# The 7 principles of successful open source communities

**Thomas Thym**
Institute for the development of viable organizations
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
DE-85577 Neubiberg
thomas.thym@gmail.com

## ABSTRACT
This paper presents the seven fundamental principles of successful open source communities, and drafts some consequences for communities and classical organizations.

### Author Keywords
Open Source, Communities, Principles

### ACM Classification Keywords
A.0 [General]: Conference proceedings

## INTRODUCTION
According to the classical management theories, open source software, built in unmanaged and unpaid communities, should not exist. However, this successful phenomenon that is way beyond a group of hobby developers does exist and has built a growing economic system around the open source movement. There must be other reasons than money and formal orders which are responsible for the creation of those open source communities.

*What is fascinating?*

## EFFECTS
The interesting effects could be grouped into two categories:

1. Open source Effects and
2. Community effects

### Open source effects

*New business models*
Open / free software licenses and open standards are the basis for a new form of cooperation between business partners e. g. joint development or open innovation (e. g. the printer software CUPS).

Revenues are not created from licenses but from additional services around the software (e. g. consulting (KDAB), software modification or hardware sales (e. g. IBM), dual licensing (e. g. former Trolltech and MySQL).[1]

*New forms of value creation*
Apart from new commercial models, many non-commercial forms of value creation exists. The aim of the cooperation is not earning money but creating direct value by sharing and collaborating. Famous examples are the free encyclopedia Wikipedia[2], OpenStreetMap[3] or the free audio books service LibriVox[4].

*New market entering strategy*
The third open source effect opens the possibility to enter a market even without much financial resources or against strong competitors (even monopolies). Examples here are Netscape / Mozilla's web-browser (Firefox) against Microsoft's Internet Explorer [5] or Brewtopia, a start-up brewery in Australia resetting the competition against the three dominating players.[6] New forms of marketing, like viral marketing, are part of that category.

Open source effects are typically based on open licenses and provide strategic advantages.

### Community effects
The second group is community effects. These are the phenomenons Linus Torvalds activated with his shift from the cathedral to the bazaar-style development model.[7]

Community effects offer a new form of collaboration and a new way of handling complexity by using **self-organization** and **collective intelligence**. These new methods support the integration of masses, reduce reaction-time of the whole organization and allow rapid growth without central control.

Impressive is the **fascination and passion** that is created by the communities leading to overwhelming commitment without extrinsic motivation (like money).

*How are the communities creating these magnificent effects?*

## ANALYTICAL FRAMEWORK
The open source phenomenons exist because humans in particular communities are collaborating in a special way. The individual actions are forming typical processes. Without these essential processes the favored effects would not exist.

*Why do people act as they do?*

Several studies are trying to explain that question with classical motivation theories. These causally determined models follow the pattern: “Developer *A* is motivated by X.” The consequences would be that managers just have to motivate their subordinates with X to create open source effects.

To look behind the effects, a social psychological model from Kurt Lewin was enhanced.[8]

Lee Ross and Steven Samuels analyzed the behavior of cooperation and cheating.[9] In the first step the personalities of students (whether or not they would cheat or cooperate) were ranked by their tutors. The students then played a prisoner's game.[10] The study showed that the correlation of cheating does not depend on the (assumed) personalities but on the naming of the game. When the game was called “community game” about 70% cooperated. In the “wall street game” only 33% did not cheat (independent of the assumed personality).

The rules of the games were identical. This experiment shows the power of a little detail of the situation (the **context**).

On the other hand the behavior depends on the **person**. Different people react differently in the same situation.

The behavior is influenced by

- the character of the person,
- the individual needs,
- the skills,
- the experience and
- the basic assumptions (the view of the world).

Summarized: The behavior of a person depends on the **personality** itself and the **context** of the situation.

Before the effect building processes could be outlined, the personalities and the context of open source communities will be analyzed.

