Agility and Quality Attributes in Open Source Software Projects Release Practices

Antonio César Brandão Gomes da Silva
Glauco de Figueiredo Carneiro
Antonio Carlos Marcelino de Paula
Universidade Salvador (UNIFACS)
Bahia, Brazil
antoniocesar01@gmail.com, glauco.carneiro@unifacs.br,
acmarcelino@gmail.com

Miguel Pessoa Monteiro
Universidade Nova de Lisboa (UNL)
Lisboa, Portugal
mtpm@fct.unl.pt

Fernando Brito e Abreu
Instituto Universitário de Lisboa (ISCTE-IUL)
Lisboa, Portugal
fba@iscte-iul.pt

***Abstract*—*Context*: The need to accelerate software delivery, supporting faster time-to-market and frequent community developers/users feedback are issues that have lead to relevant changes in software development practices. Many Open Source Software (OSS) projects have engaged to achieve this through the adoption of agile practices in software release practices. *Problem*: There is no secondary study in the literature discussing evidences of the influence of agile approaches in OSS projects release practices. *Goal*: Identify published reports in the literature that characterize to which extent agility has influenced release approaches in OSS projects. *Method*: The characterization of studies followed a five-phase process to present a panoramic view of software releases practices in the context of OSS projects. *Results*: The overall data collected from 14 studies published from January 2006 to January 2016 depicted the following scenario: nine issues that characterize the advantages/influence of agility in OSS release approaches; four challenge issues in this approach; three possibilities of implementation and two main motivations towards the adoption of software release approaches through agility; and finally three main strategies to implement it. *Conclusion*: This study provides an up-to-date and structured understanding of the influence of agility on OSS projects release approaches based on findings systematically collected from a list of relevant references in the last decade.**

Keywords*—*Frequent Releases, Rapid Releases, Agile Methods, Open Source Software Projects.

I. Introduction

The effectiveness and success of practices adopted by Open Source Software (OSS) projects aroused interest of the Software Engineering research community [10][4][6][7]. Understanding how such software projects work enable companies to draw lessons from reported best practices and apply them in their internal projects [15][11]. A relevant practice is related to how OSS projects deal with releases. Many projects adopt a traditional release plan based on a set of features [14], while others adopt a time-based strategy in which releases are planned based on fixed and pre-determined time intervals [13]. Studies have reported that traditional release strategies have associated problems that can be overcome by time-based release management [9][10]. According to [1], the concept of rapid release (aka RR) approach was introduced by agile methodologies like Extreme Programming, which claims that shorter release cycles offer various benefits to both companies and end users.

In this paper, we review published studies that addressed the growing interest of OSS projects in the adoption of agile approaches in software release practices [3] [10]. This interest is motivated by the need to provide shorter time-to-market, in response to customer requests and to enhance costumer satisfaction [3]. This explains the success of OSS projects in the developers and users communities and shows the importance of both process and product quality attributes to achieve such success.

The rest of this paper is organized as follows. Section 2 presents the problem statement and scope of this paper. Section 3 outlines the research methodology. Section 4 reports the analysis of the studies. Section 5 presents threats to validity of this study. Concluding remarks, as well as a discussion of the limitations and scope for future research, are provided in Section 6.

II. Problem Statement and Scope

In this paper, we focus on practices adopted by OSS projects release projects. To the best of our knowledge, no previous characterization or secondary study have discussed the influence of agility on OSS projects release practices. For this reason, we conducted a characterization study to identify papers in the literature that report software release initiatives in different OSS projects and to which extent they are influenced by agile approaches. Those studies are a relevant evidence of best practices and challenges faced by successful OSS projects that can be used by the software engineering community to achieve quality attributes such as timeliness and efficiency.

