Is Open Source Software Development Essentially an Agile Method?

Juhani Warsta[a] and Pekka Abrahamsson[b]
[a]*Department of Information Processing Science*
P.O.Box 3000, FIN-90014 University of Oulu, Finland
[b]*Technical Research Centre of Finland, VTT Electronics*
P.O.Box 1100, FIN-90571 Oulu, Finland
Juhani.Warsta@oulu.fi, Pekka.Abrahamsson@vtt.fi

Abstract

It has been argued that Open Source Software (OSS) development differs from the agile software development mode in philosophical, economical, and team structural aspects. This paper investigates the OSS development characteristics from four perspectives: process, roles and responsibilities, practices, and scope of use. The study shows that while from a legal perspective the OSS development paradigm could be seen more as a licensing structure exploiting the terms of the General Public License or similar, the OSS does in many ways follow the same lines of thought and practices as the main stream of existing agile methods. The principal differences and similarities are highlighted and discussed. It is suggested that the OSS community could benefit from the practical solutions put forward by the agile proponents and vice versa.

1. Introduction

Successful projects during the past years have demonstrated the fact that the Open Source Software (OSS) development paradigm is an important and valuable complement to the manifold existing software development methods [1]. During the same period also the agile movement has started to gain ground in several different manifestations. Our purpose in this paper is to discuss whether these methods have something in common and could they benefit from each other?

Agile – denoting "the quality of being agile; readiness for motion; nimbleness, activity, dexterity in motion" (*http://dictionary.oed.com*) – methods represent new approaches in the spectrum of software development methods. The aim of these practitioner-oriented software development methods is to make a software development unit more responsive to changes. These changes are imposed by rapidly evolving technology, changing business and product needs [2]. This is especially the case with the rapidly growing and volatile Internet software industry as well as with the nascent mobile application environment.

In this paper we will analyze the OSS development through an agile development perspective and we will show that OSS and agile development methods have many similarities. Much research effort is put into explicating the contradiction of the success of the OSS projects. How does it function, as there is no formal OSS method, but still the organic group of developers is usually able to produce well functioning software?

The rest of the paper is composed as follows. The next section presents a short overview of the OSS development paradigm. In the third section we analyze the OSS approach from the perspectives of process, roles and responsibilities, practices, and scope of use. In the fourth section we discuss the characteristics of OSS and agile software development. The final section recapitulates the key findings.

2. Background: The Open Source Software paradigm

The significance of the Open Source Software development paradigm has strongly grown from the days when Richard Stallman started his work on GNU UNIX back in 1980's [3]. The establishment of the Free Software Foundation in 1985 was a landmark for the OSS movement as the basic ideas of controlled but free usage of software were laid down. Later, the real impetus was given by the technological improvements in development that made it possible to combine the invention of the bulletin board system and the old custom of software developers to share code freely among colleagues. This was intensified and made possible on a global scale by the expansion of the Internet in the '90s. This development furthered the novel software development paradigm of OSS, offering an innovative way to produce applications. This further aroused growing interest along with ample research efforts and discussions concerning the possibilities and meaning of the OSS approach for the whole software industry. This interest has notably increased after several success stories; among these are the widespread Apache server, the Perl programming language, the SendMail mail handler, and especially the ubiquitous Linux operating system [4]. Microsoft has even pronounced the latter as their toughest competitor in the server operating systems market.

The OSS approach suggests the source code to be freely available for modifications and redistribution without any charges. The OSS paradigm can also be discussed from a philosophical perspective [5, 6]. Feller and Fitzgerald [7] present the following motivations and drivers for OSS development:

1. Technological; the need for robust code, faster development cycles, higher standards of quality, reliability and stability, and more open standards and platforms
2. Economical; the corporate need for shared cost and shared risk
3. Socio-political; scratching a developer's "personal itch", peer reputation, desire for "meaningful" work, and community oriented idealism.

Depending on the point of view - do you stand for the business community or are you an OSS devotee - these motivations can be insignificant.

Most known OSS development projects are focused on development tools or other platforms that are used by professionals who have often participated in the development effort themselves, thus having the role of the customer and that of the developer at the same time. OSS is not a compilation of well defined and published software development practices constituting an eloquent, written method. Instead, it is better described in terms of different licenses for software distribution and as a collaborative way of widely dispersed individuals to produce software with small and frequent increments. The Open Source Initiative [8] keeps track of and grants licenses for software that complies with the OSS definitions.

3. Some perspectives of the OSS paradigm

The purpose of this section is to analyze the OSS paradigm from several perspectives: process, roles and responsibilities, practices, adoption and experiences, and scope of use. A definition of the perspectives chosen can be found in [9]. While the focus is on the OSS, some clear differences or similarities with regard to agile methods are also disclosed.

