Influencers of Quality Assurance in an Open Source Community*

Adam Alami
IT University of Copenhagen

Yvonne Dittrich
IT University of Copenhagen

Andrzej Wąsowski
IT University of Copenhagen

ABSTRACT

ROS (Robot Operating System) is an open source community in robotics that is developing standard robotics operating system facilities such as hardware abstraction, low-level device control, communication middleware, and a wide range of software components for robotics functionality. This paper studies the quality assurance practices of the ROS community. We use qualitative methods to understand how ideology, priorities of the community, culture, sustainability, complexity, and adaptability of the community affect the implementation of quality assurance practices. Our analysis suggests that software engineering practices require social and cultural alignment and adaptation to the community particularities to achieve seamless implementation in open source environments. This alignment should be incorporated into the design and implementation of quality assurance practices in open source communities.

KEYWORDS

Open Source Software, Quality Assurance, OSS Community.

ACM Reference format:

A. Alami, Y. Dittrich, and A. Wasowski. 2018. Influencers of Quality Assurance Practices in an Open Source Community. In *Proceedings of 11th International Workshop on Cooperative and Human Aspects of Software Engineering, Gothenburg, Sweden, May 2018 (CHASE 2018).* https://doi.org/10.1145/3195836.3195853

1 INTRODUCTION

Open Source Software (OSS) communities have become a serious contender for commercial software supply. The open source paradigm is gaining momentum in strength and is increasingly adopted by the traditional software industry [1-2]. This industrial interest brings its own requirement for OSS, especially regarding quality. Traditional organizations use a combination of practices, processes, and techniques to produce quality software. Yet what can be done and achieved in a traditional setting might not be reproducible in an open source community. Hence, understanding the challenges of quality implementation in OSS communities is timely.

Little is known of how OSS communities perceive quality and the challenges of implementing quality practices in OSS communities. Community-based organizations are culturally different and have been established based on fundamentally different sets of values and goals. Software engineering practices and techniques seem to be designed generically to fit most circumstances and organizational settings. However, OSS communities have shown us that they can build highly professional products using social–technical processes [3, 4]. Although some OSS practices are inspired from the software engineering knowledge and practices, in most instances, the adoption deviates from the prescribed conduct. Scacchi [3] observes some "informalisms" in the adoption process that reflect the peculiarities of the involved community. He believes "informalisms" captures the uniqueness of how the community works and produces software. We want to learn from these particularities to assist in the implementation and adaptation of quality assurance practices.

The central objective of this study is to investigate how the community's social and cultural traits influence the implementation of quality assurance (QA) practices, techniques, and tools. We investigate the following questions:

RQ1: What are the forces influencing the implementation of quality assurance practices in the ROS community?

RQ2: How do social and cultural variables influence the implementation and execution of practices?

We define the term "practice" in line with as "… a common way of acting, acknowledged by a community as the correct way to do things. It can be taught to newcomers by letting them take part in this practice as an apprentice [5]. A community maintains the common practice through more or less formal 'articulation work' [6] which is also the means to handle exceptional situations. Ad-hoc behavior—always necessary to handle exceptions and to maintain the 'normal' [7]—is as such only perceivable by its deviation from both the formalized rules and the established practice." [8].

The ROS Community is large and diverse. Its Wiki platform receives over 1.4 million unique visitors a year and has 6,749 registered users. The community "discourse" receives an average of 150 posts a week. The total downloads of the .deb packages is over 13.4 million. Over ten years, ROS has become one of robotics' *de facto* standard operating systems. The ROS community has different attributes than the commonly studied ones (i.e., Linux and Mozilla). First, it produces software components for robotics. Second, it is a multidisciplinary community. Third, most ROS developers are not software engineers. Their educational background is diverse but mainly from mechanics and electronics

* Camera ready version, accepted at *CHASE'18*, May 27, 2018, Gothenburg, Sweden

disciplines. In addition, members have varying professional experiences (i.e. packages developers, students, CTOs, etc.)

We find that the community has a strong quality awareness. Some industry-wide accepted practices have been implemented: (1) well-defined development process, (2) defects management process and tool, (3) code review, (4) continuous integration, (5) unit testing, and (6) knowledge sharing. However, these quality practices are experiencing implementation and execution challenges. Six forces and constraints—participation motives, priorities of the community, meritocratic culture, sustainability, complexity and adaptability of the community—have greatly influenced the implementation of QA in the ROS community. The cultural traits of the community also sway the QA implementation. Furthermore, the quality practices in place are constrained by sustainability issues and the complexity of the process of developing robotic systems.