While carrying out this study, we found one SLR and one systematic mapping focusing on Rapid Releases (RR), software tests [8] and continuous deployment in software intensive products and service [12]. However, none of these studies focused on the relationship between the issues targeted by the present paper. The literature review reported in [8] suggests that RR is a prevalent industrial practice, even in some highly critical domains, and that originates from successful approaches such as agile, open source and lean software development. The authors also conclude that empirical studies proving evidence of the claimed advantages and disadvantages of RR are still scarce [8]. In the case of how OSS projects implement release strategies influenced by agile methods, the

978-1-5090-3581-6/16 $31.00 © 2016 IEEE
DOI 10.1109/QUATIC.2016.56

scope and coverage of this paper significantly differ from previous published works.

III. Research Methodology

A. Planning the characterization

Identify the needs for a characterization study. To the best of our knowledge, there is no previous report addressing the influence of agile methods in OSS projects release practices based on primary studies published in the literature. The analysis of these evidences reported in the studies will help us identifying the relationship between the effective use of release approaches in OSS practices and software quality attributes manifested in these projects. These findings can be a useful reference to develop and enhance guidelines focusing on this theme.

Specifying the Research Question. We focused on the following research questions (Table I).

TABLE I. Research Questions (RQ)

RQ1	What are the time scale options applied to software releases in OSS projects?
RQ2	What are the main motivations for the adoption of agility in software releases practices in OSS projects?
RQ3	What are the main strategies adopted by practitioners to include agility in software releases practices in the OSS context?
RQ4	What are the main advantages and challenges related to the to inclusion of agility in software releases practices in the OSS context?

Adopted Strategy. The characterization considered as a start-point 30 papers selected from a Systematic Literature Review (SLR) focusing on OSS projects release practices recently conducted by the authors of this paper. Keywords were derived from the stated research questions and used to search the primary study sources. Regarding the time frame of the mentioned SLR, we considered papers published in journals and conferences from January 2006 to January 2016. The following search string was used in the SLR with the same strategy as described in [2]:

> *((("rapid release" or "fast release" or "frequent release" or "release history" or "quick release") and ("oss" or "open source" or "open source software")))*

Figure 1 conveys the phases of the SLR to reach the 30 papers.

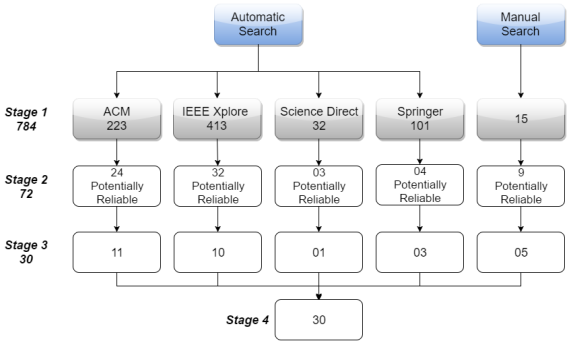

Fig. 1. Phases of the SLR Selection Process

Phases for the Studies Characterization. Figure 2 conveys the phases adopted by the authors to select the studies for the characterization and therefore to answer the research questions presented in Table I.

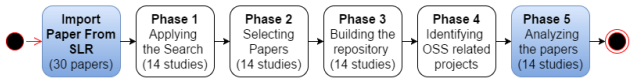

Fig. 2. Study Characterization Phases

Phase 1: Applying the Search String. The search string is presented as follows and was applied in all the 30 papers from the previous SLR. The result of this phase was a list of 14 papers containing the string "agil" as the root of the terms "agile" and "agility".

Phase 2: Selecting the Papers. The 14 papers from the previous phase were analyzed to identify evidences of the influence of agility in the software release practices. Papers that do not explicitly mentioned this influence were discarded. The result of this phase was a list of the same 14 papers.

