Dominique Le Foll, Amino Communications, Fulup Ar Foll, Sun Microsystems

Open Source Software in Telcos – a gentle Tsunami

Many people have not realised that as soon as they connect to the Internet, the large majority of software frameworks which provide the service are not owned by any corporation and are actually provided by the Community as Open Source. The fact that critical software frameworks which are so important for the world economy are managed by a ruleless system can be difficult to understand. This article introduces the reader to a definition of Open Source, why it has been so successful for the Telcos and the consequences of that change for the industry.

Technical management, who often did not know Open Source Software values, were not long in understanding the power and the benefit which could be provided by these tools and started to introduce them in critical projects.

Introduction – a short history

While Richard Stallmann launched the Free Software Foundation (OSF) in 1983, providing the compilers and a set of very useful tools, the OSF movement really became visible to the public with the arrival, in the early 90s, of three major projects – Linux, Apache and Samba.

Students could benefit from free of charge professional development tools (Linux, GNU Compiler), IT managers could solve their Unix / Windows / Mac interconnections problems (Samba) and the fast growing .com industry could ride on the LAMP (Linux, Apache, MySQL, Perl-PHP-Python) platform to deploy their infrastructure at low cost.

Launched in 1995, Apache had passed the bar of 50% of market share by 1999 and still strongly dominates the world market of public web servers (see http://news.netcraft.com/).

From the late 90s to early 2000, some of the largest commercial software companies were announcing that Open Source would never break into real business and critical applications (http://lxer.com/module/newswire/view/57261/index.html) while some Open Source advocates were predicting Linux everywhere (http://www.tomw.net.au/media/20000515.html). Both were wrong. Open Source and commercial software offerings co-exist and are likely to do so for a long time. Businesses had to learn to use the strength of Open Source to build new values for their customers.

Sun Microsystems has done this for years and more recently, Apple, with their Mac OS X operating system, have demonstrated the power of the concept. The value of commodity software had to be realigned on account of this new free competition. Downloading and running the free OpenOffice application suite can still surprise many people with its quality and capability (http://www.openoffice.org/).

1st wave : Telcos infrastructure

Telcos infrastructure software for BSS/OSS traditionally runs on Solaris and HP-UX – both of which are Unix derivatives – as well as their cousin, Linux. By the mid 90s, most smart engineers trained on Linux. Furthermore the quality and coverage of the Linux free development tools (e.g. Valgrin http://valgrind.org/info/about.html or Electric Fence http://directory.fsf.org/project/ElectricFence/) introduced Linux to the development lab by the back door. It simply provided a safer, faster way of developing a solution.

Technical management, who often did not know Open Source Software values, were not long in understanding the power and the benefit which could be provided by these tools and started to introduce them in critical projects. In 2004 the magazine Billingworld (http://www.billingworld.com/articles/feature/Open-Source-Balancing-Innovation-and-Risk.html) interviewed many CTOs (chief technical officers) involved in telecom software infrastructure development and revealed how the move to Open Source was deep but still mostly invisible.

The industry was in the turmoil of the end of .com and main motivations were to accelerate project speed and reduce costs while keeping long term maintainability.

The most surprising phenomenon was that customers, including major Tier 1 Telcos, accepted Open Source based software without difficulty. By tradition Tier 1 Telcos don't trust anything and have

extensive test and validation programs. They realised very quickly that Open Source based solutions were at least as good, if not better, than those based on commercial frameworks. Thorough tests led to the acceptance of underlying technologies such as web server (Apache) and database (MySQL). The use of modern scripting languages (Python) led to the general acceptability of other Open Source modules.

In that specific domain, one of the great value propositions of Open Source was its independence from the hardware. For the first time a development team could commit to a customer to maintain an application for up to 20 years. As there was full access to all source code (application framework, OS and development tools) it was possible to commit to run the application on hardware that did not yet exist. This would end the extra costs and nightmare produced by the requirement to upgrade from the OS to the application via the tools chain and the framework when the inevitable requirement to upgrade the supporting hardware triggers in a few years. It would also enable running critical applications on off-the-shelf lower cost computers. Google's success is built on that concept and in many Telcos numbers of Sun and HP hardware systems run Linux OS either directly or in virtualisation to provide that flexibility.

The second attraction of Open Source software was its quality and focus on customer needs. While commercial software has to add more features every year to justify the 20% maintenance contract cost, its alternative Open Source competitors focus on the actual business needs. The complexity of the functions added to the latest commercial office suite can help the reader to understand this behaviour of the market.