Process

Cockburn [10] notes that OSS development differs from the agile development mode in philosophical, economical, and team structural aspects. However, OSS does in many ways follow the same lines of thought and practices as many agile methods. For example, the OSS development process starts with early and frequent releases, and it lacks many of the traditional mechanisms used for coordinating software development with explicit plans, system level designs, schedules and defined processes. Typically, the OSS project consists of the following visible phases [11]: problem discovery, finding volunteers, solution identification, code development and testing, code change review, code commit and documentation, and release management.

Even though it is possible to depict the idea of OSS development with the above iteration stages, still the focal interest lies in how this process is managed. Mockus et al. [12] suggest that the following factors depict the challenges in the OSS development process that must be tackled in a proper manner:

1. The systems are built by potentially large numbers of volunteers.
2. Work is not assigned; people themselves choose the task they are interested in.
3. No explicit system level design exists.
4. There is no project plan, schedule or list of deliverables.
5. The system is augmented in small increments.
6. Programs are tested frequently.

Others disagree and propose that the OSS development process is organized as a massive parallel development and debugging effort [e.g., 7]. However, the method for organizing this is not described.

Roles and responsibilities

As shown above, for some, the OSS development process seems to be quite free and wild. However, we maintain that this is deceptive, as the development process has some a defined structure, or else it would never have been able to achieve such remarkable results as it has in the past years. Also, mature and established companies have started to show interest in OSS. Among these are notably IBM, Apple, Oracle, Corel, Netscape, Intel and Ericsson [7]. Especially in larger OSS projects, the companies have taken the role of the coordinator and that of the mediator, thus acting as the project manager.

While no face-to-face meetings are typically used in the OSS context, the importance of the Internet as the means to communicate is a fundamental tool. The OSS development process needs a functional version management software that tracks changes in and allows the submission and prompt testing of new source code. The role of these tools should not be underestimated, as the developed software must be orchestrated and run continuously, and the developers themselves have varying skill levels and backgrounds. A typical OSS development effort structure, according to Gallivan, [13] is composed of several levels of volunteers:

1. Project leaders who have the overall responsibility for the project and who usually have written the initial code
2. Volunteer developers who create and submit code for the project. These can be further specified as:
   - Senior members or core developers with broader overall authority
   - Peripheral developers producing and submitting code changes
   - Occasional contributors
   - Maintainers and credited developers
3. Other individuals who in the course of using the software perform testing, identify bugs and deliver software problem reports
4. Posters participate frequently in newsgroups and discussions, but do no coding.

Sharma et al. [11] suggest that OSS development projects are usually divided by the main architects or designers into smaller and more easily manageable tasks, which are further handled by individuals or groups. Volunteer developers are divided into individuals or small groups. They freely select the tasks they wish to accomplish. Thus a rational modular division of the overall project is essential in enabling a successful outcome of the development process. Furthermore, these sub-tasks must be interesting so as to attract developers.

The agile methods advocate the same ideas of small and frequent releases as well as the coordinated development process. However, in a business environment the professionals typically do not have the same possibility of selecting their assignments.

Practices

To start or to acquire an ownership of an OSS project can be done in several ways: to found a new one, to have it handed over by the former owner, or to voluntarily take over an ongoing dying project [14]. Often the actual to-be users themselves define the product (i.e., the project) and do the coding. The process is continuous, as software development is evolving [seemingly] endlessly. Even though hundreds of volunteers may be participating in the OSS development projects, typically there is only a small group of developers performing the main part of the work [12, 15, 16].

Sharma et al. [11] describe some of the central organizational aspects in the OSS approach, e.g. the division of labor, co-ordination mechanism, distribution of decision-making, organizational boundaries, and informal structure of the project. Mockus et al. [12], on the other hand, exemplify the OSS development being a process where the systems are built by potentially large numbers of volunteers, work is not assigned as people undertake the work they choose to undertake, there is no explicit system-level design, or even detailed design and there is no project plan, schedule, or list of deliverables.

These above elements are possible pitfalls of OSS paradigm if not properly handled. For example, in order to work successfully, geographically dispersed individuals must have well functioning and open communication channels between each other, especially as the developers do not usually meet face-to-face. Having no system-level designs or project plans adds into the process even more uncertainties that must be mastered.

Lerner and Tirole [17] report in brief some experiences concerning the development paths of Linux, Perl, and Sendmail software, describing the development of these applications from an effort of a single programmer to a globally scattered team of tens of thousands of individuals. This means that the OSS development method is highly dependent on the Internet enabling global networked collaboration.