This paper reports on a qualitative research on software QA practices following a mixed research method. Mixed methods deepen understanding of the research problem. Three techniques have been used: interviews with ten participants, virtual ethnography, and community reach-outs. In the last decade, the focus of OSS studies has been on "high profile" communities, like Mozilla, the Linux Kernel, and Apache. We selected ROS for its uniqueness as robotics software with a large and diverse participant base. This will contribute to the diversity of the samples studied previously.

2 RELATED WORK

We identified two streams of related work: (1) what the QA practices in OSS communities are and (2) how quality is assured in OSS development.

QA Practices in OSS Communities. ISO defines QA as "focused on providing confidence that quality requirements will be fulfilled" [9]. It is a set of activities for ensuring quality in software engineering processes that ultimately should result in quality software products. Halloran and Scherlis [10] surveyed eleven OSS projects to identify the QA practices adopted by the communities. They observed a variation in the adoption and implementation of QA. While some communities (i.e., Mozilla and NetBeans) have dedicated QA teams and well-established QA practices, other communities seem to follow the practices only rudimentarily. Unfortunately, no clear pattern emerges on when a practice succeeds and when it fails. In order to understand why some communities succeeded in implementing some QA practices successfully while others did not, we study the influencers of a successful QA implementation and execution.

Michlmayr and coauthors [11] studied quality practices in seven OSS communities. They found that the degree of adoption of quality practices influences the overall quality of the community product. They identify a set of quality practices taking place in the studied communities: new members joining, release management, branch management, peer review, testing, defects management, and standards and guidelines documentation. Although these practices are praised industry-wide, some are not formalized, and the implementation faces challenges. They list six quality practices issues: unmaintained code, managing variability, latency in security fixes and updates, ambiguous bugs reporting process, difficulty attracting new participants, and task coordination problems. However, they do not analyze the root causes of these issues.

Zhao and Elbaum [12] report that software development tools are popular in OSS communities; 75% of the respondents of their survey use configuration management tools, and 61% of the projects employ bug tracking tools. At the same time, documentation is not popular; only 32% of surveyed projects have design documents, and only 20% have documents to plan releases. More than half (58%) of the projects spent more than 20% of their time on testing, but only 15% of the projects spent more than 40% of their time on testing. It seems that larger projects tend to spend less time on their testing phase compared to smaller projects. They find that 20–40% of bugs are identified by end users. This agrees with a qualitative survey of Halloran and Scherlis [10]. The implementation of testing practices varies across communities. Also, non-programing activities (i.e., documentation) are not favored by contributors. Yet what makes some communities keenly embrace testing practices while others seem to shy from it has not been investigated. There is a need for studies to investigate further why non-programing tasks are not stimulating for OSS developers.

Rigby et al. [13] argue that despite being difficult to implement, code peer review is largely adopted by OSS communities as a central quality control practice. They studied the efficiency and the effectiveness of the practice in OSS communities. They found that the efficiency and the effectiveness of the practice depend on the level of the participation in the review process, the size of the change, and the author's experience and expertise. While the author expertise shortens the review cycle, the size and the complexity of the change elongates the cycle.

Lussier [14] recounts the experience of his team joining the Wine project, an open source implementation of the Windows API. The team is collocated and affiliated with a software strategy and research consulting services firm. Their first contribution did not pass the code review process since it did not meet the community standards. He recalls, "Team members watch the code carefully. No one wants to see bugs introduced into the source tree … a real sense of ownership, of pride in the work, exists on Wine." Initially, the team resented the rejection. However, after three consecutive rejections, they adapted their programing standards to the community standards and conventions. This narrative is consistent with the widely accepted assumption that OSS quality is owed to peer review. Elsewhere, code is also claimed to be of high quality because it is created with passion, and developers are highly motivated because they enjoy what they do [15].

How is quality assured in OSS development? In the closed source software development, QA relies on procedural rigor, extensive testing, and high testing coverage. Quality in OSS is assured by development (code modularity and frequent releases), not by control. Assuring quality is highly dependent on high

participation in the project. High participation and frequent releases create energy in the development process. This facilitates bug discovery and generates a fast feedback cycle. Consequently, defects are identified and corrected more quickly [15].

Most successful OSS projects are sustainable and apply structured and organized development processes. Otte and coauthors [15] suggest that attracting talented contributors with diverse skills, implementation of a QA knowledge sharing infrastructure, standards and guidelines, and tools help to ensure sustainable quality practices. However, sustainability remains the key parameter for achieving quality in OSS communities [16, 17].

Open source software development necessitates specific methods and techniques for assuring quality [18]. Wahyudin et al. [19] claim that quality in the OSS development process is achieved via sustainability, peer review, and code modularity. They argue that code modularity enhances features evolution and minimizes bugs' introduction in the evolution process. Aberdour [17] believes that irrespective of the community dedication to quality, having a sustainability strategy is detrimental to assuring quality. Khanjani and Sulaiman [18] believe the discovery of bugs is dependent on the size of the community. The larger the community is, the more chances there are of bugs being identified, reported, and fixed, also known as Linus's law: "enough eyeballs, all bugs are shallow" [18]. In addition, having a knowledge sharing and collaboration platform facilitates knowledge dissemination and subsequently nurtures quality contributions and effective communication [19-20].

