The Five Stages of Open Source Volunteering

Dirk Riehle

Abstract Today's software systems build on open source software. Thus, we need to understand how to successfully create, nurture, and mature the software development communities of these open source projects. In this article, we review and discuss best practices of the open source volunteering and recruitment process that successful project leaders are using to lead their projects to success. We combine the perspective of the volunteer, looking at a project, with the perspective of a project leader, looking to find additional volunteers for the project. We identify a five-stage process consisting of a connecting, understanding, engaging, performing, and leading stage. The underlying best practices, when applied, significantly increase the chance of an open source project being successful.

1 Introduction

Open source software has become an important part of the Internet and today's enterprises. There is little software left that does not at least include some open source components. Thus, understanding how open source projects work and how to utilize them is a critical capability of software product companies who wish to crowd-source some of their development work.

Open source software projects can be split into community open source and commercial open source software projects [1, 2]. Community open source software is software that is owned by a community, typically by way of distributed copyright ownership or by ownership through a non-profit foundation. Commercial open source software development is curated by a single company, which maintains the ownership of all relevant intellectual property. According to Mickos, commercial open source rarely receives and incorporates code contributions from their user communities [3]; however, community open source does.

D. Riehle (✉)
Computer Science Department, Friedrich-Alexander-Universität Erlangen-Nüremberg, Erlangen, Germany
e-mail: dirk@riehle.org; dirk.riehle@fau.de

W. Li et al. (eds.), *Crowdsourcing*, Progress in IS,
DOI 10.1007/978-3-662-47011-4_2

Community open source software projects rely on volunteer work to a (significant) extent. Projects which are more mature and relied upon by companies may gain commercial support, which reduces reliance on volunteers [4]. Commercial support can take the form of direct financial contributions, which allows foundations to acquire employees. Also, companies may assign their own employees to contribute to the project. Leaders of new and small community open source projects cannot expect commercial support, and must learn how to recruit and retain volunteers if they want their project to grow.

In this article we review best practices of open source community management, specifically, how to find, keep, and grow volunteers. The article is based on a literature review and observation of existing open source projects. We identify five stages of the open source volunteering process which we call the connecting, understanding, engaging, performing, and leading stages. For each stage, we discuss the best practices of the actors of that stage. In addition, we discuss the underlying guiding principles that we found to be common to all the practices.

Thus, this article makes the following contributions:

- It defines guiding principles underlying the volunteering process;
- It presents a five-stage model of the open source volunteering process;
- It collects and catalogs best practices applicable to each stage.

The article is structured as follows. Section 2 presents the guiding principles, Sect. 3 introduces the five-stage process and discusses its properties. Section 4 walks through the stages in detail, discussing best practices and supporting tools. Related work as relevant to the different sections is discussed in place. Section 5 concludes the article.

2 Guiding Principles of Open Source Projects

In reviewing the literature (as referenced in place) and working with open source communities as well as from prior work we identified the following three guiding principles project leaders need to understand for an effective recruiting process:

- Recruiting is Investment [5, 6]
- Open Communication [5, 6]
- Open Collaboration [5, 7]

We discuss these principles in turn.

2.1 Recruiting is Investment

According to Fogel, every interaction with a user is a chance to recruit a new volunteer [6]. At the same time, according to Fitzpatrick and Collins-Sussman, the scarcest

resource that a project has is attention and focus [8]. In combination, this leads to the primary guiding principle of open source volunteer recruiting:

- **Recruiting volunteers is a project investment**

Recruiting volunteers is obviously necessary for a project to grow. However, the time spent on recruiting takes attention and focus away from actual software development. Spending time on recruiting should therefore be viewed as an investment to be made wisely.

Investments may or may not work out. The time spent on a potential volunteer may or may not be wasted. Thus, if time is spent on recruiting, it should be spent well, and it should be spent in a form commensurate with the likelihood of success, in this case, of finding a new volunteer. The best practices of Sect. 4 all embed this principle.