One of the Open Source mottos is 'A software solution is completed when there is nothing more to take out'. It drove Open Source to a serious success in the segment of the software market where 'nice to haves' are unused and are simply additional risk and maintenance burdens. On many occasions Open Source software proved to be more reliable and delivered higher performance than its commercial counterpart. In 2000, a monitoring project aiming at a major Tier 1 Telco in Europe was salvaged by the replacement of the commercial ORB (message exchange framework) and Database by free alternatives : OmniORB http://omniorb.sourceforge.net/ and MySQL http://www.mysql.com/. The new free ORB provided the missing resilience to intercommunication error and the free Database allowed a twentyfold increase in the number of managed Alarms per hour.

The acceptance was quicker in Europe and Asia than in the USA, but the high quality of some Open Source solutions broke the resistance on the other side of the Atlantic very quickly. In 2005, a Tier 1 Telco requested in a tender that Open Source solutions should not be offered but subsequently imposed Apache as the main Web server. Open Source was in and there was nothing that would alter that reality.

2nd wave : the network equipment

In the early 90s most Telecom equipment was structured around Real Time commercial or bespoke Operating Systems. The change in the network infrastructure toward IP as a generic protocol and Ethernet as a ubiquitous interconnection media forced major equipment upgrades. The new power provided by flexible hardware using Field Programmable Gate Arrays (FPGA) (http://en.wikipedia.org/wiki/Field-programmable_gate_array) or Digital Signal Processors (DSP) (http://en.wikipedia.org/wiki/Digital_signal_processor) had reduced the dependence on strict real time software and opened the use of pseudo real time OS such as Linux. Linux had full real time extension but it did take quite a while for it to gain market acceptance and proved unnecessary for the majority of projects. Furthermore at that time most real time programmers came from an electronic engineering background and adapting to the complexity of the Linux framework represented a big hurdle to jump. However, the growing complexity of the software established a need to create a larger pool of talented engineers who joined the industry, bringing a strong Linux background.

The first Open Source software packages to be accepted were the OS (mainly Linux), some key software components such as flash files systems, boot loaders, firewalls and configuration web servers.

Once again, the acceptance of Open Source arrived via the back door. The use of Linux and all the associated tools enabled faster development and provided more realisable equipment. Linux enforced the use of a full Memory Management Unit (MMU) which was mostly ignored by Real Time OS due to its adjacent complexity on critical timing control. Once past the hurdle of the MMU, the code became easier to debug. If the internal Application Program Interfaces (API) had been well thought out, cross development and partial simulation with separated host and target became possible. Furthermore finding trained software skills on Linux prove to be significantly easier than with dedicated Real Time OS.

Once the new development and test processes were accepted, return to the old world became impossible and in a few years, justifying the use of a dedicated real time OS became a serious challenge in most companies. The ubiquitous use of Linux by large corporations such as Cisco, finalised the acceptance of the Open Source and illuminated the way forward.

3rd Wave : the customer premises equipment

In the early 2000s most customer premises equipment (CPE) was running on small CPUs and limited memory (RAM and

Flash). Attempts to use Linux as a micro OS such as the Micro Linux implementation (http://www.uclinux.org/) remained mostly academic exercises.

The push for larger Flash memory in GSM phones to support cameras – and in digital cameras to support higher resolution – changed the memory cost so drastically that the size of memory issue went away. At the same time the software complexity in GSM phones pushed the embedded CPU manufacturers to integrate full MMU support in their design. The technical barrier to Linux was removed and the flow could not be stopped.

This time the acceptance came by the front door. CPE manufacturers are driven by product cost reduction and time to market. The 'free' licences could reduce the overall production cost by 10% to 20% and the increasing availability of trained software developers, together with the reuse of ready made software modules could advance project completion dates by 30%. The adoption rate was so fast that early adopter Startup companies had difficulty beating the response times of the major players such as Linksys or Netgear. The competitive advantage of the model is such that having to republish the source code of their product, as required by the GNU licence, is not an issue. See for example Netgear web site (http://kbserver.netgear.com/kb_web_files/n101238.asp.).

The CPE in domains where heavy IP networking is needed such as router, gateway or set top boxes is mostly exclusively based on Open Source. Very powerful commercial alternatives like WxWork, WinCE or Symbian are restricted to niche areas driven by specific features such as Windows applications compatibility (WinCE) or power management (Symbian) but have failed to expand outside their niches.