Abdou et al. [21] investigated some high-profile OSS communities' (Apache, Mozilla, and NetBeans) software testing practices and how they conform or deviate from ISO/IEC standards. The studied communities have matured testing practices. Still, the implementation of testing practices deviates from the prescribed industry version. Their study is limited to four large, successful, and well-established communities (i.e., Mozilla, Apache, NetBeans, and IDE).

Most of the identified work explored QA in isolation of other research streams, such as participation motives, culture, community sustainability, and community development. Practices execution cannot be completely detached from their milieu (i.e., organization, community) and the social context. They have to be studied in relation to the social, cultural, and organizational context of their milieu.

There has been a recent shift of interest toward QA in the context of open source software development [14-15]. It seems to indicate that quality is assured via the combination of one or more of these variables: code reviews, dedicated QA team, Linus's law, and a sustainability strategy. Simultaneously, other studies [6-8] appear to agree that software engineering QA practices make their way to OSS communities. There are significant empirical evidences that OSS communities adopt QA practices from software engineering. However, it seems that in some instances, the implementation of these practices is experiencing challenges [11, 12]. We cannot understand the success of software engineering practices in OSS communities without understanding how the social fabric and cultural variables correlate with these practices. The dependencies and the correlation between OSS community social and cultural variables and QA is not well-understood. Hence, this study suggests investigating what influences the implementation and the execution of QA practices in OSS communities. This is an initial step in a three-year research project to propose an implementation strategy for software engineering practices in OSS communities.

3 RESEARCH METHOD

This paper presents results of qualitative research in one case study community (ROS). The data was gathered using a combination of techniques: in-depth semi-structured interviews with ten participants, virtual ethnography, and community reach-outs. Qualitative research methods entail a structured process for the collection, organization, and interpretation of textual material derived from conversations, interviews, or observation [22-24]. One researcher spent 120 hours studying the community online infrastructure, forums, and virtual interactions. The researchers attended four community events. These community reach-outs were an opportunity to observe, be part of conversations, and experience the community atmosphere. Field notes were used to capture this exposure to the community. This ethnographic experience was complemented by in-depth interviews with ten active community members. Their professional roles and participation in the community varied from core developers to passive as users of the community code.

The data analysis was achieved by open coding, focus coding, and theoretical coding [24]. This study did not use grounded theory as an underlying research methodology, but its grounded approach has been the guiding process for the empirical data analysis.

Subject. ROS and ROS Industrial are the community subjects of this study. The Robot Operating System (ROS) is a middleware framework that is widely used in robotics. ROS provides standard operating system facilities such as hardware abstraction, low-level device control, and commonly used robotics functionality.

The underlying philosophy of ROS is to make universal software portable to different robotics systems. ROS is based on the concept of reuse and open source software. Its origins can be traced back to 2007. The project was incepted by the Stanford Artificial Intelligence Laboratory [25]. In 2008 a startup, Willow Garage, inherited the project. Five years later, in 2013, Willow Garage was absorbed by another company, and ROS "stewardship" transitioned to the Open Source Robotics Foundation [25]. Today, ROS is the de facto operating system for robotics.

ROS Industrial is "an open-source project that extends the advanced capabilities of ROS software to manufacturing" [26]. ROS Industrial is a branch of ROS with a specific industrial application focus. Incepted in 2012, ROS Industrial has secured the collaboration of key players in the robotics industry (e.g., ABB, Yaskawa, Siemens, John Deere, BMW, Bosch, etc.). ROS Industrial's ambition is to become the worldwide open source standard for industrial robots.

4 FINDINGS

In response to RQ1 and RQ2, we observe that QA practices in the ROS community are influenced and constrained by the following forces:

1. Participation motives
2. Priorities of the community
3. Meritocratic culture
4. Sustainability
5. Complexity
6. Adaptability

The ROS community retains some of the software engineering and industry practices and processes. Many of the community QA activities still evolve; many experience challenges in the implementation and execution. Some of these issues are of a mechanical nature (i.e., outdated documentations) and require straightforward mechanical adjustment. However, a significant segment of the issues involves cultural alignment and/or alignment with the particularities of open source software development practices and processes. These issues manifest the cultural and social nature of the community. Addressing them will necessitate aligning the practice or the process with the cultural setting of the community.