Phase 3: Building the Repository. This phase built a repository comprised of studies considered relevant to characterize the influence of agility in release practices in OSS projects. The decision of which elements should be included in the repository was based on data obtained from the selected papers and their respective relevance in the context of software releases. These elements were organized in a mental model conveyed in Figure 3 to represent the studies. The nodes are numbered to identify the elements in the structure according to the RQ it is related to. Node 1 represents variations related to the time scale options applied to software releases in the OSS projects **RQ1**: (1.1) *Regular cycles*, (1.2) *Continuous Flow* and (1.3) *Short Release with not regular cycles*. Node 2 represents the motivations to adopt agility in software releases practices in the OSS context, in this case **RQ2**: (2.1) *OSS Attractiveness and Increase of Participants*, (2.2) *Maintenance and Improvement of Market Share*. Node 3 represents three possible strategies to include agility in software releases practices in the OSS context (**RQ3**): (3.1) *Test Driven Development*, (3.2) *Continuous Delivery* and (3.3) *Time-Based Releases*. The following advantages were identified in the literature (**RQ4 - part 1**): *Shorter Time to Market* (4.1.1), *Efficiency, Feedback, Customer Satisfaction* (4.1.2), *Test Effectiveness* (4.1.3) *Entry of New Team Collaborators* (4.1.4), *Pace of Innovation* (4.1.5), *Effective Planning and Monitoring* (4.1.6), *New Features* (4.1.7), *Bug Fixes* (4.1.8), and *Security Updates* (4.1.9). The challenges (**RQ4 - part 2**) reported were dealing with *Time Pressure* (4.2.1), *Technical Debt* (4.2.2), *Community Dependence* (4.2.3) and *Reliability* (4.2.4).

Phase 4: Identifying OSS Related Projects We present the *Table* II to convey software projects that were identified in this study regarding the implementation of software release practices influenced by agile approaches. In the table, each software project is referred as "P" followed by a number. The table also contains the name of the project, the study that reported its practices and the respective release time adopted.

IV. Analysis of the Studies

This section presents the results to answer research questions **RQ1, RQ2, RQ3 and RQ4** in conformance with Phase

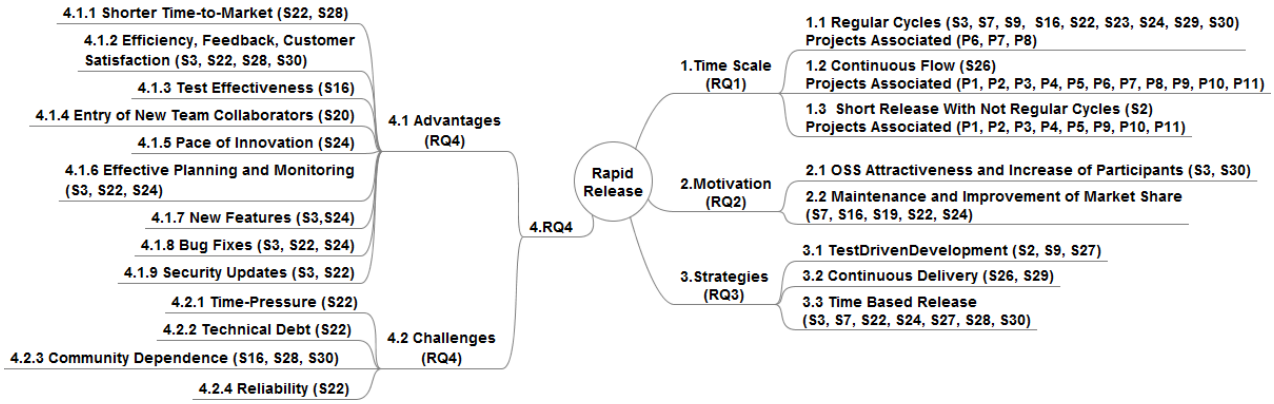

Fig. 3. Findings from the Selected Studies

TABLE II. OSS PROJECTS, STUDIES AND RELEASE TIME

ID	OSS-Project	Study	Release Time
P1	Apache Cocoon	S19	Not Informed in the paper
P2	Apache Tomcat	S19	Not Informed in the paper
P3	Debian Linux	S28, S30	15 - 18 months
P4	Gnome	S28, S30	6 months
P5	GNU Compiler	S28, S30	6 months
P6	Google Chrome	S24, S29	6 weeks
P7	Linux Kernel	S28, S30	2-3 months
P8	Mozilla Firefox	S3, S16, S22, S23, S24, S29	6 weeks
P9	Open Office	S28, S30	3 - 6 months
P10	Plone	S28, S30	6 months
P11	X.org	S28, S30	6 months

5 of Figure 2. All selected studies are listed in *Table III* and referenced as *"S"* followed by the number of the paper. A set of 14 out of 30 papers were selected from the original SLR list. For this reason, the IDs from the list of papers do not follow sequentially.

