The many hats and the broken binoculars: State of the practice in developer community management

Hanna Mäenpää, Myriam Munezero, Fabian Fagerholm and Tommi Mikkonen
Department of Computer Science
University of Helsinki
Helsinki, Finland
hanna.maenpaa@cs.helsinki.fi, myriam.munezero@cs.helsinki.fi,
fabian.fagerholm@helsinki.fi, tommi.mikkonen@cs.helsinki.fi

ABSTRACT
Open Source Software developer communities are susceptible to challenges related to volatility, distributed coordination and the interplay between commercial and ideological interests. Here, community managers play a vital role in growing, shepherding, and coordinating the developers' work. This study investigates the varied tasks that community managers perform to ensure the health and vitality of their communities. We describe the challenges managers face while directing the community and seeking support for their work from the analysis tools provided by state-of-the-art software platforms. Our results describe seven roles that community managers may play, highlighting the versatile and people-centric nature of the community manager's work. Managers experience hardship of connecting their goals, questions and metrics that define a community's health and effects of their actions. Our results voice common concerns among community managers, and can be used to help them structure the management activity and to find a theoretical frame for further research on how health of developer communities could be understood.

ACM Classification Keywords
K.6.3 Software management: Software development, Software process; H.0 Information systems: general

Author Keywords
Community management; Open source; Human factors

INTRODUCTION
Open Source Software (OSS) development has become a scalable platform for open innovation. Indeed, the more popular OSS products grow, the more interests and motivations are involved in developing them. Developer communities, however, are volatile by nature and as OSS software products gain momentum, new tensions arise from, e.g., the interplay of proprietary and ideological motivations [9, 21].

OpenSym '17, August 2325, 2017, Galway, Ireland 2017 Copyright is held by the owner/author(s). ACM ISBN 978-1-4503-5187-4/17/08. https://doi.org/10.1145/3125433.3125474

Health, maturity, and stability of OSS products is only as good as the health, maturity, and stability of the community that builds and maintains them [23]. Therefore, it is in the interest of many stakeholders – including users of the software, its developers, managers and investors – to know whether the community is and remains healthy and that it is performing well. With this aim, community managers work to ensure that existing members of the community can work productively and that new people can easily join the project, eventually becoming its contributing members [15]. The community manager thus plays a central role in shaping and maintaining the pulse of the community, yet also in monitoring and reporting its state for various stakeholders. To this end, the manager needs precise insight from many aspects of the community to facilitate its work more successfully as well as to identify alarm signals that indicate needs for reactions.

OSS communities can help companies to build new types of open innovation strategies [17]. Here, successful ecosystem governance is one of the main challenges and a multifaceted understanding is needed on how developer communities can be guided towards a healthy and sustainable direction [1]. This requires understanding, e.g., how a critical mass of developers can be built and sustained [6] and how a community's ambiance can foster productivity by encouraging developers to share knowledge to help others [10].

This paper investigates the many perspectives community managers have on the health and vitality of their developer communities. We report challenges that the managers face and strategies they can use for fostering meaningful and sustainable collaborations. The article explores knowledge that can help community managers in planning improvements that lead to health, maturity, and stability of developer communities, contributing to the success of products that many companies base their businesses on.

This article is organized as follows: Section 2 discusses community health and management from the perspective of existing literature. Section 3 describes the research design, which is succeeded by results in Section 4. Implications of our findings for research and industry practice are discussed in Section 5. Section 6 concludes the work.

BACKGROUND AND RELATED WORK

Developer communities are collaborative, knowledge creating organizations whose work results in storing and exchange of software innovations [6, 18, 9]. In new communities, inter-personal communications are efficient and the leader or initiator of the project manages many aspects of the community's work [15]. However as the community grows in size, delegation and more formal practices are required to coordinate the work and to ensure that decision-making stays transparent [10]. Clear and predictable ways of working help the community to grow and invite new contributors [18].

Involvement of companies as contributors affects both the commitment of developers as well as the attractiveness of the software project in the eyes of external stakeholders [25]. Especially in those communities that are driven by corporate interests, many elements of competitiveness emerge into the collaborative work [22]. This emphasizes the need of addressing software requirements fairly [13]. Conflicts of aims can have a detrimental effect on not only the community's stability, but also on the design of the software. Therefore, a balanced representation of viewpoints needs to be present in the software's design and development decisions [8]. In this task, the community managers can have a key role in acting as "liaison officers" [15], mediating viewpoints, communicating strategies and emphasizing the many "whys" behind the decisions that shape the software product [13]. This, along with a continuous flow of contributions helps to keep the community alive [6].