2.2 Open Communication

Open source projects follow a particular style of communication which helps support a distributed volunteer community. Fogel, for example, argues that communication styles portray project members (who may never have met in person) to each other [6]. He argues for general principles (all communication should be public) and very specific principles (no conversations in the bug tracker). Using this and other sources, we derived the following four fundamental maxims of open communication that characterize open source projects and the way project members communicate with each other and potential volunteers:

- **Public**. All communication should be public and not take place behind closed doors; any private side-communication is discouraged.
- **Written**. All communication should be in written form; if this is not possible, any relevant communication should be transcribed or summarized in writing.
- **Complete**. Communication should be comprehensive and to the extent possible, complete. Assumptions are made explicit and key conclusions are summarized.
- **Archived**. All communication should get archived for search and later public review. Thus, previous conversations are available for posterity.

Taken together, these maxims create transparency and discipline communication, leading to more effective distributed collaboration. Although not all communication within a project will embody all four maxims, a project which is motivated to be transparent and grow its community will make the according effort, for example, by transcribing or summarizing non-email forms of communication in order to provide a public archive.

Public communication ensures that all members of the community have the opportunity to participate, which creates buy-in and trust. Written communication enables asynchronous, distributed work. People who are less fluent in the language used for communication also benefit from having additional time to absorb the meaning [9],

which makes the project accessible to a wider audience. Complete communication reduces opportunities for misunderstanding and ensures that the community shares a common understanding of objectives. Archiving increases transparency by ensuring that decisions can be understood in context.

Many of these concepts reinforce one another. Writing enables archiving, as current search technology is text-based. The need to derive meaning from archives encourages more complete communication. Archives are more comprehensive and comprehensible if all conversation is public.

2.3 Open Collaboration

No two open source projects follow the same software development process. However, in prior work and by way of project reviews, we identified three underlying fundamental components of open source collaboration [7]. These three maxims of open collaboration are:

- **Egalitarian**. Everyone may join a project, no principled or artificial status-based barriers to participation exist.
- **Meritocratic**. Decisions are made based on the merits of the arguments, and status is determined by the merits of a person's contributions.
- **Self-organizing**. Processes adapt to people rather than people to processes.

Related work frequently subsumes the first two maxims under the single concept of meritocracy. We find it helpful to distinguish between them: Openness in the context of egalitarianism means that all people have the opportunity to participate, whereas openness in the context of meritocracy ensures that all work is evaluated on the basis of its intrinsic value.

These concepts are in stark contrast to traditional work inside companies. Projects are not egalitarian: Employees are assigned to work on them and cannot choose to work on other projects. Decisions are not necessarily made on the basis of the merit of the arguments, but are ultimately the choice of the person with the greatest power. Finally, processes in a large company are typically defined by a central department and employees are expected to adapt their work processes to the company environment.

Open source projects are different: It is recognized that any potential volunteer could become a valuable resource. Thus, an effective project process must be open to accepting volunteers (egalitarianism), must recognize quality regardless of the source (meritocracy), and allow processes to develop according to the needs of the community (self-organizing). The five stage process of open source volunteering described in the following Section is based on the three guiding principles of the open source volunteering process: recruitment is investment, open communication, and open collaboration.

3 The Five-Stage Volunteering Process

In [10], Behlendorf illustrates a typical example of how a developer might join a project, rise through the ranks, and become a project leader. The developer

1. needs to solve a problem,
2. searches the web for appropriate software,
3. finds a matching project,
4. checks out the project,
5. gives the project a try and is happy,
6. finds a bug and reports it,
7. makes a first contribution,
8. engages in a conversation,
9. keeps contributing,
10. receives a vote of trust, and
11. ultimately leads the project.

By correlating this 11-step process with Fogel's work [6] and by aligning it with the Onion model of roles in open source software development [11], we were led to a simpler and denser five stage model of the open source volunteering process than the one proposed by Behlendorf. This model is shown in Fig. 1.

An innovation of this model is the addition of the two complementary views of volunteer and project (leader), which lead to complementary but mutually supporting best practices and activities. The stages are defined in the following way:

- **Stage 1: Connecting**. In this stage, a potential volunteer stumbles over a project by lucky chance or, after searching for something like it, finds the project through a search engine. The project needs to prepare for this to happen, which requires marketing itself through appropriate channels and at appropriate portals.
- **Stage 2: Understanding**. In this stage, once a potential volunteer is looking at a project's website, the website needs to draw him or her in. Using a variety of best practices, the project helps the visitor quickly understand what the project is about and whether it should be of interest to them.