Figure 4 depicts the selected studies and the respective research questions they focus on. The Figure conveys that 11 studies addressed issues related to **RQ1**, 8 studies discussed **RQ2** issues, 10 studies were related to **RQ3** and, finally, 7 papers addressed **RQ4** issues.

Regarding **RQ1** (*What are the time scale options applied to software releases in the OSS projects context?*), we found evidences in the selected studies that allowed us to classify the time scale in OSS projects in three groups as follows. This is related to the item 1 and the subitens 1.1, 1.2 and 1.3 of Figure 3, where we can identify the timeliness quality attribute.

The first group is *regular cycle* time scale to provide new versions of the software project as reported by paper S3, S7, S9, S16, S22, S23, S24, S29 and 30. The second type is the implementation of the software release time scale in a *continuous flow* mode as reported by paper S26. The third type is related to *short release cycles with not regular cycle* reported in paper S2. The figure 5 represents the distribution of papers among the three types mentioned before. The identification of these software releases time scale options practiced by OSS projects are therefore references for potential new adopters that focus on agility. Moreover, considering that these three practices

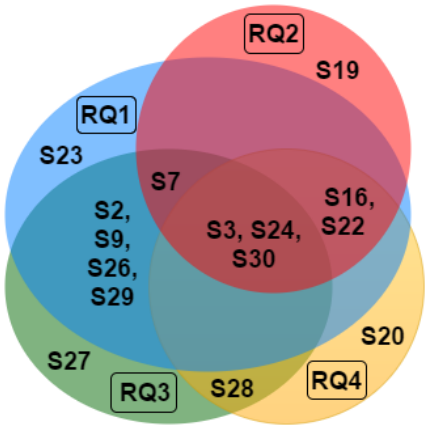

Fig. 4. Selected Studies per Research Question (RQ)

require process tailoring, they also provide evidences of quality attributes such as understandability, flexibility, maintainability and predictability.

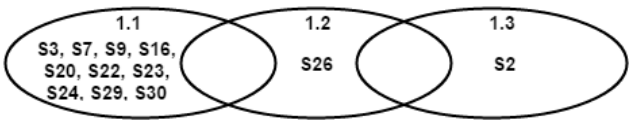

Fig. 5. Types of Time Scale (RQ1) according to Figure 3

In the case of *regular cycles* and according to *S30*, many projects moved towards a time-based release management strategy based on the early successful experiences of projects such as GNOME [1]. This increases the exposure of the software and may lead to better feedback. However, sufficient development done in release interval appear to be essential to pursue a time-based release management strategy. A large number of time-based FOSS projects have chosen a release interval of up to six months. Major Linux distributions, such as Fedora, are examples of this practice.

[1]https://www.gnome.org/

S16 highlights the influence of agility in the software release process when it reports the delivering of working software from every couple of weeks to every couple of months, with a preference to the shorter timescale. This is an evidence of Rapid Releases (RR) practices. Moreover, *S16* also reports that in the context of RR, the modifications of each release are limited, which makes the number of faults per release very small, compared to the traditional releases. *S7* investigates the consequences and impact of the RR approach on the security of the Mozilla Firefox browser. The resulting data shows that Firefox RR does not result in higher vulnerability occurrences. The pattern exhibited by vulnerability disclosure in Firefox is the result of would-be attackers having to re-learn and re-adapt their tools in response to a rapidly changing code base. In *S29*, it is mentioned that recent studies have started to evaluate the concept of continuous delivery and rapid releases from the perspective of software quality. The papers *S3, S22, S23 and S24* discuss the methodology adopted by Mozilla Firefox that moved from a traditional software release approach to a delivery of new releases every six months. On the other hand, *S9* reports that agile teams continually adapt to changing requirements late into the development lifecycle as they strive to deliver working software at frequent intervals.