Health of developer communities

Health in the context of OSS projects refers to the state of the community associated with the project [24]. Wynn defines a healthy community as "one in which the members are able to achieve sufficient returns to both satisfy their needs and encourage continued contributions" [24]. This translates to the activity and vitality of a community at a given time. A healthy community has the capacity "for producing functional outcomes and benefits (and the accumulation of these benefits) by contributing members" [24]. In healthy developer communities, people are active and responsive to questions [7]. Productivity is supported by the community's capability to both satisfy the needs of its stakeholders and to encourage continued contributions to the software source code [24, 6].

Fluctuations in the community's membership base, along with quality and frequency of contributions, are hard to predict [6]. A healthy community should have a sufficient amount of members who ensure the community's capacity for producing functional outcomes [24] and that the product of the community's work fulfills its purpose [13]. Communities that can keep their experienced developers will accumulate knowledge and expertise [20] and therefore members that become inactive should be encouraged to remain [6]. Retention of members can be crucial: it takes several inexperienced contributors and a long learning process to substitute a seasoned developer [12].

An active community member base creates a buffer that allows developers to temporarily become inactive to avoid burnout. It also provides newcomers with an environment for learning about development processes and activities with less pressure to deliver meaningful outcomes in the beginning [2]. Much of the newcomers' learning takes place as personal interactions with more expert participants and increasing capabilities of new developers requires generous training and time [16]. After interacting with the community on a repeated basis, contributors gradually assume a personal identity as its members. Through questions, refinements, and discursive challenges, meanings get continually negotiated and clarified, and opinions get exchanged. As contributors gain experience, they assume certain boundary roles and coordinative positions, naturally understanding that their own contribution in mentoring and community-building is essential [6].

The "secret" of creating value for the inexperienced lies in the social nature of the community [6]. While there are implications that new developers can be brought on board quicker in mature projects and through the support of experienced core members [4, 5], finding a common, specialized language is key [6]. Here, management activities can affect positively the tendency and willingness of the community's members to share knowledge, in turn contributing positively to many aspects of the developers' collaboration [10].

Management challenges

An important resource of on-line developer communities is the passion and devotion of developers to work to achieve common goals [6]. Developers who contribute a lot of their time to a cause have a need for focusing their efforts [2] and they can engage in heated discussions on how to develop the software in the "right" manner [19]. These conflicts create special challenges for the environment [2, 20] and experienced developers may seem uninviting to new people [6]. Here, ensuring peace for working for the intensively contributing developers requires delegation of tasks and duties. When successful, sufficient autonomy can help to create sub-communities that distribute roles and responsibilities of experienced developers for the community's members [18]. Such a division can be accomplished by modularizing tasks and allowing opportunities for independent, consensus based decision-making [11]. This creates invisible dynamics of collaboration.

A modular, loosely coupled community and task structure complicates understanding the "big picture" of the community's state, creating a need for summarizing the progress of work and decisions. For this purpose, members of the community can be required to create stories that explain the current state of the community from different angles [6]. These can regard detailed software requirements, meeting memos and action plans, yet also retrospectives on how the community's work has developed throughout a longer period of time. Faraj et al. [6] separate these into front- and back narratives. The aim of the "backstage narratives" is to improve cohesion of the community by making the several viewpoints visible and providing a comprehensive summary of decision-making.

Front narratives display a publicly available description of the community's status and its ways of working [6]. This is typically delivered on the community's website, and broadcasted in newsletters and social media. An important part of the public narrative are the documentation of the software and on-line training materials for both users and developers [15]. The front narrative is also created in community events such as conferences, meet-ups and hackathons, where users and developers can engage in ideation and creative problem solving in person [13].

The community managers can deploy a wide set of means for improving their community's state which is echoed in these narratives. Careful planning, execution and follow-through actions can be helped by a tailored, data-driven analysis approach that emphasizes community well-being over corporate culture [13]. While each community is different, their governance mechanisms are configurational [3]. For this, it is important to be able to identify which metrics are relevant for understanding each community's state, along with its weaknesses and strengths that can be used to guide the community towards its unique success.

