

Open Source and Agile: Two worlds that should have a closer interaction

Hugo Corbucci¹ and Alfredo Goldman¹

¹Instituto de Matemática e Estatística (IME)
Universidade de São Paulo (USP) - Brazil

{corbucci, gold}@ime.usp.br

1. Introduction

Open source projects usually receive the collaboration of many geographically distant people that don't share any organizational structure. At first, these arguments could indicate that such projects are not candidates to the use of agile methods since some basic values seem to be damaged. For example, the distance and diversity separating developers surely deteriorate communication, one of the most important values within agile methods. However, most open source projects share principles with the Agile Manifesto [1]. To be ready for changes, to work with continuous feedback, to deliver real features, to respect collaborators and users and to face challenges are expected attitudes of an agile developers naturally found in Free and Open Source Software (FOSS) communities.

During OOPSLA 2007, a workshop [4] about "No Silver Bullets" [2] mentioned Agile methods and Open Source Software Development as two failed silver bullets having both brought great benefit to the software community. The same workshop asked if the use of several failed silver bullets simultaneously could not, in fact, raise production levels in an order of magnitude. This paper is an attempt to start this sort of merging to stun the werewolf, that is, stop problems from appearing out of nowhere.

Section 2 will present some aspects of major open source communities that could be improved with agile practices and principles. The next one (Section 3) will focus on the problems agile methods face when dealing with distributed teams and scaling to big teams which are somehow already addressed in open source development. Finally, Section 4 will present the work planned and being done.

2. Is Open source Agile?

Open source communities could almost be considered agile and they indeed were by Martin Fowler in his first version of "The New Methodology" [3]. The methods that Eric Raymond describes in "The Cathedral and the Bazaar" [5] lack of a more precise definition but several ideas could be related to the Agile Manifesto. The next four subsections will discuss the relations of open source to the four points principles of the manifesto and the fifth one will summarize points where open source could improve towards agility.

2.1. Individuals and interactions over processes and tools

Projects usually have their processes including feature freezing, version branches, commit reviews and several other good practices or rules. Most of the time, tools are used to enable those practices and others are present and widely used. Several of the tools used on open software process are also used on agile software development, such as version

control programs. **The processes and tools** are, however, just ways to achieve a goal: ensuring a stable and welcoming environment to create software collaboratively.

Although open source businesses are growing stronger, the very essence of a community is to have **individuals that interact** in order to produce what interests them. In those communities, the interaction is usually related to source code collaboration and documentation elaboration. Those activities are responsible for driving the whole process and modifying the tools to better fit their needs.

2.2. Working software over comprehensive documentation

A lot of open source projects are heavily criticized about their documentations or the lack of it. This comes from the fact that most developers are not committed on writing documentation. More likely, they prefer to have a neat software that is intuitive for users. The result is that new projects hardly have any sort of documentation except the minimum required for the own developer team to be able to work.

Comprehensive documentation grows with the community that builds around the **working software**, as users encounter problems to complete a specific action. It is frequent to have documentation written by volunteer users to help their colleagues. This work generates documents in a language that users understand but that only deal with common problems. Specific problems and solutions are much harder to find.

2.3. Customer collaboration over contract negotiation

Contract negotiation is still only a problem to very few open source projects since a huge number of them do not involve contracts. On the other hand, those involving contracts are usually based on a service concept in which the customer hires a programmer or company to develop a certain feature for a small amount of time. Although this business model does not ensure that the customer will collaborate, it may shorten time between conversations, therefore improving feedback and reducing the strength of long and rigid contracts.

The key point here is that collaboration is the basis of open source projects. The customer is involved as much as he desires to be. **Customers can collaborate** but they are not especially encouraged or forced to do so. This might be related to the small amount of experience this communities has with customer relationships. However, several successful projects rely on fast answers to demanded features from users. In this case customer collaboration allied with responsiveness are specially powerful.

2.4. Responding to change over following a plan

Open source projects tend to have a plan of milestones or releases but, in most cases, those plans are always short term plans. Even when long term plans exist, they are not the main guidelines followed by the developer team. They are only goals sought without any pressure to be met.

Being too demanding about **following a plan** can drag a whole project down in the open source world. The main reason is the highly competitive environment of this universe where only the best projects survive. The **ability of each project to adapt and respond to changes** is crucial to determine those who survive. No marketing campaign or business deal can save a project from abandonment if it cannot compete with a newcomer that adapts more quickly to user needs.