Scope of use

At present, the most widely published results mostly concentrate on development efforts concerning software development tools, server applications and Internet based products [17]. The OSS development paradigm itself does not bring forth any special application domains nor does it set any limits for the size of the software. However, the suggested development projects should comply with the elements that the OSS approach is founded on as described above. While new OSS development efforts are continuously producing raw material that can later be harnessed to commercial exploitation, it seems that the future trends also rely on taking an advantage of global developer collaboration [5]. From a legal perspective, the OSS development paradigm should be seen more as a licensing structure exploiting the terms of the General Public License or similar. Typical examples of OSS applications are, according to Feller [7], complex back-office infrastructural support systems with a high number of users.

The OSS development process itself can be implemented using different software development methods. However, these methods should comply with the characteristics of the OSS paradigm in order to produce a successful project.

4. Comparison of OSS and agile development

Table 1 shows the comparison of the agile, OSS and plan-driven (process-oriented) methods using Boehm's [18] analytical lenses. Examining the different "home-ground areas" in Table 1 shows how the OSS paradigm places itself between the agile and plan-driven methods, though having more in common with the agile methods. It was found that the most notable differences are in the proximity and size of the development teams, the customers' representation during the development project, and with the primary objective of the development work. The analysis shows that the OSS approach is close to a typical agile method, with the distinction that the OSS operates in a geographically distributed mode – a feature that is missing from the existing agile methods. In OSS development, the customer is often also a co-developer [1], which is not the case in agile software development. This thus places a challenge to the use of OSS principles in developing commercial software.

An agile software development method has been defined with the following characteristics [9]:

- *incremental* (small software releases, with rapid cycles),
- *cooperative* (customer and developers working constantly together with close communication),
- *straightforward* (the method itself is easy to learn and to modify, well documented), and
- *adaptive* (able to make last moment changes).

In the following, it is shown how the OSS software development paradigm complies with these characteristics. In the previous section the analysis already revealed some elements that reflect the agility of the OSS approach.

Incremental

As stated earlier, the OSS development process starts with early and frequent releases. Furthermore, systems developed under the OSS paradigm are typically augmented with small increments and the programs are tested frequently [12]. Linus Torvalds has expressed that his philosophy during the development process of the Linux operating system has been to "release early, release often" [3]. Koch [1] also describes the OSS as an extremely decentralized development method with low control of further development, exploiting fast release cycles. Hence, it appears that the OSS complies well with the definition of the agile method.

Cooperative

The network effect is crucial for the success of OSS development. New prototypes are introduced into the OSS community network. After the announcement, the crucial question and moment is, does the introduced prototype gather enough interest. The project will start to attract more and more developers, if it is challenging enough. Bergquist and Ljungberg [14, 319] have studied the development of relationships in OSS communities. They conclude that "one could understand the [OSS] culture as a kind of amalgamation of collectivism and individualism: giving to the community is what makes the individual a hero in the eyes of others. Heroes are important influences but also powerful leaders."

In many cases the roles of customers and co-developers intertwine, thus giving an interesting aspect to the development process itself. The OSS approach is pushing the cooperative nature to its limits as the network of developers can be both widespread and numerous. Traditional agile methods advocate geographically close networks with small developer teams.

Straightforward

The OSS is not well documented. There does not exist any identical convention among the developers using the OSS paradigm. Instead, there are many ways to conduct an OSS project. The OSS developers use industry perceived best practices. Therefore, it is hard to argue how easy the method is to learn, as there are no guidebooks to follow. However, this feature also makes it easy to modify since there are no strict rules but only good practices. Only the OSS licensing mechanism can be considered well documented, tested, and functioning.

The programming effort includes loosely-centralized, cooperative and free of charge contributions from individual developers. The OSS development process does not include any predefined formal documentation norms or customs. The customs and taboos are learned and perceived by experience. The agile development process also emphasizes the parallel development and debugging effort in its software development process [7].

Table 1. Home ground for agile and plan-driven methods [18], augmented with open source software column.

Home-ground area	Agile methods	Open source software	Plan-driven methods
Developers	Agile, knowledgeable, collocated, and collaborative	Geographically distributed, collaborative, knowledgeable and agile teams	Plan-oriented; adequate skills; access to external knowledge
Customers	Dedicated, knowledgeable, collocated, collaborative, representative, and empowered	Dedicated , knowledgeable, collaborative, and empowered	Access to knowledgeable, collaborative, representative, and empowered customers
Requirements	Largely emergent; rapid change	Largely emergent; rapid change, commonly owned, continually evolving – "never" finalized	Knowable early; largely stable
Architecture	Designed for current requirements	Open, designed for current requirements	Designed for current and foreseeable requirements
Refactoring	Inexpensive	Inexpensive	Expensive
Size	Smaller teams and products	Larger dispersed teams and smaller products	Larger teams and products
Primary objective	Rapid value	Challenging problem	High assurance

Seen from this perspective the OSS paradigm does not comply when compared with most of the existing agile methods. Some of the agile methods, however, are nothing but loosely coupled collections of best practices [9].