RESEARCH METHOD

This study aims to unpack the complex role of the community manager and shed light on the challenges faced by community managers, their needs, and the means they have for performing their function. We approach this through an exploratory and qualitative lens. We chose to use semi-structured thematic interviews [14] to gain first-hand reports from community managers regarding their work. We constructed a flexible interview protocol which consisted of an initial interview questionnaire with a number of structured, open-ended questions, followed by a possibility to pose more specific questions to clarify the answers from the initial round. We also used an alternative means of deploying the protocol for participants who desired to discuss the topic freely, first using an initial Skype meeting followed by a web-based version of the interview questionnaire.

The structured part of the interview protocol sought to ask probing questions of a concrete nature that would encourage the participants to give contextual details of their work when explaining their answers. The questions were the following:

1. Please describe the community you are currently working with: Who are its members and what do they do?
2. How big would you say the community is?
3. What is your current role in collaborating with the community? Which tasks do you perform?
4. Please describe three of the greatest challenges you experience in your work?
5. What means do you have for addressing these challenges?
6. Where do you get knowledge that you need in your work?
7. Imagine, if you had a "perfect" dashboard for understanding activities of the community. What would be on it and why?

We searched the people directory LinkedIn for persons with the job title "community manager" and an affiliation with established OSS products that are widely used in both industry and academia. We selected a purposive sample of 42 persons for the study. We first contacted the participants through email by sending them the structured interview questionnaire and an invitation to participate. In this first stage, we received a total of 11 responses. Three additional respondents gave their answers through a web-based survey, increasing the final set of participants to 14 persons.

The final set of respondents included a wide variety of persons engaged in community management. 11 managers were found to work for large OSS projects. One respondent managed a programming language specific developer community that shared interconnections with many OSS projects. Two persons originated from organizations that advance standard-setting in terms of using OSS software in building industry specific infrastructure. One respondent was a senior consultant who had previously worked in several community management roles, yet today provided services and consultancy for building and managing open source software development communities. Table 1 gives an overview of the participants and their projects.

A qualitative, thematic analysis was performed on the written answers to questions 2 and 3. This resulted in emergence of 14 task categories such as (yet not limited to): "acquiring new contributors", "connecting people", "delegating work", "creating actionable metrics" and "managing online knowledge repositories". From these themes, we synthetized the seven work roles of community managers that are presented as findings of our study.

FINDINGS

Based on our analysis, seven typical roles for community managers were discovered. The following subsections describe them in detail, giving illustrative examples from the interview materials. To simplify discussion, we use the term "manager" to refer to community managers. To preserve the anonymity of the participants, we do not attribute the quotes.

Table 1. Overview of study's participants, project types, and developer community sizes.

Job title	Type of OSS project	Interviewees	Community size[1]
Community manager	Desktop and operating systems software	5	250–1300
	Application development framework	1	300
	Independent development environment	2	200–300
	Personal productivity toolset	1	100–200
	On-line publishing software	1	200–250
	Industry standard organization (OSS)	2	200–500
	Programming language user group	1	N/A
Management consultant	-	1	N/A
Total		14	

Role 1: The Spokesperson

The community manager's role is to act as a visible representative of the community as well as to create awareness of its work. Managers believe that by increasing word-of-mouth communication in and about the community, uptake of the software product can be increased. One manager described his communicative tasks to be:

Building a circle of like-minded friends on social media. Exchanging ideas with other free software leaders working in similar situations. Communicating with managers of other specialized communities. Talking with own team members and managers.

In addition, managers contribute to the stability of their community by securing both the human and financial resources for continuing the project. This usually requires focused communication and direct recruiting of new, capable developers and other community members. Face to face interactions at OSS developer conferences were reported to be essential for contacting and finding new people, yet they also served as a tool for building trust and a sense of belonging between the existing members of the community. A majority of the managers organized outreach events themselves, and participated in promotional fairs and industry events organized by others. However, there was a clear indication that outreach activities should be chosen selectively and that they should aim at tangible results:

Everyone likes community work, but unless you want to do marketing, its rarely valued by companies. This has been especially at the beginning the hardest challenge. Owning the conference and being able to use it as a tool for the community helps. Today I only work with companies that are sponsoring my conference. It's the only way to get them to pay for my work.