These issues are merely a manifestation of unfit practices and adaptation failure to the environment ROS community. If there is a problem, then there might be reasons for it to exist in the first place. The issues exist because of a reaction to the introduction of change. The implementation of these practices did not cater to the social and cultural particularities of the community. It assumed that a default implementation will fit the community.

In the following, we are discussing the identified influencers by presenting the grounding for them in our data and in the existing literature. In the final paragraph of each subsection discussing an influencing force, we allow ourselves to speculate in what way it could be used to improve QA practices in this community.

4.1 Participation Motives

The participants do not consciously demonstrate the motives and their impact on their engagement in the community. Our analysis demonstrates that both intrinsic and extrinsic motives have influenced the implementation of QA practices in the community, mainly ideology and enjoyment. Some participants have strong ideological grounds, and it is manifested in their conduct and engagement in the community. Programming and the challenge of complexity are sources of enjoyment for some participants.

Intrinsic motivation refers to behavior that is driven by internal rewards [27] based on internal satisfaction and self-enjoyment [28]. The motivation to engage in an intrinsic behavior arises from within the individual because it is naturally satisfying. This contrasts with extrinsic motivation, which involves engaging in a behavior in order to earn external rewards [27, 28] that arise outside of the individual. It can involve tangible or psychological rewards. Psychological forms of extrinsic motivation can include praise and public acclaim [27, 28].

4.1.1 Ideology

INFLUENCER 1: OSS ideology is present in the ROS community thinking and decision making.

Data. Openness is the ideology attriute that influences practices in the ROS community. Some community members value this norm highly. In one of the community events (ROSCon 2017) that we attended, members discussed Slack as an online communication tool adopted by a group of developers for discussions and collaboration. Several community members refused to use it, while others were happy to continue using it. One member got emotional when the item came up for discussion and asserted, "I refuse to use it. It is not open source!" Another community member joined the opposition: "It is disappointing to see some people using a closed source, but I refuse to use it." There was an awkward silence before the discussion advanced to another subject. Apparently, not all community members rank openness equally high. Some have a relaxed and pragmatic attitude toward adopting closed source infrastructure and tooling.

Analysis. Openness is a mandated community requirement rooted in some members' ideological standpoints. Although adherence to the ideological beliefs are not shared with the same enthusiasm across the community, it is a fundamental variable and should not be dismissed. Similar to the Slack case, the implementation of a tool, process, or practice that does not embrace openness and transparency would create division in the community, and the chances of the practice being abandoned by a segment of the community members is high.

Openness is a manifestation of two cultural traits of open source communities: transparency and truth [4]. Pavlicek [29] believes that truth is a fundamental community asset. He explains that truth and transparency empower the community to produce "free software." Elliot and Scacchi [4] explain that "speaking the truth" is evident in the community social life and work practices.

"Ideologies are the shared framework of mental models that groups of individuals possess that provide both an interpretation of the environment and a prescription as to how that environment should be structured" [30]. The OSS ideology origins are deeply established in the "Free Software Movement" of the '80s led by Richard Stallman [4, 31, 32]. The movement is widely accredited for paving the way for the open source development.

There has been considerable interest in understanding the ideological framework of open source software communities [4, 15, 33, 35]. In settings such as OSS communities, where entry barriers are nonexistent and institutionalization of control is a challenge, ideology seems to facilitate order [34]. However, Ljungberg [34] suggests that commitment to the ideology varies widely across developers.

Impact. Stewart and Gosain [35] found that open source participants adhere to this ideology. David et al. [36], David and Shapiro [37], Ghosh [38], and Ghosh et al.'s [39] surveys suggest that members' participation motives have underpinning ideological beliefs. OSS ideology needs to be analyzed, and its underlying beliefs and norms must be acknowledged before the design and the implementation of software engineering practices in OSS communities. We argue that ideology should be assimilated into the community practices to ensure its success. For

example, when selecting a tool for a particular process, one should consider an open source to accommodate the openness feature of the community ideological values. A closed source tool will create division, and eventually the associated tool and practice will be abandoned.

4.1.2 Enjoyment

INFLUENCER 2: Enjoyment is equated with challenge in the community culture, while the QA tasks are not viewed as challenging.

Data. We observed that enjoyment is a key driver in community participation. Non-programing tasks are either being duly executed or taking place loosely in the community practices. An attendee at the community yearly conference commented on a poster that advocated an effort to implement and promote QA practices in the community: "What you are trying to achieve is formal; we developers seek fun in writing code and creating new features." "Skipping and skimming through pull requests and during code review are common occurrences," one participant stated. This attitude has its consequences. He further explained, "Things don't get reviewed and don't get the necessary attention for a longer time."

