What Makes a Great Maintainer of Open Source Projects?

Edson Dias[⋆] Paulo Meirelles[†] Fernando Castor[◇] Igor Steinmacher[•,§] Igor Wiese[•] Gustavo Pinto[⋆]
[⋆]UFPA, Brazil [†]UNIFESP, Brazil [◇]UFPE, Brazil [•]UTFPR, Brazil [§]NAU, USA
ecdias@ufpa.br, paulo.meirelles@unifesp.br, castor@cin.ufpe.br, {igorfs,igor}@utfpr.edu.br, gpinto@ufpa.br

***Abstract*—Although Open Source Software (OSS) maintainers devote a significant proportion of their work to coding tasks, *great* maintainers must excel in many other activities beyond coding. Maintainers should care about fostering a community, helping new members to find their place, while also saying "no" to patches that although are well-coded and well-tested, do not contribute to the goal of the project. To perform all these activities masterfully, maintainers should exercise attributes that software engineers (working on closed source projects) do not always need to master. This paper aims to uncover, relate, and prioritize the unique attributes that great OSS maintainers might have. To achieve this goal, we conducted 33 semi-structured interviews with well-experienced maintainers that are the gatekeepers of notable projects such as the Linux Kernel, the Debian operating system, and the GitLab coding platform. After we analyzed the interviews and curated a list of attributes, we created a conceptual framework to explain how these attributes are connected. We then conducted a rating survey with 90 OSS contributors. We noted that "technical excellence" and "communication" are the most recurring attributes. When grouped, these attributes fit into four broad categories: management, social, technical, and personality. While we noted that "sustain a long term vision of the project" and being "extremely careful" seem to form the basis of our framework, we noted through our survey that the communication attribute was perceived as the most essential one.**

***Index Terms*—Open source software, open source maintainers, great attributes**

I. Introduction

What makes someone *great* at her profession? Decades of research have been dedicated to providing answers to this question [6, 24, 12, 37, 38]. For instance, great instructors are effective in enabling students to learn [38]. Great managers discover what is unique about their employees' strengths and capitalize on them [6].

In the context of software engineering, Li et al. [19, 18] and Kalliamvakou et al. [13] conducted seminal empirical studies that sought to understand the attributes that help to compose great software engineers and great managers of software engineers, respectively. These works uncovered dozens of (sometimes unique but sometimes shared) attributes crucial for great software engineers and their managers. For instance, for software engineers, being data-driven, risk-taking, and curious are valuable attributes, while for managers, attributes such as availability, technical expertise, and building relationship appear to be more substantial. These findings are unequivocally essential to help growing better software engineers and managers in this ever demanding software industry.

We believe, however, that these findings might not be easily translated to the context of Open Source Software (OSS). When working on OSS, software engineers are exposed to a new landscape of challenges and barriers not frequently observed when working for a company. One of our interviewees gave an example that exemplifies this difference:

> **"***When you work for a company, your manager says: 'build a product', and then everybody is on board and will work together. We also have dedicated time to complete it, say, it should be done in a few days or in a few weeks. In OSS, there are no deadlines. Normally, you find an issue that you find interesting, you work on it whenever you have the time, and this could be weeks or months. There was an issue that was open for a year before I got the intuition on how to complete the task. This is the reality of volunteer-based OSS. It takes time. Things don't get completed right way. Even waiting for code reviews. At work, it's part of everybody's day job to do code review; at work, I can get a PR review within a day. In OSS, it is usually whenever the maintainer has free time. Sometimes it's during the weekend, sometimes you have to wait once a week.***" *(P30)*

Although our interviewee mentioned time management, other concerns are intrinsically grounded in OSS development. Maintainers should also 1) make sure that the goals of the project are transparently stated, 2) devote time to help new members (that may disappear one commit ahead), and 3) think about the long term sustainability of the project. When working for a company, software engineers would hardly bother about these concerns.

In this paper, we aim to uncover the set of attributes that are invaluable to help OSS maintainers to excel in their careers. We used the work of Li et al. [19, 18] and Kalliamvakou et al. [13] as inspiration to reach our goal. To guide our research, we designed three research questions:

RQ1. What are the attributes of great OSS maintainers?
RQ2. How do these attributes relate to each other?
RQ3. How do contributors perceive the importance of these attributes?