It was found that the managers can also work to increase the general awareness of the OSS culture through collaborations with non-profit organizations and educational institutions. Interviewees reported that they spent generous time in a public role, writing blog posts, case studies, white papers and magazine articles for the larger audience in order to e.g. educate stakeholders about the OSS culture and its many facets. In the role of spokesperson, the community manager needs strong communication skills and the ability to perform fluently in front of both virtual and live audiences.

Role 2: The Concierge

Community managers have a key role in lowering the entry barrier for new users and developers of the software. This requires forwarding new people to relevant on-line content and work tasks and helping them to get know the community as their new work environment. One manager described:

Our technical landscape is complex and there are many levels, areas, and programming languages involved. Which is probably good as we offer variety, but it's feels hard to "display" that and align it with the 'entry path' that new contributors take.

Many managers emphasized the importance of creating different starting points for aspiring contributors, however, identifying which tasks would be good for different types of newcomers was found challenging.

We have some landing pages for new and experienced hackers but keeping the 'good tasks for starters in a certain code area' in our issue tracker is not trivial. Also linking on the Wiki pages to those lists of tasks is hard as the project landscape changes over time and search queries need to be adjusted.

Managers mentored newcomers and answered their inquiries personally, yet with the aim of increasing the community's ability to support itself through social interactions. An effective approach reported by the managers was to find sociable developers that love to communicate about their work and to provide them with opportunities and space for claiming an area of responsibility. An important task of managers was found to be connecting new community members with the right people:

I would have all the members listed with their different expertise and have them organized into different groups for my own resource management. It would be really helpful in every way.

Some projects had grown vibrant sub-communities that acted autonomously in helping peers, answering questions and creating materials that facilitated uptake and use of the software. Here, the community manager's role was to ensure that questions were not left unanswered and that they were asked in the appropriate forums. While performing as a concierge, the manager needs to understand the pain points of becoming a user or a developer of the software. In this role, it is essential to be able to help new members to find their own place in the community.

Role 3: The Janitor

The community requires several socio-technical systems in order to function. These include project homepages, work-flow coordination tools[2], software development infrastructure[3], user and developer support wiki-pages, on-line question and answer forums[4], mailing lists, chats and social media[5]. The choice of these software tools with features and workflows sets the ground of the community's ways of working. Maintaining and configuring the community's technical infrastructure was in many cases the responsibility of community managers. This required facilitating use of the tools and constantly evaluating fitness of the systems for their purpose. For this, managers needed to identify needs for streamlining the processes that aid development of the software, directly impacting many aspects of the community's productivity. Prior experience as a developer or team leader was reported to help in this task. As one manager described:

> *I believe that a well-designed and well-integrated software forge will require people to learn less and simplify the workflow.*

Some community managers were central in managing software requirements and were practically resolving overlaps and redundancies in work requests. Therefore, a major emphasis of the managers' work was to communicate with the project's internal stakeholders: the people who were part of and involved with the community itself or the product of its work. Many managers participated in project meetings and summarized the developers' actions in public memos, blog posts and social media updates for the community's members. Many of the manager's tasks include maintenance and routine work that could be automated or delegated to others. However, doing so would leave the manager unaware of the pulse of the community's work, making the role of a janitor an essential part of the community manager's work.

Role 4: The Mediator

In communities with strong commercial influence, the manager has to understand viewpoints of the community's stakeholders and navigate their interests for the community's best. One of the interviewees explained the hardship of this task to be:

> *Lack of alignment in interests of participants in the project. Different companies want to get different things out of the community, so progress is slow.*

This can be especially difficult in large projects, which may have hundreds of active contributors with varying roles and personalities:

> *We have all kinds of people: programmers, artists, usability people, sysadmins, marketing, translation, user support, community work and more.*

In sponsored communities managers communicate both the main organization's goals and priorities to the software developers, likewise communicating back the community members' needs. Due to this dual nature, managers can be key in noticing anomalies:

> *With a diversity of interests comes a diversity of activities. It is difficult to know everything that is happening in the project and we occasionally find sub-projects working at cross purposes, or creating competing solutions to the same problem.*

The open and loosely coupled nature of OSS projects creates difficulties in keeping track of activities overall. There are also cases where development of the software product is highly dependent on related technologies produced by other projects, and thus the manager can be required to keep abreast of a wide array of developments at the same time.