Analysis. Enjoyment as an intrinsic motivation has been associated with programming ("coding"). However, programming is not the only activity taking place in an open source software development environment. Other inherent tasks include code reviews, release management, documentation writing and maintenance, etc. There is a need to understand the relationship between non-programming tasks and enjoyment. In a community based largely on volunteers, this is a condition *sine qua non* for making software development processes function beyond programing.

Deci and Ryan's [40] self-determination theory is a widely supported contemporary intrinsic motivation theory. It suggests that humans have three intuitive psychological needs: a need to feel competent, a need to belong, and a need to feel independent. Intrinsic motivations emerge in people's behavior to support these psychological needs. Deci and Ryan [40] explain that when people feel competent, autonomous, and self-determined, they will seek to fulfill their internal self-satisfaction. Freedom of choice, the presence of a challenge, and the ability to overcome the challenge are the three variables that, when met, stimulate intrinsic motivation [40]. Non-coding tasks are not "challenging" or at least are perceived as not being so. They attract fewer contributors, and consequently, QA practices in the community (i.e., maintenance and testing) receive less attention.

Impact. It has been suggested that enjoyment is a key construct for understanding and explaining the motivation of OSS participants [27, 41, 42]. Lakhani and Wolf [27] suggest that enjoyment is a prevalent motivation amongst OSS contributors. Hence, it's not a force to ignore. Then what impact does this have on the implementation of QA and other non-programming tasks in OSS communities? It appears that one needs to immerse fun into non-coding tasks. This can be done by reframing the tasks or the process. For example, developing automated tests (that are programmed and leave credit in code repositories) may be perceived as more fun than writing manual tests. That would make it more challenging and more akin to programming.

4.2 Priorities Of The Community

INFLUENCER 3: QA is not high in the priorities of the community. Consequently, QA tasks are neglected.

Data. Priorities of the community are determined by the order of importance between various contributions and initiatives. Priorities are subject to change with changes in the community or with changes in people's objectives, motives, or knowledge. Innovation and functional depth and breadth are the priorities of the ROS community. "Everybody is aiming for new things," one participant stated. They thrive on innovating and resolving challenging and complex technical issues. This has been observed at the community yearly conference (ROSCon 2017). The main program was dedicated to new innovative features and use cases running on the community technological platform. One interviewee stated, "More importantly, our focus is features and functionality. The process is not always the priority." Another participant confirmed, "We want to also focus on the new stuff." In addition, new features are commonly announced and showcased in the community forum (i.e. Monocular Camera, and New packages for Lunar).

Analysis. Consequently, quality practices and continuous improvement are under-prioritized. A participant commented on the current QA processes in the community, "It takes a lot of time to set things up properly, and a lot of people see that as wasted time because you are developing a new component that is doing something. You want to focus on developing your component; you don't want to focus on setting up tests, gathering data, putting [out] a simulation, [and] all this kind of collateral work."

Impact. To counter this de-prioritization of QA, the OSS core team model could possibly be replicated for QA. In the core team model, a community sets up a dedicated team guarding and enhancing the core modules of the project. The core team model has been successful in the ROS community. This model could possibly be replicated for quality by creating a dedicated team to own and guard QA practices in the community. This would elevate priority rank of quality assurance to the level similar to programming core modules.

4.3 Meritocratic Culture

INFLUENCER 4: ROS culture of meritocracy is not integrated into QA practices.

Data. The cultural traits of open source software communities are grounded in the ideological beliefs and members' motivations. The study of the ROS community indicates that status attainment, openness, freedom of choice, and the strive to innovate constitute the cultural traits of the community. However, meritocracy is a significant attribute of ROS culture.

Unfortunately, we observed that the community's cultural variables are not crafted into the implementation of QA practices. Fame and reputation are the rewards for those with superior technical knowledge, and they generously help others to resolve

their technical challenges in the community forums. There is no such visibility or reward given to those who perform testing activities or documentation.

If enjoyment is one of the participation drivers, then quality should be fun. However, this is not the case. Fun is not constructed into QA practices. Fun is also the intellectual stimulation and challenge. Quality practices are conformance to rules, standards, and processes. Consequently, QA activities do not attract contributions. One participants stated, "Maintenance! No one wants to do that. I mean I am saying I am not happy that this is actually a real problem."

Analysis. Culture stems from a Greek word "cultura," meaning "to tend, cultivate, till, educate or refine" [43]. Bennett [44] defines culture as a shared mental system that distinguishes the members of one group from another. Culture is transferred from one generation to another. Fellows & Liu [45] argued that culture is ever changing as generations add something new to the culture before passing it to the next generation.