#	Stage	Volunteer View	Volunteer Role Name	Project View
1	Connecting	Find project	Seeker	Market project
2	Understanding	Understand project	Visitor (Reader)	Explain project
3	Engaging	Engage with project	User	Engage with user
4	Performing	Work within project	Contributor	Work with contributor
5	Leading	Lead project	Leader	Enable career

Fig. 1 A five-stage model of the open source volunteering process

- **Stage 3: Engaging**. In this stage, a potential volunteer is inspired to engage with the project, for instance by installing the software or joining a mailing list. The project strives to welcome users to the community and direct them toward the next stage by providing information of simple ways to volunteer.
- **Stage 4: Performing**. In this stage, a volunteer contributes to the project. The project community needs to be receptive to initial efforts by reacting quickly to contributions and creating conversations to improve quality. The project needs to guide users towards becoming regular contributors.
- **Stage 5: Leading**. In this stage, the volunteer accepts responsibility for the direction of the project or community. The project must have a mechanism for identifying potential leaders and making decisions on their promotion to leader status as well as for communicating this clearly.

Not all volunteers pass through all stages, but stages can only be taken one after another. Each subsequent stage will be reached by fewer volunteers. The best practices described in the next section support each stage. For a project, they help increase volunteer commitment. When recruitment is viewed as an investment, best practices are aligned with promoting long-term involvement. For a volunteer, best practices advise on how to achieve goals, from finding a project that fulfills a need to gaining recognition within a project. We now describe each stage in detail, along with selected best practices from each perspective and tools to support them.

4 Best Practices and Supporting Tools

A best practice "is a broadly-accepted, typically informally-defined, method for achieving a particular goal that is considered superior to most other known methods" (author's adaptation of the Wikipedia entry on "best practice" [12]). Thus, a best practice is a method reflecting the state-of-the-art as applicable in a particular context.

The following best practices have been derived from the respective references, in particular [6, 8, 10, 13–15]. They have been grouped according to the phases described in Sect. 3 and divided into the two perspectives described there: the volunteer's view and the project leader's view. They are based on the application of the three principles of open source volunteering described earlier in Sect. 2. Due to the large number of sometimes mundane best practices, not all are discussed in detail.

The principle which informs all best practices from the project view is that of recruitment as an investment. In a project, time is the scarcest resource, and recruitment takes time. Increasing the long-term return on time invested [6]—or encouraging volunteers to move to each successive phase—is therefore the objective of the project's leadership. Open communication and open collaboration are also reflected in the best practices; they are the underlying tenet that make open source projects work.

A volunteer is not a passive subject to be recruited, but an individual with objectives in mind. At each phase, a volunteer wants to ensure maximum value for the investment, which is where best practices come into play. The volunteer who is prepared will achieve better results than one who fails to consider the project's needs.

Volunteer (Seeker) View	Project View
Stumbling	**Active Outreach**
• Stumble upon project • Follow word-of-mouth	• Choose a good name • Define relevant channels • Use channels consistently • Announce visibly • Be matter of fact
Searching	**Passive Inflow**

• Use general search engine
• Search on open source portal
	• Register on all portals
	• Support search engines
	• Support lucky chance
	  ◦ Use portal features
	  ◦ Provide findable summaries
	  ◦ Work towards portal metrics

Fig. 2 Best practices of Stage 1, the Connecting stage

4.1 Stage 1: Connecting

Figure 2 displays all best practices of Stage 1, the Connecting stage. Volunteers can be separated into two categories, those that stumble onto the project by luck, and those that search for a solution to a problem they have. Project best practices can be split into active outreach being performed and passive inflow that needs to be prepared for. Two terms stick out among the project best practices, channel and portal:

- An open source project channel is a communication channel for a project to reach potentially interested parties, in particular volunteers. Examples of such channels are:
  - Social media channels like Facebook or Twitter
  - Targeted communication channels like Slashdot or Hacker News
  - Specific open source conferences like OSCON or ApacheCon
- An open source project portal is a portal website dedicated to open source projects. Examples of such websites are:
  - Project hosting sites like SourceForge or Github
  - Meta-sites like Freshmeat or Open Hub

Channels are mostly used for active outreach and when the project has a story to tell, for example, the initial release. Portals are used for passive inflow where searchers can find them when they are seeking a solution.

