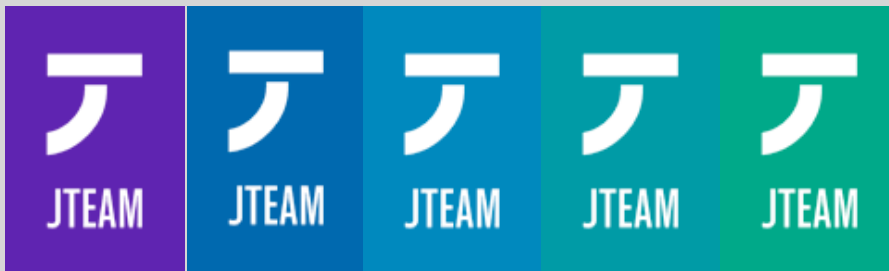


Open source

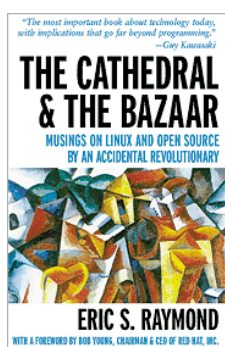


October 3, 2008

Does professionalization kill open source?

Some time ago I was talking to a friend of mine. We were talking about an article I had written "[Open source, fashion or commodity](#)". He asked what had changed in the last two years in the open source landscape. One of the things that came to mind is that open source is becoming more professional. Together we started discussing whether this was good for open source in general or not. This article discusses the question about professionalization and what it does to open source and most of all, the community.

When talking about open source, what are we talking about? I am not talking about the [sources of my samples](#) that are online at google code. Yes, these sources are open for all to use. But there is more to open source than that. For me open source is about the community behind open source projects. A lot has been written about what drives



these communities. An excellent, and now classic, book is "[The Cathedral & the Bazaar](#)" by Eric S. Raymond. In his book Eric describes the foundations of what open source has become today. The first thing that struck me in the book was not written by Eric however. There is a foreword

by Bob Young, from Red Hat. He is talking about freedom. In his eyes, open source is bringing freedom to the software world. I'll get back to this later on. The book itself is about the

differences between the two major development styles.

Two major development styles

The Cathedral style is used mostly in the commercial world, and the Bazaar style is mostly used in the open source world. The cathedral style is based around the idea of the perfect release. Each release should be feature complete and without bugs before the actual users see the end product. The bazaar style takes a completely different approach. The mantra for bazaar style development is "*Release early and release often*". Releases can be buggy, but the users are mostly developers who can tackle problems, provide fixes or at least concise error reports. Of course, there is a danger in this as well for some users. Not all users want these bugs, therefore you should educate them and help them in making the right version choice of your project. Smells like new requirements for the developers when creating code. That results in less time for developing those nice new features. So it

is negative to have non developer users who do not want the latest and greatest, or not?

The community is the secret

The predecessor to open source software was free software. Open source reflects better what the community wants to achieve, sources open to everybody and an open communication style. Free software, as in no license costs, reflects better what a lot of users think about when talking about open source. Now what is the ideal community for an open source project? A community can consist of a number of different users. It has developer type of users that are on the bleeding edge. They give immediate feedback on new features and if you are lucky they will come up with new features. Without this type of users nothing will happen in the project. The other type of users are the non-developer and the enterprise users. They require more stability of the product, less releases and preferably less bugs. The question you need to ask is, does a community need these non-technical and enterprise users?

What does a community need?

Eric Raymond has done a lot of research into open source communities. His findings are of course related to software development. I believe however that his findings are generic group dynamics. An effective group needs a strong leader. The leader is respected for his deeds. Consider a soccer team, the captain does not have to be the best goal keeper, or the best striker. It is the player with a lot of experience and a player who is highly respected by the team members. There are a lot of other examples to find, but we'll return to software development. Great communities are led by great leaders. Some well known examples are Linus Torvalds (Linux), Rod Johnson (Spring framework), Roy Fielding (Apache) and Marc Fleuri (JBoss). They all know how to bind people to their projects. It is pretty amazing that people will spend a lot of free time to create some code without being paid, just for honor and recognition.