*Who is contributing?*

## PERSONALITIES

A couple of detailed studies examined the characteristics of open source developers. [11]

### Statistic characteristics

The major part of open source developers are male (>95% in most communities) and are younger than 30 years old (70%). They live all over the world but most of them come from developed western countries (especially from Europe and Northern America).

In the last years the proportion of paid contributors rose. Still for half of the developers open source activity is just a hobby. Most of the work although is accomplished by the paid community members.

### Characters

Krogh, Spaeth and Lakhani analyzed the characters of potential new contributors in mailing lists.[12] Based on that work three types could be found.

- **Proactive problem-solver:** They use the program, find a bug, and work out the solution. In the first mail to the list they send the patch. These people are very successful in communities and often become continuous contributors.
- **Waiting volunteer:** This group offers their abilities to the community and waits until they get a job allocated. In general this character is not very active. Most communities can not integrate them successfully.
- **Visionary:** They use the program and have ideas on how the program should be improved. Although visions and aims are important in communities, the character-type visionary is not successful. In the past his/her visions were not identical with the ideas of the code developers. The resulting costs of conflicts exceed the benefits of the discussion.

An analysis of a few famous open source contributors draws a first picture of the character of a successful open source developer.

The typical open source contributor is

- passionate,
- searching for challenges,
- curious, and enthusiastic about technology,
- oriented on technical arguments not on the origin (person) of the argument and,
- humble, with no ambition to assert oneself.

### Skills

Developers need special knowledge and skills to be able to contribute. In addition to **technical knowledge,** (e. g. the ability to code) **social skills** (like respectful communication) play an important part. Especially in a situation of disagreement constructive feedback is essential to keep the community healthy while finding the best technical solution.

All successful contributors share the ability to **learn autonomously**.

## CONTEXT

The context consists of other protagonists and their interactions. Not only is the behavior of others important for the decision but also their expected reaction. The behavior depends again on the personality and the context.

Major components of the context are the premises (basic assumptions) of that group (e. g. found in the organizational culture). The basic assumptions include a common understanding about core values and processes (how things should be done).

### Culture

Open source / free software exists as long as computers exist. Universities and other research facilities developed the first computers and the software to run those machines. In this scientific culture it was obvious to share knowledge, (including the source code of the software) and build upon these findings. The traditional values of that environment (freedom, growing knowledge and fun) resemble persisting core values of open source communities today.

### Values

In detail these values could be found in the code of conduct or social contract of several projects.[13] The documents were created by the members of the community. Corporate values in enterprises are often defined by the top management and are not completely accepted by the employees.

Values may differ from one community to another. The following points could regularly be found:

- Respectful and open communication
- Sharing and modification (sharing spirit)
- Pragmatic coding ("just code")
- Collaboration (Think of others and maximize the success of the project, not yours)
- Freedom (self-determination and responsibility)

### Leadership

There are hierarchies in communities. The position depends on the achievement of that person. In contrary to many classical organizations the level of hierarchy is not based on the accomplishment for his superiors but for his followers.

Leaders are measured by their actions, not their words. They lead by doing the first work and offer help to those who want to follow. The philosophy could be described as "making things possible for the followers" and "challenge them from time to time". Community members don't want to be pushed, but guided!

*How do open source communities work?*

## THE 7 PRINCIPLES

Open source contributors with particular characters and skills create open source and community effects in the special context. These effects are provoked by characteristic processes. All processes subject the seven principles of open source communities.

### Openness

The core of an open source project is the code and the open (or free) license of that code. The license allows the user to run, copy, distribute, study, change and improve the software.

Furthermore, structure and processes of communities are open. There are no boundaries between users, developers and project-manager. It could be difficult to decide how big a community is because it is not clear who is a member and who is not. The definition when a user becomes a community member is blurry.

In general, communities welcome new members. Everyone who would like to participate is invited to join the team. The core team grants unrestricted access to their information and provides transparent communication (almost all mailing lists, code repositories, documents etc could be accessed by anyone).