A project should choose a good name that is easy to remember and ideally indicative of the project's purpose. As an alternative to descriptive names, wholly artificial

names may serve the project equally well. Any communication then should stick to that name and use it consistently. The relevant channels and portals (see above) need to utilized repeatedly, consistently and predictably. Any communication should be matter-of-fact rather than hyperbole-projects are trying to create a long-term reputation, not a short spike of attention followed by disappointment over the hyperbole. The most common form of communication is the announcement of new releases of the software, followed by announcements over major developments in the project community or sponsorship.

4.2 Stage 2: Understanding

In the second stage, the emphasis is on communicating the project's purpose to a volunteer who wants to quickly learn if the project answers his or her need. Figure 3 lists relevant best practices.

Various pieces of information need to be easily accessible, both in terms of finding and understanding the information. A first step is to have a clear mission statement that spells out the project's purpose and does so in a highly visible place, for example, the front page of the project's website on a software forge. Examples and screen-shots should be easily accessible to make it straightforward for visitors to assess what the software does in practical and tangible terms (short of downloading and installing the software, which would be the next step). Words are only so good-examples and screen-shots sometimes communicate more clearly.

Many visitors will also want to know about related project information like software licenses or (assumed) quality of the software (by way of development status). Thus, a project should display prominently which open source license it is using, what state of development it is currently in, and what future expected developments are, including upcoming releases and key new features and functionalities. The

Volunteer (Visitor) View	Project View
Reading Up	**Explain Project**
• Read up on project	• Have clear mission statement
	• Provide examples and screen-shots
	• State license and terms
	• Provide simple downloads
	• Show current and future releases
	• Show development status
	• Provide user documentation
	• State whether volunteers are welcome
	• State rules of engagement

Fig. 3 Best practices of Stage 2, the Understanding stage

visitor, who wants to try the software, may need user documentation, which should therefore be provided.

Visitors have questions or may want to become volunteers, and hence a project is well advised to spell whether volunteers are welcome and what the project rules are so that someone considering to participate will know what they are getting into.

4.3 Stage 3: Engaging

In the engagement stage, the emphasis is on facilitating communication between the project and the volunteer. Figure 4 lists relevant best practices.

It needs to be clear (and clearly displayed) how current project members can be reached. At this stage, the project may only be perceived as an anonymous entity with no particular face. Potential volunteers need starting points, for example, forums or mailing lists where they can ask questions.

A first response should be welcoming of a new potential volunteer, and any possible rudeness, whether incidental or deliberate, needs to be stopped immediately. It is paramount that any project member redirects any privately posed questions to a public forum and avoids answering questions in private; this would be a highly inefficient use of their time. Visitors need to understand that they consume time and hence should do their homework or should be guided to do their homework before asking. Doing one's homework implies reading existing materials to avoid redundant questions. Also, visitors have to learn to ask in public so that everyone can

Volunteer (User) View	Project View
All Volunteers	**Towards all Volunteers**
● Do your homework before asking	● Show how to reach project
◦ Search archives	● Welcome to community
◦ Read documentation	● Stop any rudeness
● Communicate prudently	● Avoid private discussions
◦ Be matter of fact	● Accept initial redundancy
◦ Don't jump to conclusions	● Provide simple tasks
	● Provide incremental tasks
	● Call out lurkers from the shadows
Software Developers	**Towards Software Developers**
● Respect project practices	● Provide tool access
● Respect project culture	● Remove arbitrary tool obstacles
● Accept guidance	● Show requirements list
	● Provide developer guidelines
	● Provide developer documentation

Fig. 4 Best practices of Stage 3, the Engaging stage

learn from their considerations and questions. Communication, both on the visitor and the project side, should be matter of fact and content focused, trying to help solve the problem or question at hand.

For more advanced visitors, or users of the software, it should be possible to learn about simple tasks that the project would benefit from. The project should spell out such tasks, even if writing them down may cost nearly as much time as performing them, because simple tasks provide a mechanism to engage volunteers. Similarly, there should be incremental tasks to be picked up, which will allow volunteers to work with existing developers rather than alone. Incremental tasks also introduce volunteers to existing technical aspects of the project.