To answer these questions, we employed a mixed-methods approach. We started by conducting 33 semi-structured interviews with high profile OSS maintainers, then proceeded to create a conceptual framework about these attributes, and

finally performed a survey with 90 contributors. This paper's contributions include 1) a comprehensive list of 22 attributes that define the role and responsibilities of OSS maintainers, 2) a conceptual framework that relates the attributes of a great OSS maintainer, and 3) quantitative evidence about contributors' perception of the importance of the attributes.

We believe that, by understanding more about the role and the attributes of great maintainers, it is possible to evolve how projects are managed. By having maintainers that care about the interpersonal and social aspects of the OSS communities—without giving up quality and technical excellence—it is possible to enhance the landscape of OSS with more inclusive and diverse communities that will better serve the society.

II. Research Method

We followed a mixed-method approach. To answer **RQ1**, we analyzed 172 attributes mentioned during the 33 interviews with experienced OSS maintainers (Section II-A). In answering **RQ2**, we created a conceptual framework to understand, at a higher level of abstraction, the relationships between these attributes (Section II-B). Finally, in **RQ3** we conducted a rating survey with 90 maintainers and contributors aimed to understand how they prioritize these attributes (Section II-C).

A. Interviews

1) Participants: We used a convenience sampling approach by recruiting a group of maintainers close to our network. We also invited other notable maintainers (e.g., those that maintain very popular OSS projects) by reaching them by e-mail and social networks. We used a snowballing approach, asking the participants to put us in touch with other colleagues that would qualify. We repeated this process throughout the interviews. To foster diversity, when inviting the participants, we prioritize women and non-US based maintainers. We took this decision to avoid having too many "Silicon Valley" participants, as they are over-represented amongst OSS maintainers [7, 21, 22].

We interviewed 33 OSS maintainers (36% are women). Since some of these interviewees work on projects with a small number of maintainers, to protect their anonymity, and following guidelines on Ethical Interviews [30], we only mention the name of the projects if the number of maintainers is greater than 10. These maintainers work on a diverse set of projects, including: operating systems (e.g., Debian and the Linux Kernel), desktop interfaces (e.g., GNOME and KDE), social coding environments (e.g., GitLab), well-known programming languages (e.g., Python), and educational projects. Our interviewees are located in different countries, such as Brazil, Canada, Czech Republic, USA, Germany, and Portugal. On average, they have nine years of experience in OSS, although only 11 of them are paid, full-time OSS contributors. We provide additional information about our interviewees in the supplementary material upload along with this submission.

2) Interview Conduction: We conducted *semi-structured* interviews, which is a flexible type of interview, enabling the interviewer to add or remove questions as they see fit, during the flow of the interview [25]. We started by inviting 59 maintainers to participate, and only 28 accepted. We conducted the 28 interviews from April 2019 to September 2019.

During the coding procedure, we noticed that some attributes were commonly reported. For instance, communication-related attributes were mentioned in 16 out of the 28 interviews. We also noticed that fewer new attributes were mentioned after the 24^{th} interview, which gave us a signal about saturation. From June 2020 to August 2020, we nevertheless invited seven additional women maintainers to interview (five accepted). These additional interviews had two goals: (1) to increase the number of women maintainers (which was around 25% of our sample); and (2) to verify whether we had achieved saturation or not. These five interviews did not introduce any new code to our codebook, which was another sign of saturation. Therefore, we stopped recruiting, and ended up with 33 participants.

The interviews were conducted remotely using online channels (e.g., Jitsi, Whereby), according to participants' preference. On average, the interviews lasted 44 minutes (min: 26, max: 60). The audio was recorded and later transcribed. Five authors conducted all the interviews and the first author transcribed the audios, supported by `otter.ai`.