Next to a strong leader, the community needs passionate users. These users feel that the project can help them take away an itch. Sometimes the itch is to have an open source alternative to a

commercial product. In other situations, it's because there is no solution yet. Sharing the source and helping each other out is very important in the community.

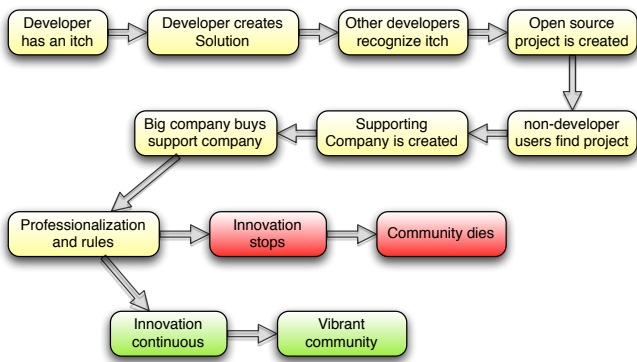
There are some lines the open source community should not cross. Always be respectful to the actual creators of code (thou shall not steal). The community must be open to new members. Aspirant members must be willing to prove their ambitions and capabilities before becoming full members and for instance getting commit rights to the repository.

The communities want to spend time on creating cool software. Anything that prevents them from doing so will be demotivating, with the risk of losing valuable resources for the project.

Steps an open source project goes through

Each (open source) project undergoes a certain evolution. A project usually starts with a developer having an itch (again from Eric S. Raymond). The developer starts finding a cure for his itch. If he finds the cure, he'll try to find others with the same itch. Then the community starts growing, other (core) committers or members are added to the project. Next step is the growing user base that is testing every release. Now that the users and members groups have become large, the initiator loses detailed control and starts doing only marketing to get more people on board. Big companies are become interested, but they need a release strategy and longer support for older versions. As a result, more people need to do maintenance work. They want to get paid and therefore more money is needed. More marketing budget is needed. More customers, commercializing the new features. No innovation from the community, no credit to the community, no fun for the community. All rules from the Bazaar style software building are broken.

The following image gives a schematic overview. It is all about innovation, no innovation means no community.



If the owner is lucky he'll earn a lot of money when selling his company. Money he can use to start working on his new ideas. Hopefully he did not lose his credibility and he will again find a vibrant community. His old product, now under the hood of one of those big companies, dies a silent death. Again a nice open source project with a good community is killed.

This might be a bit exaggerated as there are a lot of projects out there that prove that big projects that are highly supported by a big company do not die. Maybe the user base becomes less developer minded and more user minded. A few examples are [Openoffice](#) backed by Sun, [MySQL](#) backed by Sun, [JBoss application server](#) backed by Redhat and the [Apache project](#). Most of the downloads for these kind of products are done by users not interested in the source. Still the products are very interesting, with a vibrant community. You can also say that these projects are backed by companies because there is a large non-developer user base. It is this community that makes it interesting for companies to get involved.

There are a lot of open source projects out there that are backed by companies. What does that mean for Enterprise involvement in open source?

Enterprise involvement

The past years were very important for getting open source software between the walls of the bigger enterprises. There are a few directions that are important. In the end it all comes down to the same thing, money. Creating software takes man hours, more software means more hours. When a project has become successful, the owner(s) feel they deserve to make some money. The owner(s) also

believe they can spend more time making next versions of their excellent projects if they earn some money. There is a catch however. Getting money for software gives you some rules. Just a few of them are: bug fixes in old releases, answering questions and strange problems of your paying users. Erik van Oosten wrote a blog post about paying open source developers titled "[Do not offer money](#)". He gives good arguments why we should not offer money to open source projects as a reward. He also concludes that money can result in less service than you had before you gave money as a motivation for the service you had received. An other problem is keeping your community alive. Why should people help you if you get paid and they don't?