A lot of things can go wrong when setting up a project for engaging potential volunteers, and appropriate attention needs to be paid so that tools and project artifacts like task and requirements lists are accessible, and that developers can find appropriate guidelines and documentation.

Underlying all these project best practices is the guiding principle of making it as easy as possible for a volunteer to make a first contribution. Getting to that first contribution is the single most important hurdle a project has to overcome. Thus, many of the best practices work hand-in-hand to make that first contribution happen.

4.4 Stage 4: Performing

The performance stage is when the volunteer contributes to the project. It can be subdivided into a first contribution and later more regular contributions. Figure 5 lists appropriate best practices for project leaders.

A volunteer's first contribution is like dipping a toe into the water. Depending on how the experience feels, the volunteer may not come back. Thus, it is important to ensure that this first contribution becomes a positive experience. For one, a contribution should be well received and reacted to. Nothing is worse than no reaction, for example, by letting a patch sit idle. The appropriate reaction to a patch is to turn it into a conversation, not only to say thanks, but also to encourage further contributions by pointing the volunteer to related issues. In all but the most simplest patches or contributions, the volunteer may have to be guided to reworking the contribution, for example, to ensure compliance with the project's programming guidelines. Code review of a patch submission is a general best practice, but also shows the volunteer that their contribution is being taken serious, even if it leads to a request to fix a problem with the submission. Finally, after a successful contribution, it is critical to pay credit to who credit is due and list the volunteer as a contributor to the project.

Volunteers who have become regular contributors may then be willing to pick up other tasks outside their original interests. Still, project leaders should track and play to volunteer interests when asking them for help, for example, to work on a particular feature. Volunteers, who have bought into the project are frequently willing to pick up work that they originally did not join the project for. This includes unloved tasks like project documentation and is not restricted to technical tasks alone.

Volunteer (Contributor) View	Project View
General Practices	**General Practices**
• Contribute	• Be component-oriented • Work from features • No discussions in bug tracker • Decide using consensus • Vote as a last resort
First Contributions	**Towards First Contributions**
• Contribute	• React speedily, don't sit on patches • Turn contributions into conversations • Practice conspicuous code review • Track contributions, provide credit • Praise plentifully, criticize specifically • Prevent territoriality
Regular Contributions	**Towards Regular Contributions**
• Contribute	• Track interests, assign accordingly • Distinguish inquiry from assignment • Share technical and managerial tasks • Follow-up on delegated assignments • Document practices and traditions • Archive practice descriptions

Fig. 5 Best practices of Stage 4, the Performing stage

A project leader who asked a contributor to perform some work and received a commitment needs to fulfill a managerial role now. For example, if the contributor is not providing the promised feature, the project leader may have to inquire about progress, nudging the contributor along (and making mental notes as to whether this was a good request that matched the volunteers interests). Sometimes, a project may run into difficult people. "Difficult" or even "poisonous" people, according to Fitzpatrick and Collins-Sussman, may waste a projects time or split and even ruin a project [8]. Best practices to prepare for the problem are to

1. build a healthy community and
2. document all decisions.

It is necessary then to detect the problem: Difficult people typically don't show respect, miss social cues, are overly emotional, and make sweeping claims not based on any data. Best practices to handle the problem are to

1. not engage them,
2. ignore them if possible,
3. remove them from the project if necessary.

General engineering management advice applies as well. The system software architecture needs to match its social structure, which typically implies a well-componentized structure so that developers can work independently of each other and in a distributed fashion. The requirements and open tasks in contrast should be feature-oriented, as completing a feature is a major motivation for a developer because it provides meaning to the work being performed.

Unlike in traditional (non-open) contexts, however, the principles of open communication and open collaboration need to be maintained. This requires appropriate consensus-oriented and merit-based discussion as to decisions to be taken. Voting to make a decision is a last resort to resolve a conflict and should be used rarely.

4.5 Stage 5: Leading

In the final phase, the volunteer becomes a project leader and takes responsibility for the best practices of the project view for the earlier phases. Figure 6 shows some of the best practices at this level.