The interview was composed of five parts. First, we asked about the maintainers' background and demographics. Second, we asked questions about their job as maintainers, such as what they do and how they do it, how do they find new features, and how much time do they devote OSS. Third, we discussed specifically the topic related to what makes a great maintainer. Following the instructions of Kalliamvakou and colleagues [14], to guide this part of the interview, in our invitation email, we asked the participants to send us five attributes that they consider the most important for a great maintainer. Focusing on only five attributes would: (1) avoid participants rushing their answers, (2) require them to provide focused responses, discarding less relevant attributes, and (3) reduce the cognitive load of remembering specific experiences in the past. Fourth, we asked the participants whether they remember other great maintainers that they have worked with in the past, and describe why they think that person was great at maintaining OSS projects. Lastly, we closed the interview by thanking the participant, and asking whether they could suggest other maintainers that could be interviewed.

3) Interview Analysis: We analyzed the more than 30 hours of interviews (resulting in 88,586 words of transcripts). We performed two coding procedures, as follows.

Coding the attributes. We started with 172 raw attributes that the maintainers sent to us before the interviews. This number is higher than 165 (5 attributes $\times$ 33 interviewees) because some interviewees mentioned more than five attributes. To group attributes into high-level categories, two authors followed a card sorting approach [27]. We started by reading each of the attributes and grouping those clearly representing the same code. We then categorized the attributes in higher level clusters, based on similarities between the meanings of the codes. This process took five sessions ($\sim$3 hours each). This process was conducted using continuous comparison [31]

during the card-sorting sessions and discussions about codes and categories until they reached consensus. After almost 15 hours of discussions, 22 attributes emerged, clustered in four high level categories. These four categories are: Management, Social, Technical, and Personality. Each category has three to seven attributes. Table I breaks these categories down in detail.

TABLE I
THE FOUR IDENTIFIED CATEGORIES

Category	# of Attributes	# of Mentions
Management	7	43
Social	6	67
Technical	3	38
Personality	6	24
Total	22	172

Coding the interviews. We used open coding to analyze the interview transcripts. We start by reading the transcripts, identifying key points, and assigned them a code (i.e., a 2–3 words statement that summarizes the key point). In the context of this work, the 22 attributes identified in the previous step were used as seeds for this analysis; other codes emerged throughout the coding procedure, though. For instance, we had a "maintainer role" code that synthesizes how maintainers characterize their role, and a "gender bias" code to represent women maintainers that experienced different treatments in OSS communities. By constantly comparing the codes [31], we grouped them into categories that gave a high-level representation of the codes. This coding process was conduced by one researcher and constantly discussed with a second researcher.

B. Conceptual Framework

We developed the Conceptual Framework (CF) to describe the relation between the attributes, thus helping us to answer RQ2. The conceptual framework was inspired by the one designed by Leite et al. [17]. The conceptual framework development involved three phases: 1) concept identification, 2) CF development, and 3) CF refinement. During these phases, we interrogated our data again, in particular, looking for potential relationships among codes.

1) Concepts identification: During the interviews, we asked the interviewee to explain why the mentioned attribute was important to her. The understanding that an interviewee has about a particular attribute is what we called "concept". To find these concepts, we revisited the maintainer's explanation about the specific attribute, paying particular attention to extract the main idea from the explanation.

2) Conceptual Framework development: After identifying the concepts, we used an online tool to design the conceptual framework. When analyzing the concepts, we looked for relationship among them, which we further explored in this step. In the graphical representation, the attributes are represented by bold, upper case text, whereas the concepts are represented as smaller, capitalized text. A relationship happens when one interviewee mentions multiple concepts while describing an attribute. For instance, P29 mentioned that maintainers have to "*be willing to mentor and help contributors and having empathy. An effective communication will help them become a better mentor.*" In this statement, we extracted concepts related to Community building, i.e., to mentor and help contributors, Empathy, i.e., having empathy, Communication, i.e., effective communication, and Leadership, i.e., becoming a better mentor. We used arrows to illustrate relationships, and bold blue text to describe the participant who mentioned the concepts. Figure 1 exemplifies our conceptual framework.

Fig. 1. Illustration of our conceptual framework.