Considering the possibility of *continuous flow*, *S26* mentions that rapid and continuous software engineering refers to the ”organizational capability to develop, release and learn from software in rapid parallel cycles, such as hours, days or very few weeks”, in the case of *short releases with not regular cycles*, *S2* presents an approach augmented with a number of engineering best practices specifically tailored to attain a weekly release cycle for a hosted software product. It is noted that the problem of managing rapid releases for a hosted product is likely to become even more important in the future, cutting across vast tracts of enterprise IT in addition to publicly hosted ”Software as a Service” products.

According to the Agile Manifest, *Regular Cycles* and *Short Cycles with not Regular Cycle* follow the principle of delivering new software releases within weeks or months. The software projects P6, P7 and P8 reported in papers (S3, S16, S22, S23, S24, S29 e S30) implement the *Regular Cycle* release timeline. On the other hand, the software projects (P1, P2, P3, P4, P5, P9, P10, P11) reported by the studies S19, S28 and S30 implement *Short Cycles with not Regular Cycle* release timeline. The *Continuous Flow* for software releases is one of the reference practices in the context of agile approaches aiming at prioritizing the clients demands and needs through the delivery of releases in a continuous fashion. The following software projects implement this type of release timeline: P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11 as reported in the studies S3, S16, S19, S22, S23, S24, S28, S29 e S30.

In the case of **RQ2** (*What are the main motivations for the adoption of agility in software releases practices in OSS projects?*), two main motivation factors were identified as favorable for the adoption of rapid releases: *OSS attractiveness and increase of participants* (S3, S30) and *maintenance and improvement of market-share* (S7, S16, S19, S22, S24). There are reports of projects that increased the number of participants where the majority are volunteers motivated by attractiveness and challenges associated with short releases activities (S30). Moreover, it raises the possibility of user feedback in a shorter feasible time interval. Maintenance and improvement of market share occurs as a result of the pace of changes and hence new features provided by the software. These motivations will help practitioners to a successful adoption of software release practices based on their their needs and expectations.

The *OSS attractiviness and increase of participants* are related to the agile principle due to the reported motivation of the software project participants as well as their trust and engagement in the project. The *maintenance and improvement of marketshare* provided evidences of prioritization of the customer satisfaction and the management of changes and evolution of the software project focused on competitive advantages of their customers. The Figure 6 represents the distribution of papers among the two types of motivation issues mentioned before.

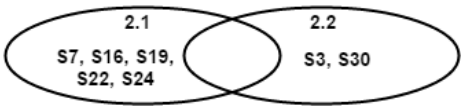

Fig. 6. Types of Motivation Issues (RQ2) according to Figure 3

S30 notes an increase in the number of participants, most of which originate from the open source community. The OSS model matured and participation in open source projects can add status to its participants as well. Another relevant factor is a means to maintain and increase market share (S7, S24). Fast changes in the software tend to increase its competitiveness. *S7* remarks that the survival of a software project entails the frequent introduction of new features. *S16, S22, S24* points out that the Firefox project motivated by declining market share adopted the rapid release model to be able to compete with Chrome, which already used it. The fast expansion of mobile platforms resulted in increased competition between applications, making rapid releases very important for the applications to remain competitive. *S19* points that frequent system releases are performed in line with user expectations

In the case of **RQ3** (*What are the main strategies adopted by practitioners to include agility in software releases practices in the OSS context?*), the time-based release strategy is mentioned as a relevant strategy by the following selected studies: S3, S7, S11, S22, S24, S27, S28, S30. The key point is that a successful implementation of time-based releases is based on trust among its contributors and the release manager, as well as on appropriate control structures that should be accepted by developers. Once the time-based release is chosen, it is time to determine the release interval. To this end, *S28* identified five factors that affect the choice of interval: (i) regularity and predictability; (ii) nature of the project and its users; (iii) commercial factors; (iv) cost and effort; and (v) network effects (need to synchronize the project release schedule with the schedules of other projects from which it can leverage benefits). The Figure 7 represents the distribution of papers among the four strategies to implement software releases as mentioned before. These strategies are a relevant starting point for building a body of knowledge regarding the the inclusion of agility in software releases practices in the OSS context.