In a social system where meritocracy dictates the social structure and technical knowledge is awarded by social merits to attain higher community status, quality practices become trivial. This becomes more problematic when the innovation's functional depth and breadth dominate the community's priorities. In a local community meeting, a highly regarded developer was introduced to the crowd as being "famous worldwide" for developing a feature that was enthusiastically appreciated. Apparently, his contribution was of exceptional technical complexity. Others had unsuccessfully attempted to deliver it previously. In his presentation, he stated, "When you contribute a new feature, why think about quality? Just develop it and put it out there." This attitude of features first and quality later shows that quality is not built into the cultural environment.

Impact. According to Crosby [46], quality should be crafted into its cultural environment. It has to be part of the organization fabric, not part of the fabric. Culture and practices should be in synergy with each other. We observed that when a practice is alienated from the community culture, its implementation and execution fail. QA practices should be aligned to the meritocracy system. Non-programming tasks, especially QA, should be rewarded "karmas" similar to answering community members' questions. The "karmas" system is a reward scheme whereby members are rewarded "karmas" (i.e., points) for helping to answer questions in the community forum. Members with high "karmas" are highly regarded in the community.

4.4 Sustainability

INFLUENCER 5: The absence of a working sustainability strategy puts constraints on the execution and the development of QA.

Data. A subject states: "[Sustainability's] always the problem because if you don't have the large exposure, the project does not have much chance to survive after it has been developed." Finding a balance between quality and stimulating growth through ongoing contributions has been a challenge for ROS. While the flow of new contributions is steady, the core team does not have the capacity to ensure their quality.

"The main challenge is basically time and resources." In addition, the absence of a working sustainability strategy has led to a resourcing issue in the software maintenance activities. The community attempted a few initiatives to attract new maintainers; however, these have been unsuccessful. One participant explained, "So it is good to get people in, but it's hard to get maintainers in, both of which will actually continuously spend some of their time triaging and contributing. That's a huge challenge, and we have not figured out a good way to get more people involved, and that, I think, is one of the biggest challenges for the project." Consequently, a high number of packages end up being orphans and unmaintained. This applies to non-code artifacts as well; some QA Wiki documentation has not been updated for years.

Analysis. A sustainable community is "one that is economically, environmentally, and socially healthy and resilient" [47]. Resilience transcends inception and the ability to produce a product but rather than the product's ability to evolve and continuously innovate and thrive. Failing to create a sustainable environment to support themselves, OSS communities usually vanish.

Impact. Sustainability is difficult to achieve; however, a project's ability to attract and retain development and user resources increase the possibility of sustainability [48]. Communities that fail to create a sustainable environment to support themselves vanish. The evolution of the community product relies on ongoing creative contributions. Hence, OSS communities need to design and implement a working sustainability strategy to support growth and innovation.

4.5 Complexity

INFLUENCER 6: The complexity of robotics systems adds challenges to the implementations of QA.

Data. An interviewed mechanical engineer defines quality as the robot functioning defect free. Simulanously, a software engineer is disappointed that quality practices are not adhering to software engineering standards.

Analysis. QA of robotic systems is a challenging endeavor. Robots are complex distributed systems, combining control, AI, concurrency and mobility. Their development is a complex interdisciplinary practice. Their life cycle varies from the traditional software. This complexity of robotics development is not reflected in the current implementation of QA processes in the ROS community.

Impact. The complexity and interdisciplinarity of robotics systems is inherent to the ROS community; it is unlikely that it can be exploited to provide better QA. We consider it as a force that additionally complicates an implementation of a successful quality management strategy for ROS.

4.6 Adaptability

We consider the organic self-alignment of practices to the community's social and cultural characeteristics. An organic modification does not occur via staged implementation nor via a

design change process, but arises informally. The community implicitly acknowledges non-conformance to industry practices. For instance, there is no formal requirements engineering process in place. Instead, requirements are collected from ideas and code contributions of the members.

The self-alignment of practices is not always successful. The difference between a community-based adaptation of practices and a standard change management project lies chiefly not in the change but in the way the change is developed and integrated into the community. An adaptation should rooted in the community context. It requires deep understanding of what the community is doing, why people participate, how did they cope with past and present changes, their informality and how an organic adaptation has developed in the past.

4.6.1 "Informlisms" Of Processes

INFLUENCER 7: The ROS community does not follow QA practices as prescribed. It prefers an organic development of practices.

Data. Processes in the ROS community tend to take an organic course to full implementation, informed by its own trial of a practice as opposed to a planned implementation that comes from a well-established change design process known to the software engineering researchers. Some of the community practices were intended to be trials, which have since become abiding. "Most of the current processes in place have been thought in flight," a participant states. Another one explains why the adherence to the code review is fluid: "There is no formal process for that, which is probably something we could improve on, but there is no fixed rule [on] what to check for."