> *We lack a complete end-to-end vision of what the platform will look like at the end of the road, which creates conflict and the opportunity for miscommunication.*

In the role of mediator, the manager needs to be on track not only regarding the stakeholders, but also with their current and planned activities. This is especially hard, as open ecosystems are especially volatile environments. Managers who assume the role of a mediator need to be able to understand complex situations, to highlight the key take-aways and to be able compose a message that is both timely and delivered in a way that is relevant for the recipient.

Role 5: The Referee

Developers can reportedly engage in heated and detailed discussions. When escalated, these conflicts can both decrease motivation and cause disruptions in the workflow of the community:

> *Programmers are usually very smart people who are proud of solutions they invented themselves. The downside is that they sometimes can't accept solutions created by others or even reasoning behind them. Typically this escalates when there are two quite contradictory ways how a functionality should be implemented.*

[2] Jira, Bugzilla, Mingle, Trello
[3] Bugzilla, Git, Mercurial, Gerrit
[4] StackExchange, Askbot
[5] Twitter, Facebook, Google Groups, IRC, Slack

One interviewee emphasized the difficulty and importance of resolving such issues in a timely manner:

> *These disputes can be very detrimental to the development project as different opinions may have large groups of people backing them.*

The role of a referee involves stepping in when the community's progress stalls due to their unability to resolve issues. Many managers expressed that identifying when this happens is not straightforward. The community manager must react to concerns of developers and communicate a solution that the community can believe in and act upon. For this reason, the manager must be highly respected and trusted, and must maintain a neutral position in order not to be perceived as biased when the role of referee is activated.

Role 6: The Leader

The community managers campaigned visibly to activate the community's members by arranging both one-off and continuing activities such as testing and defect triage events, wiki clean-up weeks and bees for improving the technical documentation. These activities were used to engage with inactive contributors and to invite, e.g., new non-technical stakeholders to participate in community's work. Sometimes these campaigns can become powerful tools and continuing traditions:

> *I am responsible for coordinating Beta test programs, which we call 'Community Acceptance Testing'. The goal of the program is to get early feedback on the main features and a formal confirmation from the community that the quality of the product is ready for public release.*

Managers described a process of grooming new sub-community managers. Here, the community manager helped to establish necessary contacts both within the key persons and related sub-communities, yet also to the key persons outside of the project. As a leader, the community manager needs creativity in designing these campaigns, as they are vital for the quality of the software, the activeness of the community's members, and for the throughput of the software development process. For successful leadership, distributing knowledge, responsibility and enabling autonomy help to build the strength of the open community.

Role 7: The Clairvoyant

While the community's socio-technical systems provide rich and detailed views into the community's activities, they also contain redundant metrics that do not contribute to increased understanding or provide help for decision-making. Many managers expressed their uncertainty in choosing appropriate metrics for each channel and in understanding which changes in these metrics require attention. Also, the lack of clarity with regard to the link between causes and effects was reported to encumber planning and reflecting on events and actions. One interviewee described a need for an end-to-end vision:

> *In general, what is the pain-point for the project? Is it awareness, acquisition, adoption, advocacy, advancement? Think of this as the community funnel: Do people know about your project? Can they easily find out about it, get the software, learn how to use it, scale their use, and in turn join your community, amplify its adoption, and participate in its improvement?*

Some managers were happy with the detailed metrics that are produced by the different statistics aggregators[6] that typically show the frequency and history of source contributions along with active users and their organizations. While each of the community's supportive systems provided their own statistics, some managers reported creating their own scripts for obtaining complementary and aggregate metrics about the phenomena that they considered important. Managers described their perfect dashboards as:

> *It's show who is new in the community and needs attention, who is dropping out and needs prodding and asking. It'd show me numbers about our user base. It'd also show me the overall quality of our software and which parts of the community are lacking behind in their work and need help.*
>
> *I'd love to see a short topic summary on [what] discussions take place where, and how 'heated' they are across the hundreds of venues for discussions that we have which are all separate.*

Two strategies for obtaining the metrics were mentioned: 1) measuring everything that could be acquired, and 2) measuring only the metrics that the manager thinks that they can affect. A possible solution for switching between high level and detailed views was illustrated as:

> *A "red-green" status indicator that would show me the health status of various community metrics first. If the metric was not-green (yellow, orange, or red), I could then drill down into the raw data and discover what needs to be addressed.*