According to *S28*, while a time-based release strategy provides several benefits, it is important to realize that it does not necessarily benefits all projects. The first step is to decide

whether or not a project is suitable for a time-based release strategy. It is not advisable to implement regular releases if there has been little work done that would warrant a new release. Other studies mention an effective quality process associated with a range of automated testing tools (S2 , S9 , S27) as an essential aspect in order for development to support frequent releases. Continuous delivery process is considered essential to adoption by companies that practise rapid releases (S26 , S29). Figure 7 represents the distribution of papers among the three types of motivation issues mentioned before.

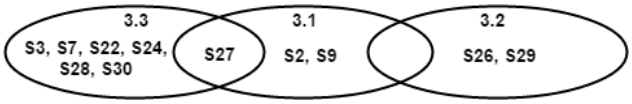

Fig. 7. Types of Strategies to Implement Software Releases (RQ3) according to Figure 3

S3, S7, S22, S24 describes the Mozilla strategy, which is characterized by each release of Firefox to completely replace the previous one. In addition, every new version goes through the following 4-release-channels work flow:

S30 explain that in a time based strategy, a specific date is set for the release well in advance and a schedule created so people can plan accordingly *S2* notes that it is fundamental for an effective QA process to use some automated test tool in an integrated manner, so to support frequent releases. *S9* stresses that focus on testing is critical for small teams that support quick releases. *S26* reports a systematic mapping based on continuous delivery and notes that continuous evolution of features drive this process, enabling for example the monthly delivery of releases. *S29* also lends strength to the idea that continuous deliveries are a recent phenomenon that is suitable for the rapid release of modern applications.

The analysis of **RQ4**. *What are the main advantages and challenges related to the to inclusion of agility in software releases practices in the OSS context?* had as a result the identification of the following quality attributes associated with the advantages presented in the branch 4.1 of Figure 3: 4.1.1 Shorter Time-to-Market (*efficiency* and *effectiveness*), 4.1.2 Efficiency, Feedback, Customer Satisfaction (*credibility*, *failure transparency*, *manageability*), 4.1.3 Test Effectiveness (*testability*, *traceability*), 4.1.4 Entry of New Team Collaborators (*learnability*), 4.1.5 Pace of Innovation (*flexibility*), 4.1.6 Effective Planning and Monitoring (*manageability*), 4.1.7 New Features (*evolvability*, *extensibility*), 4.1.8 Bug Fixes (*maintainability*, *repeatability*, *testability*, *traceability*) and 4.1.9 Security Updates (*securability*)

Advantages. The main positive points associated to rapid releases are: quick return on customer needs, rapid delivery of new features, quick bug fixes, immediate release security patches (S3, S22, S24), increased efficiency (S22) and greater focus on quality on the part of developers and testers (S22, S28, S30). The knowledge of these advantages and challenges beforehand is useful for the implementation of effective strategies to deal with reported problems raised by the implementation of agility in the software release approaches in the context of OSS projects.

S22 presents a semi-systematic review that associates rapid releases to testing. It reports that frequent releases driven by testing allow for a greater and quicker feedback in each release, and also increases the incentive for developers to deliver a quality product. Noteworthy is the increased efficiency due to greater time-pressure. The study also adds to the list of benefits that narrower tests - due to the reduced test time - also allows for deeper testing. Narrower tests are also easier to manage. *S3, S22 and S24* highlight the ease of planning and testing, since the tests are more focused and run more frequently, easing the monitoring of progress and quality.

S24 proposes the increase in the pace of innovation, since the high rate of releases encourages the team to continuously attempt new solutions and new tools. In addition, a greater rate of releases provides more marketing opportunities for the company. *S28* notes that short release intervals allow for more competitive OSS projects compared to proprietary designs, since rapid release brings significant advantages over competitors using the traditional cycle. For example, the *Beta* cycle corresponds to several new product versions arriving to the market. Another factor raised is the enhancement of reputation and employee satisfaction, as they see their code quickly being used by users.