Analysis. These voices from the ROS community are consistent with earlier analyses. According to Elliot and Scacchi [4], OSS projects are often managed informally. Scacchi [3] suggests that OSS practices *software "informalisms"*, by not adhering to the traditional engineering practice, standards, and rationale. Sometimes the informalisms are democratically agreed upon through a voting system [33]. Sometimes they emerge implicitly. For instance, the traditional code inspection is fundamentally different from OSS code review [13].

Impact. We learn from this that the implementation of QA should be organic and allow for "informalisms." Prescribing practices top-down does not work. The OSS communities prefer to reflect, deliberate, and democratically consult the wider membership before adopting a change. During this process, practice adaptation occurs. Action research regarding QA processes in OSS should definitely take this into account.

4.6.1 "Ease Of Use

INFLUENCER 8: The ROS community has an affinity for ease of use.

Data. One participant summarized this elegantly: "Prioritization should also look at how long the task would take versus how important it is—and how adaptable the community to the task. You cannot bring a game-changing thing. People would say this is too complicated; I'm not going to do it. It's open source, not everybody aiming for stability. Everybody is aiming for new things. So you want to make sure those people are not scared away with complicated processes of testing."

Analysis. The community definition of ease of use is "minimum annoyance" and an enjoyable user experience. This is in line with "effectiveness," "efficiency," and "satisfaction of use." Consequently, the community members expect that QA practices are effective and efficient. QA processes should not delay or constrain developers' dedication to innovation.

Impact. Tools and processes should facilitate innovation and not constrain the creativity and participation. Ease of use should be a factor in the implementation of QA practices, and tools in OSS communities (which is obviously a challenge).

5 CONCLUSIONS

The ROS community has adopted accepted QA practices, but it is struggling with their effective implementation and execution. We focused on what influences this implementation and execution (RQ1 and RQ2) in order to understand the variables that contribute to the success and the establishment of QA practices in an OSS community. The implementation and execution of QA practices in the ROS community appears to be influenced by social and cultural factors and is constrained by sustainability and complexity. This shapes the practices toward a community-tailored variation rather than following the traditionally prescribed software engineering recipes.

What does this tell us? The identified influencers should be weaved into the design and implementation of QA in OSS communities. This necessitates some ingenuity and boldness. In commercial software production, QA practices are prescribed and enforced by management. In the future, We aim to change the trajectory of the ROS community to prioritize QA practices higher and to execute them effectively without management in traditional sense.

External validity. ROS shares the cultural and social attributes of other OSS communities. Hence, the related influencers will likely be applicable to other communities as well. The complexity of robotics software may not be representative of many other OSS projects, but it clearly is for some.

Acknowledgments. Work partially supported by EU's H2020 programme under ROSIN project, grant agreement No. 732287. We thank the interviewees for their participation.

REFERENCES

[1] Ø. Hauge, C.-F. Sørensen, and R. Conradi, "Adoption of open source in the software industry," in *OSS* 2008.

[2] Nagy, D., Yassin, A. M., & Bhattacherjee, A. (2010). Organizational adoption of open source software: barriers and remedies. *Communications of the ACM*, *53*(3).

[3] W. Scacchi, "Understanding the requirements for developing open source software systems," *IEE Proceedings-Software*, 149(1), 2002.

[4] M. Elliott, W. Scacchi, *Free software: A case study of software development in a virtual organizational culture* ISR 2003.

[5] Wenger, E. *Communities of practice: Learning, meaning, and identity*. Cambridge University Press, 1998.

[6] E. M. Gerson, and S. L. Star, "Analyzing due process in the workplace". *ACM TOIS*, 4(3), 1986.

[7] L. A. Suchman, "Office procedure as practical action: models of work and system design". *ACM TOIS*, 1(4), 1983