In general, managers expressed their frustration with having to work with sub-optimal statistics. However, aggregating detailed information over long time-spans was not necessarily perceived as a valuable goal:

> *On the other hand, a lot of that big data is bullshit in my opinion, its very easy to analyze and see things, which don't exist.*

Also, it was emphasized that measurement strategies do and should change over time:

> *Due to the complexity of the endeavor of software development, this is always going to be a moving target.*

In summary, the managers use statistics, along with their extensive, tacit knowledge of the community in

[6]Open Hub, Grimoirelabs, Jira analytics, Synergy

Table 2. Questions of community managers.

Understanding the community How big is the community? Who are its currently active members? What are the members' responsibilities and competencies? How to define boundaries of the developer- and helper communities? Which organizations do contributors originate from? How many of them are being paid to work for the community?
Acquiring and sustaining contributors How to increase uptake of the software? How to increase word of mouth? How to access people that would be potential new developers? How do newcomers experience their learning? How to increase experience of new developers? Do they stay or leave? Where are the pain points? How to involve "free loaders" in contributing? Who are in the risk of becoming inactive? When would incentives be appropriate? For whom? What were the last activities of retired developers? How to increase uptake of the software? How to increase word of mouth? How to educate stakeholders about benefits / obligations of the licensing?
Creating actionable metrics What are the appropriate metrics for different platforms? What should be considered an alarm signal? How to prove success of management activities?
Facilitating collaborative work How to enable/improve communication between developers? How to encourage peer to peer mentoring? How to create new sub-communities and sub-community managers? Which tasks are good for beginners? How to automate this? Is the documentation fit for its purpose? How identify overlapping work? How to resolve conflicts?

deciding which actions they should perform to improve the community's state. However, they rely largely on a "sixth sense" in decision-making. Table 2 draws together some of the common questions that emerged from the interview material.

DISCUSSION

The seven roles described above illustrate the versatility of the community manager's work. Next, we present further observations related to the roles and narratives and discuss implications of the study for theory and practice. Last, we describe limitations of the study.

Key Observations

The renowned quality of OSS products depends on the health and stability of the communities that builds them [23]. This study has highlighted the versatile nature of the community manager who works to ensure that the ambiance of the community supports developers and helps them work productively. Next, we highlight factors that can have a positive influence on the community and knowledge that is required for planning actions that improve its state.

Governance of OSS communities includes organization of the community and principles, practices and processes according to which they should operate [12]. Most of the governance decisions reach far beyond the community manager's influence, yet the manager is key in his role in acquiring and presenting the knowledge that is required for making such decisions. Managers should be able to identify strengths, weaknesses and challenges within the current state of the community and plan, execute and recommend activities that would contribute to sustaining its health.

The "health" of a developer community has not yet been explicitly defined. As these communities are fundamentally social organizations, success of the community manager's activities can be achieved only through reflective practices. Here, managers rely greatly on their intuition, yet recognizing the benefits of statistics and aggregated knowledge about the community. The managers reported challenges in understanding what to measure and which changes in the metrics indicate a need for intervention. This, on its part reflects the ill-defined nature of community health. Our study provides a view on the commonalities that different types of unique communities possess.

Managers have trouble in getting an overview of the size, composition and internal structures of their communities, a problem that is due to the distributed nature of the socio-technical systems that support the community. While the most active members of the community currently are easily distinguishable, trouble in handling volatility of the community members was reported. This task requires recruiting new members, rewarding those who excel in their activities, along with understanding which developers are at risk of leaving the community and need encouragement.

Sustaining a critical mass of developers is key for keeping a community alive. Managers expressed that they lacked means for identifying the most recent newcomers and understanding the successes and failures in their on-boarding. Here, automating the identification and recommendation of suitable work tasks for special groups would provide clear advantages. However, this would require knowing more about the newcomers' competences, their entry paths, and activities that need facilitation in order to sustain their learning. Also, understanding discrimination of new developers could bring about new management challenges that need to be addressed. For creating a welcoming ambiance, early interventions such as identifying frustration of developers and emergent conflicts could be facilitated through understanding where heated discussions take place and which members are currently involved in several conflicts.