### Scalability

Big open source communities are successful because they are highly scalable.

The size of a central controlled organization is limited by the ability of the leader. The bigger the organization gets, the higher the workload gets and decisions for the central leader become more complex.

**Self-organized and decentralized systems** do not suffer that limitation. They can grow continuously. At a certain size they just split into autonomous sub-projects.

The sub-projects (the code as well as the teams) are **modular**. The development of one module does not effect the other autonomous parts. This modularity allows **simultaneous engineering.**

In most open source environments there are several modules for similar purposes. One the one hand, this **redundancy** is inefficient. On the other hand, this **diversity** increases stability and quality of the whole system. If one module fails, another could be used. An environment which allows riskless **experiments** is the basis for evolution.

### Circular feedback

Developers and users share their thoughts, comments and evaluations with each other (peer review). The feedback process is the basis of learning and pushes the quality to the next level as long as the feedback was **given** in a professional way (respectful and constructive), and was **accepted** by the counterpart. The best feedback is not worth anything if no one learns from it, and does not change his/her behavior.

### Pragmatism

The philosophy of experimenting could also be found in the way of production: Rapid prototyping.

Most developers favor a pragmatic and simple approach. The advice for new contributors often is not to develop detailed concepts, but to start coding. The real problems reveal themselves during the programming, not the concept phase.

Instead of developing perfect applications, they further suggest to release a prototype in an early stage (release early) and provide updates in short periods of time (release often).

This method uses the knowledge and skills of the whole group most successfully (use of collective intelligence).

### Social interaction

Another important point way beyond technical processes are social relations between the community members. In addition to technical knowledge and ideas, many contributors share personal information (e. g. on their blogs, micro-blogs, conferences or developer sprints).

On the first glance social interaction seems to be inefficient too. In the long run the advantages weigh more. Personal relationships bind the groups together, and belonging to a group has a significant impact on the behavior of a human and helps to solve problems in conflicts.[14]

Communities demonstrate the success of the recovery of humanity in productive environments (in contrary to short term thinking or shareholder value orientation).

***Communities are about people. Great software code is the result of a great community.***

### Freedom

Open source software is not possible without freedom. In this case the focus is not on the freedom of the source code (therefore see “openness”) but on the **freedom of choice**. The freedom of personal decisions includes the choice about the tasks, the preferred solution for a problem, the invested time, the colleagues, the leader etc.

There are no formal orders from superiors what to do, how to do it, and when. Even when there are hierarchies in projects, the final decision (e. g. in conflict situations) is up to those who actually do the work, not the project-leaders. To force one's opinion or code down to anyone will fail.

The contributors decide by themselves. They chose which information they want to get. They subscribe to the media (mailing list, bog etc.) they think might be interesting for them. They get the information they want (pull communication). In classical organizations the superiors decide which information is important for the employees and will push it to them (if they like it or not).

Communities put freedom into practice and cultivate a consequent form of **self-determination**. Without self-determination open source projects would not be self-organized and scalable.

However, with great freedom (self-determination) comes great **responsibility**. (Otherwise the projects would fall apart.)

Developers take the consequences of their own actions for others into account. They think outside their own modules and take care of the interoperability of the whole project.

New contributors try to solve the problem on their own in the first place. They learn by themselves, read documentations, search the web, and experiment. Only the last step involves consulting with others about their problem.

### Personal relevance

Open source contributors are doing things they care about. They love what they do but more importantly: they do what they love. They don't contribute (at least in the first place) to earn money. They live their passions.

Communities are built around the same personal needs, problems, interests, challenges, values and passion.

It is quite impossible to grow passion in an environment of pressure and fear or in a context where the employee thinks her/his commitment only increases the bonus of somebody else, or a shareholder's value. A passionate, supportive, positive, and fearless context is needed.

These seven principles shape the processes in open source communities. The processes are highly supported by software tools (wikis, source repositories, mailing lists, blogs, instant messaging, micro-blogging etc.).