Another often used business model is setting up a consultancy firm backing up an open source project. These companies earn their money by offering training and consultancy for their products. You will also see that a lot of these companies provide enhancements to their products that are commercial. I think this is a dangerous model. It is hard for your users to grasp that they can have the core parts, but the nice additions are only for the paying customers. Again why should they contribute if it might disappear into the commercial part?

A recent example of using this strategy is the company Spring Source. They are the company behind the very big Java framework called "Spring framework". Rod Johnson managed to create a very large community. Spring Source is becoming more and more commercial. A lot of employees of Spring Source are creating high-end software and they need to be paid. The guys are doing their best in creating a business that remains very open to the community. Still they have been openly criticized by the community. Of course the competing projects, and companies backing these projects, are leading the attack, but still it hurts the community to create a dual license for open source usage and enterprise usage. Recently they have come up with a new [maintenance license](#). I really did not like it at the beginning. After reading it closely and reading the response of [Rod Johnson on the serverside](#) I do understand it better. In the end, they keep on doing what they did, delivering a high quality new release

every quarter and maintain that version until the next version comes out. Within the enterprise world a more demanding version support is requested. They need a release to be supported for at least a number of years. A good point of view on the Spring Source licensing is written down by one of the co-authors of my blog Ben Tels. Check his blog item here: [When good guys start looking like bullies....](#) Will they succeed in keeping their community? Time will tell.

A lot of companies have started with open source in their business model. If you are looking for more information on earning money with open source, I recommend the book “[Open source for the enterprise](#)” by Woods and Guliani. Do not forget to cherish your community and keep innovating.



Does a professional approach kill open source?

In my eyes professionalism can hurt the community. The open source project will remain to have their sources open. Still the community might be endangered. This can mean that a professional approach to an open source project leads to a project with open sources and a killed community.

Of course there are exceptions. In these exceptional cases, professionalism comes out of the community itself. This tends to happen when the user base becomes very big and a lot of the users have become non-developers. Still it is the developers that need to have the ambition to come up with things like a release strategy and long term support. Some of the open source projects that have done this successfully are linux distributions like Ubuntu, the whole Apache organization and the Openoffice community. Companies that back open source projects need to find a way to take these professionalization steps without frustrating the developer community. Most of all, the companies must keep innovating.

References

Books

“[The Cathedral & the Bazaar](#)” by Eric S. Raymond

“[Open source for the enterprise](#)” by Woods and Guliani

Blogs

“[Open source, fashion or commodity](#)” by Jetro Coenradie

“[SpringSource Announces Enterprise Maintenance Policy](#)” by SpringSource

“[New Spring maintenance policy](#)” by different authors on the serverside

“[When good guys start looking like bullies....](#)” by Ben Tels

“[Do not offer money](#)” by Erik van Oosten

About the author

Jetro Coenradie is working for JTeam B.V. as the Chief Architect. JTeam is a company used to working with innovative products. These products are mostly open source, but using open source is not a religion. If there is a better fit with a closed source product, that product is used. JTeam has gained a lot of experience with open source by making contributions to open source products and by helping open source projects start up. JTeam works together with a lot of companies that support open source frameworks.

Jetro has gained a lot of experience with open source in the 10 years he has worked in ICT. Jetro started with products like linux and Openoffice. Later on, he started using a lot of java based open source frameworks and products. While working for Capgemini and Accenture Technology Solutions he was always recognized as a pioneer and open source expert. Jetro has advised customers on open source usage and guided them with their first steps in the open source world. Recently Jetro has become a member of the open source expert panel of the Computable, a dutch ICT magazine.

Jetro is a heavy user of open source software. He is also very interested in the way open source projects work and how the communities behind the open source projects are evolving. Jetro is using these practices to improve the way of working on his own projects.