3) CF refinement: We created one conceptual framework for each category (four in total). One author created the first version of each conceptual framework. While reviewing the framework, we attempted to reach a granularity level that is not too coarse-grained that the readers not get any new insight and not too fine-grained that the readers might be confused about similar concepts. For instance, in the first version of the Social conceptual framework, we had "Encourage the contributor" and "Inspire the contributor" which were merged into a single concept. Three authors collaboratively revised each framework. Each refinement session took $\sim$2 hours.

C. Survey with Contributors

1) Survey Design: We started our survey by getting consent from the participants, explaining the goal of the study, the team involved, and its volunteer nature. We then asked some demographics questions. The next section was requesting the participants to rate the importance of a subset of the attributes identified in the interviews. We thought that asking the participants to rank all 22 attributes would introduce a heavy load on them, which might lead to drop-outs in the middle of the questionnaire. Instead, we focused on ranking the three most recurring attributes per category.

Each participant rated the 12 attributes in terms of an asymmetric scale, with the options: 1) Essential, 2) Worthwhile, 3) Unimportant, 4) Unwise, and 5) I don't know. This scale was also employed in the work of Begel and Zimmermann [4], and it is based on the scale defined by Kano et al. [15]. This asymmetric scale makes a distinction between must-be (Essential), attractive (Worthwhile), and undesirable (Unwise) quality, in the context of customer satisfaction.

Except for the demographic questions, all other questions were required. A brief description introduced the questions related to the attributes. This description provided the definition and a representative quote (from the interview) of the attribute. By following this for all attributes, respondents would quickly understand and answer them. The questionnaire is available as supplementary material. To illustrate, Figure 2 presents an example of a question about the Vision attribute.

2) Survey Conduction: Before deploying the survey, we conducted a pilot to identify eventual problems with the

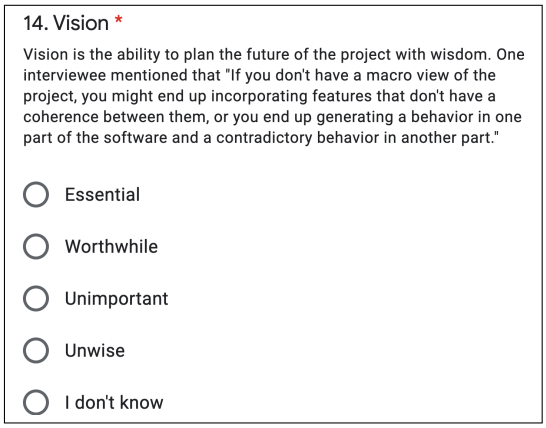

Fig. 2. A screenshot of the survey question for the "vision" attribute.

questionnaire and to ease the understanding of the questions. We invited eight people (three Ph.D. students in Software Engineering and five Ph.D. in Software Engineering) to answer the questionnaire and provide feedback (five answered the pilot survey). Their answers and feedback answers helped us to make some questions clearer. For instance, one researcher suggested the inclusion of a "prefer not to say" option in the gender question. We time boxed the answers and found that they took, on average, five minutes. We suggested this time when inviting the participants to complete the questionnaire. We did not consider the pilot answers in our analysis.

Our questionnaire was not restricted to OSS maintainers; anyone who contributed to OSS in the past was welcome to participate. Our rationale was that, since OSS contributors have to interact with maintainers to have their patches evaluated, these interactions could shape the contributors' perception of what makes a great maintainer. For instance, it was noted elsewhere that when contributors do not receive constructive feedback, they become less likely to contribute again [29]. Instead of sending unsolicited emails to GitHub users, a practice perceived as "worse than spam" [3], we followed a multi-step recruitment approach. First, we invited the 33 maintainers who participated in our interview to answer our questionnaire. We created a blog post and asked them to share it with their private networks. Second, we posted the blog post on social networks such as Twitter, Facebook, and LinkedIn (in a group with 26k members). We also ran a paid advertising campaign on r/opensource, a Reddit group with 121k members. For seven days, we received 21 clicks and paid 19.56 USD. We checked the timestamp of the clicks and the answers and estimate that no Reddit user filled the questionnaire, so we closed the campaign in one week. Finally, we shared the survey with contacts in our network ($\sim$40 OSS developers), asking them to forward this to other colleagues.