2.5. What is missing on open source?

Although several points of the Agile Manifesto are followed within open source communities, nothing is certain because there is no such thing as an open source method. Raymond's description is a great example of how the process can work but it does not discriminate guidelines and practices to be followed. If a full open source agile method description was written with the use of compatible tools merging the ideas presented by Raymond, it would follow the same selection rules as the projects. If successful, its adoption would then spread around the community improving and correcting it over time.

Communities created around FOSS projects involve users, developers, and sometimes even clients working together to craft the best software possible. The absence of such community around a program usually denounces a recent project or one that is dying. This means that the development team must be very attentive to this community since it shows how well the project is going. Nowadays, concerns related to this aspect of FOSS development are not specifically considered by the most known agile methods.

3. Agile going Open source

Not only, as previously shown, agile methodologies lack some special solutions related to open source development, but they could also benefit from solving some of the issues FOSS project suffer from. Section 3.1 will describe with more details some specific points of FOSS development that would need a specific approach. The next one (Section 3.2) will shortly present some benefits that agile would receive from attempts to solve those problems.

3.1. Agile helping FOSS

Agile development so far has been described as a way to develop software within companies with contracts and employees. Forming and maintaining a community bounded to the system is the responsibility of the marketing and sales people. As long as the contract exists, there is no danger regarding the adoption of the project and its user base. In a FOSS project, none of those factors are ensured at any moment. Even if there is a contractor and there are employees, the community must be kept active and welcoming. Addressing a community, responding to its requirements and providing feedback to its members is not an easy task. What, when and how to provide feedback must be wisely chosen and is a time consuming activity that cannot be undertaken.

- How to balance between customer requirement and community requirement?
- How should providing feedback be handled within an iteration?
- Should plans be made counting on external help or not?
- If so, how to estimate expectations about outside contributions?
- What tools or measures should be used to make it easier for people to contribute?
- How should commits be approved or denied?

Those are only a few questions that are unanswered when dealing with open source communities using an agile method. This work intends to provide a wider analysis of those issues to gather a more complete and precise list of issues related to open source development that agile methods do not provide answers for.

3.2. Agile contributions improving itself

Most of the problems pointed out before are related to communication issues triggered by the amount of people involved in the project and their various knowledges and cultures. Although in open source those matters are taken to a limit, distributed agile teams face some of the same problems. Evolving a software that will be used by many people around the world with slightly different processes and laws may require distributed agile teams working geographically distant with specific local clients.

As the current situation of Internet makes evident, users are becoming more and more important to the success or failure of a system. In such perspective, providing feedback and absorbing suggestions and critics will become essential to survival of a project. Just like the ability to adapt placed agile methods to a very important position, the ability to receive, select and incorporate suggestions from communities will probably make the difference in the near future. According to its own principles, agile methods should respond to those changes and adapt to this growing matter. The best place to start such work is within an extreme community such as open source.

4. Conclusion

In this preliminary work we have shown several evidences that a synergy with agile methods can improve software development on FOSS projects. Several already adopt some agile techniques to be more responsive to users but a complete description of a method that considers all FOSS factors would surely increase adoption in those communities. On the other hand, solving the problem is a challenge that would consolidate agile methods to a distributed environment relying on a large user community.

As part of this work, two surveys are planned. One to be conducted at FISL (International Free Software Forum) 2008 to understand how much open source developers and enthusiasts know about agile methods and what keeps them from using them. The other one to be conducted at Agile 2008 will try to discover how involved is the agile community with open source development. Both surveys will be used to provide a deeper understanding of the interaction between both communities and how to improve it. Also, interviews with leaders of both communities could help address more specific topics and gather suggestions and support for the results of this work.

References

- [1] Kent Beck, Alistair Cockburn, Ward Cunningham, Martin Fowler, Ken Schwaber, and al. Manifesto for agile software development. <http://agilemanifesto.org/>, 02 2001. The agile manifesto official web site.
- [2] Frederick P. Brooks, Jr. No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19, April 1987.
- [3] Martin Fowler. The new methodology. <http://martinfowler.com/articles/newMethodologyOriginal.html>. Original version on Fowler's website.
- [4] Dennis Mancl, Steven Fraser, and William Opdyke. No silver bullet: a retrospective on the essence and accidents of software engineering. In *OOPSLA Companion*, 2007.
- [5] Eric. S. Raymond. The cathedral and the bazaar. <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>, 08 1998.