At this stage, a project leader is basically a manager, but without the power found inside traditional organizations. He or she has to rely on the power of persuasion and goodwill that contributors have developed towards the project. With increasing commercialization and paid-for participation in open source, some of these challenges become less serious, and developers may need less intrinsic motivation. Still it remains good practice to personally motivate developers through their work beyond the possible salary that an employer may be paying for their open source work.

The power of leadership rests on setting a good example by taking responsibility and acting accordingly, by praising other people's work and acknowledging their contributions. Additional actions may be necessary to help contributors outside the project, for example, if they are performing open source work on company-time

Volunteer View	Project View
General Practices	
• Take responsibility	
• Praise other contributors	
• Acknowledge contributions	
• Support others towards their manager	
• Provide tokens of appreciation	
• Define community career	
• Define roles and positions	
• Decide promotion privately	
• Announce promotion clearly	

Fig. 6 Best practices of Stage 5, the Leading stage

without the employer having a particular interest in the project. Then, the open source project leader may have to help motivate why the developer's work ultimately benefits his or her employer.

The open source project itself needs management in that contributors find the work formally acknowledged in the form of traditional credits. Contributors are also having a form of open source career. Taking steps in this career, most notably from contributor to committer, may be touchy subjects, and are one of the few discussions that the existing project leaders may have to decide privately and not in the public eye. This is justified, because such a discussion is typically more about the social aspects of working with the to-be-promoted person rather than his or her technical capabilities. In case of a positive decision, the promotion needs to be announced publicly and documented accordingly so that everyone in the project knows.

5 Conclusion

This article first identified the three guiding principles of open source projects. Volunteers are the lifeblood of an open source project, and an effective recruiting process considers all three principles.

First, recruiting is an investment: time and effort are invested in order to yield long-term results. Second, open communication facilitates the volunteer process by creating transparency. Third, open collaboration opens the recruitment process to any potential volunteer and allows them to contribute to their fullest extent.

A model of five phases of engagement is presented. This model looks at the different levels of volunteer commitment from both the perspective of the volunteer and of the project.

The three volunteering principles are used to advance a number of best practices which are tied to the five stages of engagement. At each phase—connecting, understanding, engaging, performing and leading—there are objectives and best practices for both the volunteer and the project, as represented by its leadership. The best practices are derived from existing literature and observation.

Acknowledgments I would like to thank Ann Barcomb and the anonymous reviewers for helpful comments that improved the paper.

References

1. Riehle, D.: The economic motivation of open source software: stakeholder perspectives. Computer **40**(4), 25–32 (2007)
2. Riehle, D.: The economic case for open source foundations. Computer **43**(1), 86–90 (2010)
3. Mickos, M.: Open for business: building successful commerce around open source, (2010)
4. Riehle, D. Riemer, P. Kolassa, C. Schmidt, M.: Paid vs. volunteer work in open source. In: 47th Hawaii international conference on system sciences (HICSS), pp.3286–3295, January 2014

5. Riehle, D.: The open source knowledge sharing and volunteering process, (2011)
6. Fogel, K.: Producing Open Source Software. O'Reilly, Farnham (2005)
7. Riehle, D., Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi, Y., Naveh, B., Odenwald, T.: Open collaboration within corporations using software forges. IEEE Softw. **26**(2), 52–58 (2009)
8. Fitzpatrick, B., Collins-Sussman, B.: How open source projects survive poisonous people, (2008)
9. Carmel, E., Tija, P.: Offshoring Information Technology. Sourcing and Outsourcing to a Global Workforce. Cambridge University Press, Cambridge (2006)
10. Behlendorf, B.: How to contribute to open source projects, (2011)
11. Crowston, K., Howison, J.: The social structure of free and open source software development. First Monday **10**(2) (2005). First Monday, Special Issue # 2: Open Source—3 October 2005 The social structure of free and open source software development (originally published in Volume 10, Number 2, February 2005)
12. Wikipedia. Definition of Best Practice
13. Bacon, J.: The Art of the Community. O'Reilly, Farnham (2012)
14. Delacretaz, B.. Open source collaboration tools are good for you, (2009)
15. Gabriel, R., Goldman, R.: Innovation Happens Elsewhere. Elsevier (2005)