In less than 10 years Linux has grown from nothing to 50% of market share in the embedded market. http://www.linuxdevices.com/articles/AT7065740528.html

How 'free' is free Open Source software?

We all know that there no such thing as a free lunch. It is important to remember that Free software is to be understood as in 'Free speech' and not as 'Free beer'. The total cost of Open Source software acquisition is often (but not always) lower than its commercial counterparts. But this cost is never nil.

When you decide to 'buy' free software, you need to define your need, hunt the internet and various forums to find your best candidates. The general problem is that none of the existing propositions will do exactly what you want and tens, or worse hundreds, will almost cover your need. The next phase consists of finding the best candidate which will cover the need, provide the required quality and offer the desired flexibility to add the missing functionality.

Once you have made up your mind, depending on the activity supporting the selected Open Source module, you can have amazing support or be completely on your own. In this phase of the project you need to apply your best engineering resources to select the right options and properly predict the remaining tasks and correctly assess the associated risks. Software engineers actually capable of smoothly running this phase on complex projects are very rare and command high salaries. Many software houses specialising in Open Source are now operating in the market place and can be used to mitigate risks associated with this type of work.

Required skills and engineering talent

It is quite common that teams which have been correctly delivering projects using a commercial framework fail with an Open Source platform. Apart from exceptions related to major projects (webserver, database, virtualisation, office suite ...) in the Open Source world, there is almost no commercial support. In this world where nothing is hidden (source code is public) it is easy to get lost in the volume of information and nobody will back you up if you make a wrong step.

The software community generally states that correcting code is 10 times more difficult than writing it. When you select an Open Source framework you will also need to correct and improve code that you have not written.

In general the adoption of an Open Source strategy, will impose requirements to significantly change the engineering force. In a nutshell, Open Source based projects require fewer but smarter engineers. It's not uncommon to notice changes of up to 50% of the engineering group within two years. To survive in the Open Source world, Engineers need to be motivated to keep informed on what's happening in the community, spend time investigating, reading magazines, following mailing groups and be 100% immune to the 'Not Invented Here' syndrome.

Practical experience of large projects (50+ developers) shows that having a minimum of two solid architects and code experts is the absolute minimum and will make the difference between failure and success. Due to the nature of these experts, conflict with traditional management is common. Defining a hierarchical structure to recognise their critical technical value without burying them under heavy people management responsibility must be set up. Without these precautions, discord will arise and reduce the chances of success.

A common surprise is that most Open Source code is better documented and better written than much commercial code.

Nowhere to hide produces higher quality

A common surprise is that most Open Source code is better documented and better written than much commercial code.

When a developer publishes code to the Open Community the developer knows that any search engine can show it to their next boss (or their colleagues). Worse still it will still be visible 10 years later. Nobody can afford not to do it properly if they hope to survive in the jungle of the Open Source. Type the name of a programmer you know well in http://www.google.com/codesearch and you will understand the high pressure imposed on programmers for quality in a world where there is no hiding place.

Secondly, when trying to push a new project on one of the Open Source portals, high competition is to be expected. For example, Sourceforge, one of the most popular portals, referenced more than 175,000 projects in April 2008 (http://sourceforge.net). The Open Source echo system imposes an active selection based on evolution, which is not that different from Darwin's concept. Strong and well-managed projects gain high support and become stronger until a shift in the community need favours an alternative architecture - while projects badly designed are not followed and die very quickly.

A new type of software company

A new type of software company has been created around Open Source solutions and enables an approach which mixes the benefits of the free and paying worlds. Some of them provide very professional services which are certainly at the level of the best

traditional commercial offering. MySQL, Xen, Novell, RedHat or Montavista can be listed in that category. The main important difference to notice is that they sell a service and not a right to use. This makes the long tails and large deployment of projects more financially attractive in the Open Source world.

Other companies provide consultancy in specific domains like writing drivers, enabling complex frameworks and can be a valuable helper at start-up time.