S30 notes a trend regarding the increasing maturity of the practice of OSS, with an increase in their significance and economic potential. OSS projects are increasingly adopting rapid releases so as to allow for greater quality and sustainability. The quicker feedback that these practices allow also provides more information on what parts of the software are more in need of attention.

Challenges and Issues: *less reliability in new versions* (S22), *increase in the "technical debt"* (S22), *pressure felt by employees* (S22) and *community dependence* (S16, S28, S30).

In its systematic review, *S22* presents the main weak points of rapid releases. On one hand, tests become more focused but on the other, it also becomes practically impossible to test all possible options. In addition, the short time available does not allow for test quality requirements, e.g., performance. Another point mentioned is increased pressure on the team, which can lead to exhaustion. Finally the increase in technical debit are observed, since it allows for less time for activities such as refactoring. Neglect of those issues risks compromising the quality of the software and negatively impact organizations in the long run. In his study, *S16* claims that the difficulty in attracting a large number of volunteers for the test community makes the tests in rapid releases more deadline oriented. *S28* claims that projects maintained exclusively by volunteers require a significant planning effort to cope with periods of shortage of volunteers, e.g., Christmas in December.

V. Threats to Validity

The following types of validity issues were considered when interpreting the results from this review.

There may be bias in data extraction. This was addressed by defining a data extraction form to ensure consistent extraction of relevant data to answering the research questions. The findings and implications are based on the extracted data. One possible threat is the selection bias. We addressed this threat during the selection step as described in Figure 2, i.e. the studies included in the characterization were identified through a thorough selection process which comprises

of multiple phases. The studies identified from a preview systematic review conducted by the authors were accumulated from multiple literature databases covering relevant journals and proceedings. One possible threat is bias in the selection of publications. This was addressed through specifying a research protocol that defines the research questions and objectives of the study, inclusion and exclusion criteria, search strings that we intend to use, the search strategy and strategy for data extraction. The set of 14 papers selected in Phase 5 as described in Figure 2 is a potential validity threat. In this case, there is a threat that the results so far obtained could not be generalized. However, the studies were selected having as a start point the ones published between 2006 and 2016 from a previous systematic review conducted by the authors. For this reason, the set of the 14 studies are considered representative enough as a sampling for this characterization.

VI. Conclusions and Future Work

This paper presents an initiative to identify published reports in the literature to characterize to which extent agility has influenced release approaches in OSS projects. Far from being anecdotal, the evidences collected and discussed in this work have the goal to gain and share insight from the literature. The expected result is that the new OSS projects decision makers could be more confidence and hence adopt reported practices. The first principle of the Agile Manifesto states that top priority is to satisfy the customer through early and continuous delivery of valuable software [5]. This principle is achieved through the implementation of software process initiatives that are evidences of quality attributes of *efficiency* and *effectiveness*. Agile approaches have as a basis the reduction of time associated to deliver new versions to users that can be achieved, for example, through earlier feedback. These approaches can be in turn associated with process quality attributes such as *credibility*, *failure transparency* and *manageability*. The primary contribution of this paper is to reveal the motivations behind the adoption of agility in OSS project releases, strategies applied and identification of the potential advantages and difficulties faced in this regard. As on going work, we are collecting data from real OSS project repositories so that we can analyze in fact to which extent reported practices have been adopted in new tendencies of selected projects in release practices.

References

[1] C. Andres and K. Beck. Extreme programming explained: Embrace change. *Reading: Addison-Wesley Professional*, 2004.

[2] L. Chen and M. A. Babar. A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, 53(4):344–362, 2011.

[3] D. G. Feitelson, E. Frachtenberg, and K. L. Beck. Development and deployment at facebook. *IEEE Internet Computing*, 17(4):8–17, 2013.

