnu.org)).

### **Protect your reputation i.e. don't contribute bad code**

Always remember you only have one reputation and while it takes a long time to build it can vanish in seconds. If you are planning to make significant contributions to an open source community, perhaps complete components, then it is just as important to ensure that the code has been tested and debugged as it is when making internal software releases. Do not knowingly contribute buggy code expecting others to debug it for you. Simply put the community will not stand for it and you will quickly be seen as a poor community member. Further more if you are relying on your code being integrated into future baseline releases then you will be in for a major disappointment. Other impacts may be on your ability to attract the right talent to your business especially if you are looking to hire from within the open source community. That said releasing alpha and beta code is perfectly acceptable provide you make it clear that this is what it is and document the known issues. So as the heading states protect your reputation and treat others in the community in a way that you would like them to treat you.

### **Identify the right business model**

One of the common debates in open source is can you make money. In our experience there are a wide range of valid business models that can happily co-exist with using open source software or indeed providing support services to those that are engaged with the community.

For example a number of companies build full software distributions, typically based on Linux that includes a wide range of open source software but fully integrates it in to a useable platform. Typically, consultancy, support, maintenance and adaptation services are provided around these distributions.

Many companies including Teleca, provide integration services taking open source components and integrating them in product base lines.

Customisation services are also valid business models. As already stated open source software means that anyone can have access to the same code. Therefore relying on the base components to build value added products is frequently not realistic and therefore services design to provide customisation or extensions to these components are perfectly valid.

These are a few examples of successful business models but there are plenty more and critically there are many examples of companies that are making a strong living from working with the open source community.

### **Appreciate that people don't like to pay ransoms**

While there are many valid business models there are also business models that can alienate the open source community. One such model is what we call asking people to pay ransoms – i.e. contributing core engines or components to an open source community with the intent of then making users dependent upon you for overly expensive customisation and extension services. The community is likely to quickly reject this model and before long your components will have been removed from the baseline in favour of alternative contributions. It can be surprising how fast the community can move when faced with being held to ransom.

### **Maintenance is NOT free**

I have attended many industry conferences where the subject of maintenance has fuelled long debates. It is often stated that the community will maintain software and contributions and that this is a strong justification for contributing your code back to an open source community. While there are many good justifications for making contributions, expecting free maintenance is not one of them. Looking at the basic dynamics of open source quickly underlines the issue. Most open source projects and communities are not structured like commercial organisations but rely on the good will of the members be it on their personal time or sponsored by their employer. However you cannot contract with the community to provide SLAs for maintenance support, achieve commitments for defect analysis and fixing or even ensure that you can set the priority of the bugs to be fixed. So my advice is simple – If you are delivering a commercial product or service then plan to support your code.

## Summary

Hopefully the points highlighted in the article will help you to develop an effective and rewarding model for leveraging and engaging with the open source community. The core theme running through this article is to plan in advance and to think through not just what software components you want to leverage but your overall engagement model with the wider community as the move to leverage open source has ramifications that ultimately touch all areas of a business.

For more expertise and advice, please refer to [www.teleca.com](http://www.teleca.com)