An active community should be able to produce outcomes that are both timely and fulfill the needs of users of the software [24]. Identifying where in the project work is stalling due to lack of developers or exceptionally active due to too much work could help recruit developers with relevant competencies. Also, pinpointing needs for modularizing the software, creating new sub-projects and distributing roles and responsibilities could be helped by finding those individuals that are burdened and those individuals that are potential for taking more responsibility.

Community managers are key in reflecting the will of a central organization back to the community. This knowledge can influence the motivation and loyalty of developers, while at the same time the reactions of the community can affect e.g. the project's funding decisions. As developer communities are an arena of versatile interests, new means and perspectives for understanding stakeholders' influence should be sought. As an example, the holistic end-to-end perspective of the community software development process, with the considerations mentioned above, has not been explored much by academia. Similarly, while contemporary analysis and monitoring tools provide lots of detail, their ability to give decision support in areas that community managers consider important appears to be lacking. Community managers would need support to tackle questions of the community structure, evolution and current activities, the ability to assess the consequences of their actions, and a level of situational awareness that is not attainable by following the massive stream of data coming directly from the socio-technical systems and indirectly through current analysis tools.

Limitations

This study aims to uncover experiences by community managers and synthesize them into a theoretical contribution. The internal validity of the results hinge on their ability to represent the perceptions and observations of the participants in a trustworthy manner. To ensure this, we employed a number of tactics in data collection, analysis, and reporting. In data collection, we recruited persons identifying themselves as community managers. This brought variety to the sample in terms of both context and content of their work. Several common themes expressed in our findings were clearly distinguishable from managers' answers, giving us confidence for presenting them as representative of the current state of practice.

We designed our interview protocol to solicit recollections and appraisals of real and concrete, rather than abstract, conditions in the respondents' own settings. By directing the respondents' attention to concrete episodes, we sought to reduce some of the bias that is inherent when using humans as information sources. We maintained the chain of evidence in analysis so that our categorizations and classifications, and the insights regarding them, were traceable back to the original participant statements. In reporting, we sought to give examples of the actual statements where appropriate, rather than interpret their meanings.

The external validity of the results is mainly limited by the representativeness of our sample. The communities in our study all have a central organization hosting developments tools and mandating practices. While we consider purposive, rather than statistical, sampling to be appropriate for an exploratory, qualitative study, we note that the limited set of participants could mean that some viewpoints are not represented. Our sample also consists of large projects with a high involvement of commercial interests, and the results may not apply to small projects or projects that are exclusively community-driven without an affiliation to commercial companies. However, bearing in mind both the versatility of respondents and the elaborate answers they gave, we believe the sample represents a broad view of the responsibilities and challenges that people working with community management in OSS development communities face today. Bearing in mind the limitations, we believe the results are trustworthy and can be transferred to other OSS contexts.

CONCLUSIONS

In this paper, we have studied the complex role of the OSS community manager, resulting in increased understanding of the challenges faced by them, their needs, and means they have for performing their function. Based on thematic interviews with 14 industry experts, we uncovered seven roles that community managers can take when gauging the state of their community and influencing it in a desirable direction.

The success of the community manager's work is important for many aspects of the health and sustainability of the community. This study reports common practices community managers use, problems they face and highlights the knowledge that can be helpful in overcoming obstacles. Results highlight the people-centric and versatile nature of the community manager's work. These characteristics are reflected on the challenges the managers experience while seeking support for their work from currently available, state-of-the-art analysis tools. With this, we provide insights on what themes the future of intelligent requirements engineering and work-flow coordination can contribute to.

ACKNOWLEDGEMENTS

This work is funded partially by the "Innovative Requirements Engineering Methods, Algorithms and Tools" research project (OpenReq) that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732463.

REFERENCES

1. Carina Alves, Joyce Aline Pereira de Oliveira, and Slinger Jansen. 2017. Software Ecosystems Governance - A Systematic Literature Review and Research Agenda. In *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 3, Porto, Portugal, April 26-29, 2017*. 215–226.
2. Kevin Crowston and James Howison. 2006. Assessing the health of open source communities. *Computer* 39, 5 (2006), 89–91.