[8] C. Hansson, Y. Dittrich, B. Gustafsson, and S. Zarnak, "How agile are industrial software development practices?". *JSS*, 79(9), 2006.
[9] *ISO 9000: Quality management systems — Fundamentals and vocabulary*. 2015.
[10] T. J. Halloran, W. L. Scherlis, "High quality and open source software practices," *Workshop on Open Source Soft. Eng.*, 2002.
[11] M. Michlmayr, F. Hunt, and D. Probert, "Quality practices and problems in free software projects," in *OSS*, 2005
[12] L. Zhao, S. Elbaum, "A survey on quality related activities in open source," *SIGSOFT Soft. Engineering Notes*, 25(3), 2000.
[13] P. C. Rigby, D. M. German, L. Cowen, and M. A. Storey, "Peer review on open-source software projects: Parameters, statistical models, and theory," *ACM TOSEM*, 23(4), 2014.
[14] S. Lussier, "New tricks: How open source changed the way my team works," *IEEE Software*, 21(1), 2004.
[15] E. S. Raymond, *The Cathedral & the Bazaar*. O'Reilly 2001.
[16] T. Otte, R. Moreton, H. D. Knoell, "Applied quality assurance methods under the open source development model," in *COMPSAC'08*.
[17] M. Aberdour, "Achieving quality in open-source software," *IEEE software*, 24(1), 2007.
[18] A. Khanjani, R. Sulaiman, "The process of quality assurance under open source software development," *Computers & Informatics 2011*.
[19] D. Wahyudin, A. Schatten, D. Winkler, and S. Biffl, "Aspects of software quality assurance in open source software projects: two case studies from apache project," in *EUROMICRO*, 2007.
[20] R. Stallman, "Transcript of Richard M. Stallman's speech," free software: Freedom and cooperation" NYU, 29 May 2001,"
[21] T. Abdou, P. Grogono, and P. Kamthan, "A conceptual framework for open source software test process," in *Computer Software and Applications Conference Workshops*. IEEE, 2012,
[22] K. Malterud, "Qualitative research: standards, challenges, and guidelines," *The Lancet*, vol. 358, no. 9280, 2001.
[23] A. Howson, "Qualitative research methods," *Research Starters: Sociology (Online Edition)*, 2010.
[24] K. Charmaz, "Premises, principles, and practices in qualitative research: Revisiting the foundations," *Qualitative Health Research*, 7(14), 2004.
[25] *Open Robotics*. https://osrfoundation.org/ [6-Dec-2017].
[26] http://rosindustrial.org/. [Accessed: 06-Dec-2017].
[27] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being." *American Psychologist*, 55(1), 2000.
[28] K. R. Lakhani, R. G. Wolf, et al., "Why hackers do what they do: Understanding motivation and effort in free/open source software projects," *Perspectives on Free and Open Source Software*, vol. 1, 2005.
[29] R. Pavlicek and Foreword by R. Miller, *Embracing Insanity: Open Source Software Development*. Sams, 2000.
[30] A. T. Denzau and D. C. North, "Shared mental models: ideologies and institutions," *Kyklos*, vol. 47, no. 1, 1994.
[31] G. Von Krogh, S. Haefliger, S. Spaeth, and M. W. Wallin, "Carrots and rainbows: motivation and social practice in open source software development." *MIS Quarterly*, 36(2), 2012.
[32] S. C. Özbek, "Introducing innovations into open source projects," Ph.D. dissertation, Freie Universität Berlin, 2011.
[33] M. Bergquist and J. Ljungberg, "The power of gifts: organizing social relationships in open source communities," *Information Systems Journal*, vol. 11, no. 4, 2001.
[34] J. Ljungberg, "Open source movements as a model for organising," *European Journal of IS*, 9(4), 2000.
[35] K. J. Stewart, S. Gosain, "The impact of ideology on effectiveness in open source software development teams," *MIS Quarterly*, 2006.
[36] P. A. David, A. Waterman, and S. Arora, "FLOSS-US the free/libre/open source software survey for 2003," *Stanford Institute for Economic Policy Research*, 2003.
[37] P. A. David and J. S. Shapiro, "Community-based production of open-source software: What do we know about the developers who participate?" *Information Economics and Policy*, 20(4), 2008.
[38] R. A. Ghosh, "Understanding free software developers: Findings from the FLOSS study," *Perspectives on Free and Open Source Software*, 2005.
[39] R. A. Ghosh, R. Glott, B. Krieger, and G. Robles, "Free/libre and open source software: Survey and study," 2002.
[40 E. L. Deci and R. M. Ryan, "The general causality orientations scale: Self-determination in personality," *Journal of Research in Personality*, vol. 19, no. 2, 1985.
[41] B. Luthiger, C. Jungwirth, "The chase for OSS quality: The meaning of member roles, motivations, and business models," in *Emerging Free and Open Source Software Practices*. IGI, 2007
[42] K. Lakhani, E. Hippel "How open source software works: 'free' user-to-user assistance" *Research Policy* 32(6) 2003.
[43] L. Smircich, "Concepts of Culture and Organizational Analysis". *Administrative Science Quarterly*, vol. 28, no. 3, 1983.
[44] T. Bennett, "Cultural Studies and the Culture Concept". *Cultural Studies*, vol. 29, no. 4, 2015.
[45] R. Fellows, and A. M. Liu, "Use and misuse of the concept of culture". *Construction Management & Economics*, 31(5), 2013.
[46] B. Crosby Philip, "Quality without tears: The art of hassle-free management," 1984.
[47] *Institute for Sustainable Communities*. [Online]. Available: https://www.iscvt.org/. [Accessed: 16-Dec-2017].
[48] I. Chengalur-Smith, A. Sidorova, S. Daniel, "Sustainability of Free-Libre Open Source Software projects: A longitudinal study." *Journal of Association for Information Systems*. 11, 2010