Adaptive

Peculiar to an OSS project is that the requirements are constantly being elaborated, and that the product is never in a finalized state, until all development work has ceased on the product [1, 7]. Depending on management, however, this can be seen both as a problem and as a strength.

"Informalismus", a term introduced by Schacci [19], describes the adaptivity in the OSS approach. In specific, resources can be browsed, crosslinked, and updated on demand. One of the strengths of agile methods is just the ability to allow last moment changes in a controlled way.

On the other hand, agile projects have clear starts and endings.

Summing up

From the above discussion it can be seen that in essence, the OSS paradigm complies with the agile methods. However, some focal differences remain that must be resolved before the OSS is viable also in the business community. The central question regarding Open Source Software is how to stabilize and manage the entire development process and how to commercialize the code to an application level that satisfies the business community. Based on the analysis presented above, it is suggested that the OSS community could benefit from the practical solutions put forward by the agile proponents and vice versa.

5. Conclusions

Even though it has been argued that the OSS development differs from the agile development mode in philosophical, economical, and team structural aspects we have shown that the OSS development method is rather close to the definition of an agile software development method. Geographically and culturally dispersed organizations could benefit from analyzing the pros and cons of the different OSS paradigms, and adapt the most prominent solutions into use in their specific context of software development.

Open Source Software is still fairly new in a business environment and a number of interesting research questions remain to be analyzed and answered. Among these are e.g. the effective and clear functionality of the legal licensing network and incorporating the visible benefits of the OSS paradigm into the context of a normal software developing organization [20]. Analysis showed that the OSS approach can be seen as one variant of the multifaceted agile methods. Software companies are showing more and more interest in finding out the possibilities of harnessing the OSS method to support their daily work as the method itself embodies several interesting aspects that could benefit the software industry.

6. References

[1] S. Koch, "Open Source Software-Entwicklung: Analyse und Aufwandsschätzung an einem Beispiel," in *Wirtschaftsinformatik*. Wien: Universität Wien, 2002, pp. 187.

[2] J. Highsmith, *Agile software development ecosystems*. Boston, MA.: Pearson Education, 2002.

[3] A. Gawer and M. Cusumano, *Platform Leadership. How Intel, Microsoft, and Cisco Drive Industry Innovation*. Boston: Harvard Business School Press, 2002.

[4] D. Voth, "Open Source in the US Government," *IEEE Software*, vol. January/February, pp. 73, 2003.

[5] T. O'Reilly, "Lessons from Open-source Software Development," *CACM*, vol. 42, pp. 33 - 37, 1999.

[6] R. Hightower and N. Lesiecki, *Java Tools for Extreme Programming*. New York: Wiley Computer Publishing, 2002.

[7] J. Feller and B. Fitzgerald, "A Framework Analysis of the Open Source Software Development Paradigm," presented at 21st Annual International Conference on Information Systems, Brisbane, Australia, 2000.

[8] www.opensource.org, 2003.

[9] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods. Review and analysis," VTT Technical Research Centre of Finland, Oulu VTT Publications 478, 2002.

[10] A. Cockburn, *Agile Software Development*. Boston: Pearson Education, Inc., 2002.

[11] S. Sharma, V. Sugumaran, and B. Rajagopalan, "A framework for creating hybrid-open source software communities," *Information Systems Journal*, vol. 12, pp. 7 - 25, 2002.

[12] A. Mockus, R. Fielding, and J. Herbsleb, "A Case Study of Open Source Software Development: The Apache Server," presented at 22nd International Conference on Software Engineering, ICSE, Limerick, Ireland, 2000.

[13] M. Gallivan, "Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies," *Information Systems Journal*, vol. 11, pp. 273 - 276, 2001.

[14] M. Bergquist and J. Ljungberg, "The power of gifts: organizing social relationships in open source communities," *Information Systems Journal*, vol. 11, pp. 305 - 320, 2001.

[15] B. Dempsey, D. Weiss, P. Jones, and J. Greenberg, "Who Is and Open Source Software Developer," *CACM*, vol. 45, pp. 67 - 72, 2002.

[16] R. Ghosh and V. Prakash, "The Orbiten Free Software Survey," vol. 2002: Orbiten.org, 2000.

[17] J. Lerner and J. Tirole, "The Simple Economics of Open Source," vol. 2002, 2001.

[18] B. Boehm, "Get Ready for Agile Methods, with Care," *Computer*, vol. January, pp. 64 - 69, 2002.

[19] W. Scacchi, "Is Open Source Software Development *Faster, Better*, and *Cheaper* than Software Engineering?," presented at 2nd ICSE Workshop on Open Source Software Engineering, Orlando, FL., 2002a.

[20] J. Dinkelacker, P. Garg, R. Miller, and D. Nelson, "Progressive Open Source," presented at ICSE'02, Orlando, FL., 2002.