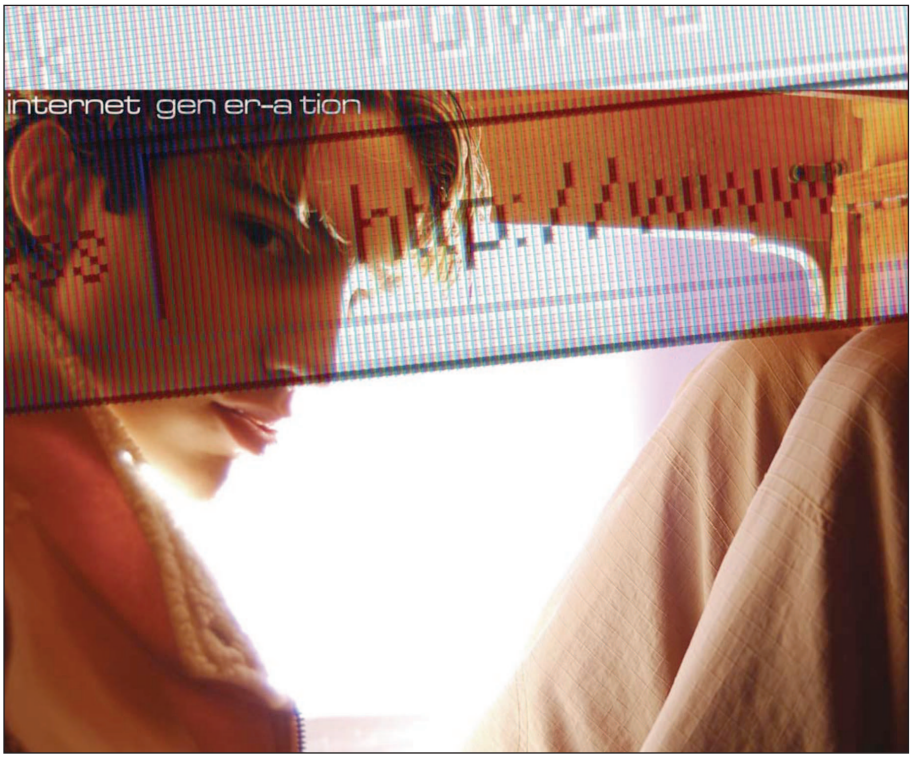
In Open Source as in the commercial world, the main risk is to start on the wrong foot through lack of preparation. It is not uncommon to meet managers who did not take the time to build it right first time, but spent the time doing it twice. With the lack of constraints and guidance within Open Source, that risk is even bigger.

A new way of creating and selling software for established companies

Quite surprisingly and independently of new players who build their business natively on open source, the biggest part of the open source contribution remains funded by well established commercial companies. While Sun chose a very clear software Open Source strategy a few years ago and became the first Open Source vendor and contributor to the community, other commercial companies like IBM, Oracle, Novell and even Microsoft participate actively with communities either to develop new projects or to improve existing ones.

While many people still do not understand the open source model, it has many key advantages over traditional software design. A traditional approach for commercial software requires you to have many different teams (marketing, development, support, sales, QA, documentation, localisation, ...), not only all do these people cost money, but too often they have trouble understanding one another. Open source

It is not uncommon to meet managers who did not take the time to build it right first time, but spent the time doing it twice.



completely changes the cycle. Marketing is done directly by the customer, support is the shared responsibility of developers' and users' communities and localisation is done directly by the people who need it, ... the result is a much shorter development cycle, better support, and a higher version/platform/localisation density matrix.

If we take Open Office as an example, while Sun remains a main contributor to core development, nevertheless most of the localisation (e.g.: three Norwegian dialects, three Celtic languages: Welsh, Breton, Gaelic, ...) porting and test to non-native platforms (ex: MacOS), promotion and branding as well as support is effectively done by the community. Without the help of the community Sun would never have been able to build in only few years a product that competes with Microsoft Office Suite. We could say the same thing for many other products like MySQL, Mozilla, Apache, Jabber, ... The Open Source model allows small teams, even when starting late in well established markets, to compete (ie: OpenOffice/Microsoft-office, MySQL/Oracle, Mozilla/IE, Linux/VxWorks, ...)

The selling cycle is another advantage of Open Source. Big commercial companies spend a significant part of their income in sustaining marketing and commercial effort. Companies like Sun and others – even when providing only open source software – still need revenue to pay developers. Many Open Source oriented companies allow free downloads but propose some form of paying support for whoever needs it. Customers do not have to pay upfront for a licence fee, but start paying only when they consider their application critical enough to justify the cost of support. This reduces the selling cycle to almost nothing; customers call in when the application is already in production. They do not ask for a demonstration, presentation or a benchmark, ... they ask only for the cost of support. Furthermore, as customers pay only when they consider the cost justified, this model never upsets customers. We have now a cycle where developers download a product, go to production and when management asks 'what about support?' then – and only then – they call back the vendor. MySQL consider that they make money on only 1% of the installed base. This did not prevent Sun spending $1 billion for its acquisition: http://www.sun.com/aboutsun/pr/2008-01/sunflash.20080116.1.xml

How does it work?