The developers have the ability to modify the tools to their particular needs if necessary. Again they have the freedom to decide.

*Why are the communities successful?*

## SUMMARY

Communities are successful because they follow the seven principles. Not only are the processes shaped by the principles, but the personalities (characters, values) and context (culture, common values, expectations) reflect them as well. All elements pull into the same direction, and follow one aim.

- Openness
- Scalability
- Circular feedback
- Pragmatism
- Social interaction
- Freedom
- Personal relevance

*What consequences follow for communities and other organizations?*

## CONSEQUENCES

Open source effects could be enhanced by improving or introducing the open source principles in the areas of processes, personalities and context. Processes depend on the acting personalities in a specific context. Consequently personalities or context could be modified to boost open source effects.

It is very difficult (or rather impossible) to change personalities (beyond technical training). The more promising option is to change context. Every person is an actor in his/her particular environment. That is to say that everybody can **change context** very easily.

Unfortunately there is no causality while dealing with humans. It is not possible to predict the exact behavior of a person. Therefore it is impossible to draw ***one*** foolproof instruction. The success of an intervention depends on too many factors.

The recommended suggestion is to **experiment**. Trial and error offer the possibility to learn about the context. Experiments have (by definition) no certain outcome. They might and they certainly ***will fail***.

The important point is to learn from failures, to change the experiment, to adjust it to the unique situation and to try again.

There are boundless possible experiments. Here is just one simple example:[15]

### Spread your idea and grow your community

*Personal relevance*
People have different problems, experiences, needs etc. It might occur that issues that are important to one person are irrelevant to another. So make sure that your idea is fascinating for the target group.

Business persons might not be convinced to use open source software by ideological arguments of freedom of speech. They have other values and expectations. They might listen when they hear that they could solve their problems easier, faster, and cheaper than before.

Do not tell them what is interesting for you, tell them **what is fascinating for them!**

*Pragmatism*
Draft a version and create a simple prototype that works, e. g. a concept how to spread the idea, information brochures, basic presentations etc.

Make it simple and easy to participate.

*Openness*
Release it under an open license and allow anyone to improve it (e. g. on a Wiki). Invite others to participate. (Remember the personal relevance.) Provide open and transparent communication. Judge the quality of the content, not the person.

*Scalability*
Don't lead the project. If someone steps up and wants to take responsibility for a part (module) be happy that he/she is following and support him/her.

*Freedom*
Provide the freedom of decision wherever possible. Support decentralization to make the project scalable.

*Social interaction*
Additionally, to technical communication also support social interaction. Know your fellows, know their needs and ambitions. Integrate them, show them that they belong to the community and that you care about them. Especially think about the “waiting volunteers”. They need another context than the usual “proactive problem-solver”.

Define your common goal and your values together and pay attention that you (and everyone else) follows them. Otherwise kindly remind them and let them remind you. Mentoring social skills (in addition to technical mentoring) is an excellent way to insure a healthy community.

And last but not least: Have fun! Only very few people like to spend their leisure time in a boring or frightening environment.

*Circular feedback*
Kindly ask for feedback from fellow community members as well as from the target group. Structure the discussions and make sure it leads to a concrete result.

Create a trustworthy context by making clear that you are interested in his/her opinion and really want to improve your work. Optimize the quality of the product and support individual learning as well as learning as a group (e. g. ask directly what everyone has learned in the discussion).

Keep in mind: Whatever your partner says, it is her/his truth, based on his assumptions, on her experiences etc. This external view is extremely valuable even if you think it is offending you.

Often, the first thought in conflict situations is that the other person is wrong and therefore should change the behavior. The funny thing is: Your discussion partner thinks the same of you. Instead of trying to force others it is more effective to modify the context – and that means to change oneself.

It is tremendous difficult to mistrust your own senses, your experiences, your values, your picture of the world, and your perfect solution to the problem you are all facing. It is so hard to believe that you are wrong. But in fact, you sometimes are.