3. Ivan De Noni, Andrea Ganzaroli, and Luigi Orsi. 2011. The Governance of Open Source Software Communities: An Exploratory Analysis. 6, 1 (2011), 1–18.
4. Fabian Fagerholm, Alejandro S. Guinea, Jürgen Münch, and Jay Borenstein. 2014a. The Role of Mentoring and Project Characteristics for Onboarding in Open Source Software Projects. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '14)*. ACM, New York, NY, USA, Article 55, 55:1–55:10 pages.
5. Fabian Fagerholm, Alejandro Sanchez Guinea, Jay Borenstein, and Jürgen Münch. 2014b. Onboarding in Open Source Projects. *IEEE Software* 31, 6 (2014), 54–61.
6. Samer Faraj, Georg von Krogh, Eric Monteiro, and Karim R. Lakhani. 2016. Special Section Introduction—Online Community as Space for Knowledge Flows. *Information Systems Research* 27, 4 (2016), 668–684.
7. Jonas Gamalielsson, Björn Lundell, and Brian Lings. 2010. Responsiveness as a measure for assessing the health of OSS ecosystems. In *Proceedings of the 2nd International Workshop on Building Sustainable Open Source Communities (OSCOMM 2010)*. Tampere: Tampere University of Technology, 1–8.
8. Jesus M Gonzalez-Barahona and Gregorio Robles. 2013. Trends in Free, Libre, Open Source Software Communities: From Volunteers to Companies. *it–Information Technology* 55, 5 (2013), 173–180.
9. Eric von Hippel and Georg von Krogh. 2003. Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science* 14, 2 (March 2003), 209–223.
10. Zilia Iskoujina and Joanne Roberts. 2015. Knowledge sharing in open source software communities: motivations and management. *J. Knowledge Management* 19, 4 (2015), 791–813.
11. Karim R Lakhani, Hila Lifshitz-Assaf, and Michael Tushman. 2012. Open innovation and organizational boundaries: the impact of task decomposition and knowledge distribution on the locus of innovation. *Harvard Business School Technology & Operations Mgt. Unit Working Paper No. 12-57; Harvard Business School Organizational Behavior Unit Working Paper No. 12-057* (2012).
12. M. Lynne Markus. 2007. The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management & Governance* 11, 2 (2007), 151–163.
13. P.M. McGarry. 2011. Built it and they didn't come: The right way to build and maintain developer communities. *Bell Labs Technical Journal* 16, 2 (2011), 251–262.
14. Sharan Merriam. 2009. *Qualitative research: a guide to design and implementation* (2 ed.). Jossey-Bass.
15. Martin Michlmayr. 2009. Community management in open source projects. *The European Journal for the Informatics Professional, X (3)* (2009), 22–26.
16. Audris Mockus, Roy T Fielding, and James D Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 11, 3 (2002), 309–346.
17. Hussan Munir, Krzysztof Wnuk, and Per Runeson. 2016. Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering* 21, 2 (2016), 684–723.
18. Siobhán O'Mahony. 2007. The governance of open source initiatives: what does it mean to be community managed? *Journal of Management & Governance* 11, 2 (2007), 139–150.
19. Daniel Schneider, Scott Spurlock, and Megan Squire. 2016. Differentiating Communication Styles of Leaders on the Linux Kernel Mailing List. In *Proceedings of the 12th International Symposium on Open Collaboration*. ACM, 2.
20. Sonali K Shah. 2006. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science* 52, 7 (2006), 1000–1014.
21. Matthias Stuermer, Sebastian Spaeth, and Georg Von Krogh. 2009. Extending private-collective innovation: a case study. *R&D Management* 39, 2 (2009), 170–191.
22. Jose Teixeira, Gregorio Robles, and Jesús M González-Barahona. 2015. Lessons learned from applying social network analysis on an industrial Free/Libre/Open Source Software ecosystem. *Journal of Internet Services and Applications* 6, 1 (2015), 14.
23. Dan Woods and Gautam Guliani. 2005. *Open Source for the Enterprise: Managing Risks, Reaping Rewards*. O'Reilly Media Inc., Sebastopol, CA.
24. Donald Wynn Jr. 2007. Assessing the health of an open source ecosystem. In *Emerging Free and Open Source Software Practices*. IGI Global, 238–258.
25. Minghui Zhou, Audris Mockus, Xiujuan Ma, Lu Zhang, and Hong Mei. 2016. Inflow and Retention in OSS Communities with Commercial Involvement: A Case Study of Three Hybrid Projects. *ACM Trans. Softw. Eng. Methodol.* 25, 2, Article 13 (April 2016), 29 pages.