Open source software is provided under a large variety of licence schemes. To be accepted as a Free Licence by the Open Software Foundation (http://www.gnu.org/philosophy/free-sw.html) it has to respect certain basic principles:

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose

A common misunderstanding is that if a part of your software is free, all the software running beside it will also be free.

- The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this
- The freedom to redistribute copies so you can help your neighbour
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

A common misunderstanding is that if a part of your software is free, all the software running beside it will also be free. This viral effect, which is much published by some commercial companies, is generally not true. Many of the 'Free' licences do not impose a requirement to republish your code back into the open and the most commonly used licence (General Public licence) only requires it in very specific cases (direct change in the source code or static link to the code).

The general principle of these licences is that the initial creator remains the owner of the creation but leaves the right to the public to develop it further. If your proprietary code has to be statically linked to an Open Source code, then you need to pay close attention to the original licence. This risk is bigger with non Linux embedded software or drivers.

The concept is not limited to software and is also applied to documentations, specifications (http://www.homegateway-initiative.org/publis/index.html) and even hardware (http://www.sun.com/processors/opensparc/index.jsp).

Why does it work?

A large part or the software developed for a project does not represent a real asset for the company which develops it. Either it is an enabling technology, as for example a file system in an IP Set Top Box (STB), or it has no value on its own because it is a required commodity such as a Firewall in a router.

In the Telco world where software remains in service for years, about 80% of the full cost a software project is linked directly or indirectly to its maintenance. The main value of using Open Source blocks is to reduce the maintenance burden.

If you create a nice file system and you can convince a large community to use it, this community will have to maintain it, helping you to share the burden. If you self engineer a commodity you will not have a long advantage over your competitors. If you succeed in reusing and improving a commodity you will gain time over your competitors and raise the market expectation by creating a value for your solution.

Finally, by releasing software back to the community, companies gain independence from their software developers – limiting risk in case of conflict. An Open Source published project will motivate engineers to deliver high quality code and more complete documentation better than any project leader. Becoming a member of the club is highly valued and remaining respected is challenged by the newcomers every day. The Open Source echo system produces natural motivation and provides surprisingly powerful results.

Conclusion

The independence of software in relation to hardware enables 20 years of support commitments and the availability of trained software engineers has opened wide the doors of all Telcos in the world to accepting Open Source software.

As the initial acceptance of Open Source has been gained on operation critical projects, its widespread acceptance did not meet strong management resistance.

The main challenge remains the availability of highly skilled software engineers who can master the complexity of the solutions offered by the Open Source community and the availability of technical architects who can organise the integration of independently developed software blocks.

With the right people, an Open Source based project is likely to deliver a better, safer, cheaper solution in a shorter time, but with the wrong team it will fail to deliver anything with nobody else to blame but themselves.

Being your own master as usual is a privilege and a serious responsibility. Open source is no different.

The authors

Dominique Le Foll holds a Master Degree in Computer Science from the French Military School ESAT. Before joining Amino he was a research engineer for 10 years on voice and data switching technologies for the French Department of Defence and then joined Acterna-JDSU as architect for service assurance and test solutions. Over the course of his career, Dominique has used disruptive technologies to create diagnostic product lines for digital TV, DSL and ISDN test systems as well as TVoIP and VoIP. He has architected and engineered test and monitoring solutions, which have been deployed by companies such as DT, FT, BT, EutelSat, BBC, Verizon, AT&T and TWC. Dominique has also won several patents in the US, Europe, and Asia. Currently he works for Amino Communications as vice president of engineering, where he develops solutions for high definition IPTV that are deployed in many countries in Europe, the Americas and Asia.

Fulup Ar Foll holds a Master degree in Computer Science from the French Military School ESAT. Before joining Sun he was a research engineer for 10 years on distributed technologies for the French Department of Defence and he taught internet and Java technologies for six years at South Brittany University. For Sun he has been the lead internet architect for many projects related to European Telecom operators, as well as for other strong identity and security infrastructure users such as banks and governments. In the recent past he helped France and Norway to move toward the Liberty Alliance Federated model. He is currently master architect inside Sun global software practice and focuses on high scale federated identity issues. He represents the Sun software customer service group inside Liberty Technology Expert Group standardisation committee, and inside OMA MWS group and works as lead architect for major identity projects on a world-wide level. He has also been speaker at many international conferences.