So be open minded and try to think (at least for one second) that this time the other person might be right. Perhaps she/he is.

These are just proposals for one experiment. Play with that idea, extend it, evaluate if it works for you and your situation and if necessary, adapt it.

## CONCLUSION

Particular processes built by individual action creates open source and community effects. The behavior of a person depends on the personality and the surrounding context of the situation.

The underestimated importance of context and personalities, is the reason why many attempts to introduce open source effects in organizations fail. Installing a Wiki on a company server may provide the tool for collaboration but not the necessary context.

In the undetermined world of human interaction trial and error provides a way to learn and influence context. The experiments should incorporate the seven principles of successful open source communities in all their areas.

The heart of an open source project is the community. The prosperity correlates with the ability of the members to form a passionate, respectful, healthy and focused community.

Create a **great community** and
**great results** will follow.[16]

## REFERENCES

1. Glyn Moody, Rebel code: Linux and the open source revolution, 2001, p. 205.
2. Wikipedia, Wikipedia, the free encyclopedia, 2007, http://www.wikipedia.org.
3. OpenStreetMap, OpenStreetMap, 2009, http://www.openstreetmap.org.
4. LibriVox, LibriVox - Acoustical liberation of books in the public domain, 2009, http://librivox.org.
5. Joseph Feller and Brian Fitzgerald, Understanding Open Source Software Development, 2002.
6. Brewtopia, What is Brewtopia?, 2010, http://brewtopia.com.au/about-brewtopia.php.
7. Varda Liberman, Steven M. Samuels, and Lee Ross, The Name of the Game: Predictive Power of Reputations versus Situational Labels in Determining Prisoner's Dilemma Game Moves, 2004, http://psp.sagepub.com/cgi/content/abstract/30/9/1175.
8. Eric S. Raymond, The cathedral and the bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, 2001.
9. Wikipedia, Prisoner's dilemma, 2010, http://en.wikipedia.org/wiki/Prisoner%27s_dilemma.
10. Kurt Lewin, Feldtheorie, 1982, p. 196ff.
11. Rishab Aiyer Ghosh and Vipul Ved Prakash, The Orbiten Free Software Survey, 2000, http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/769/678,

    Gregorio Robles, Hendrik Scheider, Ingo Tretkowski and Niel Weber, Who is doing it?: A research on Libre Software developers, 2001, http://widi.berlios.de/paper/study.pdf,

    Alexander Hars and Shaosong Ou, Working for Free? Motivations for Participating in Open-Source Project, 2002, http://search.ebscohost.com/login.aspx?direct=true&db=epref&AN=IJEC.F.BE.HARS.WFMPOP,

    Rishab Aiyer Ghosh, Understanding Free Software Developers: Findings from the FLOSS Study, 2005, http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11216&mode=toc and

    Karim R. Lakhani and Robert G. Wolf, Why Hackers do what they do: Understanding Motivation and Effort in Free/Open Source Software Projects, 2005.
12. Georg von Krogh and Sebastian Spaeth and Karim R. Lakhani, Community, joining, and specialization in open source software innovation: a case study, 2003, http://www.sciencedirect.com/science/article/B6V77-48WB8M0-1/2/92b48386d215bdf3884018f4f6172043.
13. E. g. the Code of Conduct from the KDE community: KDE, KDE Community Code of Conduct, 2010, http://kde.org/code-of-conduct.
14. Elliot Aronson and Timothy D. Wilson and Robin M. Akert, Sozialpsychologie, 2004 and

    Günther Bierbrauer, Sozialpsychologie, 2005.
15. These experiments are for the reader. Therefore the style of writing of the next paragraphs will address the reader directly (“you”-form) instead of the more scientific third person.
16. In accordance with Angela Byrons' quote „Create a great community and great code will follow.“
    Angela Byron, Lessons on Community Management from the Open Source World, 2009, http://www.osbr.ca/ojs/index.php/osbr/article/view/890/860.