[4] B. Fitzgerald. The transformation of open source software. *Mis Quarterly*, pages 587–598, 2006.

[5] M. Fowler and J. Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

[6] J. M. Gonzalez-Barahona, D. Izquierdo-Cortazar, S. Maffulli, and G. Robles. Understanding how companies interact with free software communities. *IEEE software*, 30(5):38–45, 2013.

[7] J. M. Gonzalez-Barahona and G. Robles. Trends in free, libre, open source software communities: From volunteers to companies. *IT–Information Technology*, 55(5):173–180, 2013.

[8] M. V. Mantyla, F. Khomh, B. Adams, E. Engstrom, and K. Petersen. On rapid releases and software testing. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 20–29. IEEE, 2013.

[9] M. Michlmayr and B. Fitzgerald. Time-based release management in free and open source (foss) projects. *International Journal of Open Source Software and Processes (IJOSSP)*, 4(1):1–19, 2012.

[10] M. Michlmayr, B. Fitzgerald, and K.-J. Stol. Why and how should open source projects adopt time-based releases? *Software, IEEE*, 32(2):55–63, 2015.

[11] P. C. Rigby, B. Cleary, F. Painchaud, M.-A. Storey, and D. M. German. Contemporary peer review in action: Lessons from open source development. *Software, IEEE*, 29(6):56–61, 2012.

[12] P. Rodríguez, A. Haghighatkhah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo. Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, 2016.

[13] G. Ruhe. *Product release planning: methods, tools and applications*. CRC Press, 2010.

[14] M. O. Saliu and G. Ruhe. Bi-objective release planning for evolving software systems. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 105–114. ACM, 2007.

[15] K.-J. Stol and B. Fitzgerald. Inner source–adopting open source development practices in organizations: A tutorial. *IEEE Software*, 32(4):60–67, 2015.

TABLE III. Selected Studies

ID	Author, Title	Venue	Year
S2	Puneet Agarwal, Continuous SCRUM: agile management of SAAS products.	ISEC	2011
S3	Khomh,F., Dhaliwal T., Ying Zou and Adams, B., Do,Faster Release Improve software Quality? An Empirical Case Study of Mozilla,Firefox.	MSR	2012
S7	Sandy Clark, Michael Collis, Matt Blaze and Jonathan M. Smith, Moving Targets: Security and Rapid-Release in Firefox.	CCS	2014
S9	Anil Shankar, Honray Lin, Hans-Frederick Brown and Colson Rice, Rapid Usability Assessment of an Enterprise Application in an Agile Environment with CogTool.	CHI	2015
S16	Hemmati,H. , Zhihan Fang and Mantyla, M.V. Prioritizing Manual Test Cases in Traditional and Rapid Release Environments.	ICST	2015
S19	Bifflˌˌ,S. , Sunindyo, W.D. and Moser, T. Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release, Software Projects.	CISIS	2010
S22	Mntyl M, Adams B, Khomh F, Engstrom E and Petersen K On rapid releases and software testing: a case study and a semi-systematic, literature review.	ESSE	2015
S23	Souza,R, Chavez C and Bittencourt R. Patch rejection in Firefox: negative reviews, backouts, and issue reopening.	SBES	2015
S24	Adams B, Khomh F, Dhaliwal T and Zou Ying, Understanding the impact of rapid releases on software quality.	ESSE	2015
S26	P Rodrguez, A,Haghighatkhah LE and Lwakatare, Continuous deployment of software intensive products and services: A systematic mapping study.	JSS	2016
S27	MV,Mntyl, K Petersen and, TOA Lehtinen, Time Pressure: A Controlled Experiment of Test Case Development and Requirements Review.	ICSE	2014
S28	M,Michlmayr, B Fitzgerald and KJ Stol, Why and How Should Open Source Projects Adopt Time-Based Releases?.	IEEE	2015
S29	B Adams, M Michlmayr Modern Release Management in a Nutshell: Why Researchers should Care.	SANER	2016
S30	B Fitzgerald, S McIntosh Time-Based Release Management in Free/Open Source (FOSS) Projects.	LERO	2011