We applied some principles as an attempt to increase survey participation [26]. We employed the *social benefit* principle by donating 100 USD to one OSS project (if we received more than 100 answers), which respondents could vote for. We employed the *authority* and *credibility* principles by introducing ourselves as professors and researchers from accredited universities. The *brevity* principle was employed by asking closed and direct questions as much as possible. After four weeks, we closed the survey with 90 answers. These answers come from all continents except for Australia. We received answers from 17 women, 3 non-binary, and 70 men. In terms of experience with OSS development, 20 have between 0 and 2 years, 21 between 3 and 5 years, 17 between 6 and 8 years, 10 between 9 and 10 years, and 22 reported more than 10 years of experience. The respondents contributed to many high profile OSS projects, including Linux Kernel, Debian, Tensorflow, GCC, GNOME, OpenStack, and Kubernetes.

3) Survey Analysis: Based on the scale used to rate the attributes, we employ three metrics to analyze the survey results. These metrics were previously employed in the study of Begel and Zimmermann [4]. We avoid converting the ordinal data from the survey into numerical data by dichotomizing [16] it in the definition of the metrics. The first of the metrics aims to capture top-rated attributes considered essential by (most of) the survey respondents. It is the percentage of Essential responses among all responses for a given attribute:

$$\%Essential = 100 \times \frac{Essential}{Essential + Worthwhile + Unimportant + Unwise}$$

The second metric captures the overall good disposition of the respondents towards the attributes. It calculates the percentage of respondents that considered each attribute as Essential or Worthwhile:

$$\%Good = 100 \times \frac{Essential + Worthwhile}{Essential + Worthwhile + Unimportant + Unwise}$$

The third metric looks at the bottom-rated attributes considered less important or even negative by the respondents. It computes the percentage of respondents that considered each attribute Unimportant or Unwise:

$$\%No\ good = 100 \times \frac{Unimportant + Unwise}{Essential + Worthwhile + Unimportant + Unwise}$$

We rank the attributes based on the values of the metrics. We also attempt to discern whether the respondents' experience with OSS development is associated with them considering an attribute to be Essential. We dichotomize [16] respondent experience in terms of experienced (9+ years of OSS experience) and non-experienced (0-5 years). Our rationale is that experience may bring different perceptions to contributors, and, therefore, affect their opinions about the attributes. We also considered analyzing whether gender has a relationship with rating an attribute as Essential, but did not do so because of the low number of women and non-binary respondents.

To understand the strength of the association between respondent experience and an attribute being rated Essential, we calculate the odds ratio. The odds ratio establishes the odds that an outcome will occur (attribute considered Essential) given a particular exposure (respondent experience level) [32]. When reporting the odds ratio, we also report the confidence interval at 95% confidence level. We only report the results where the size of the confidence interval was less than two.

III. What are the attributes of a Great Open Source Maintainer?

Our qualitative coding procedure allowed us to summarize the 172 attributes into four categories, listed in Table II. For each attribute, we provide its definition and a representative quote. Our participants are referred to as P1–P33.

A. Management

This category refers to attributes that help managing an OSS project, in the sense of understanding the vision of the project, establishing and communicating project goals, proposing timelines, managing the quality of documentation, etc. In the following, we highlight three management attributes:

Availability. This attribute refers to the response time to answer a question, or to provide feedback to patch proposals. It was the most often mentioned attribute in this category (cited by 15 out of 33). P9 mentioned that *"being available is useful when a person can help another, supporting those who are starting, avoiding them going the wrong way and wasting time"*. Maintainers might want to give feedback faster, avoiding very long wait times which may discourage contributions, as stated by P20: *"long responses generate dissatisfaction"*.

Discipline. A disciplined maintainer ensures that the process and guidelines are followed. We found six mentions to this attribute. P23 reported that maintainers might *"have a well-defined agenda and know what to do, avoiding leaving open tickets"*, while P25 stated that *"defining tasks for contributors is a strategy of establishing their commitment to the project"*.

Vision. Having a global view of what to achieve with the project in the future might help maintainers define priorities. Four participants mentioned this attribute, such as P1, who argued: *"a long-term vision needs to be developed in close cooperation with stakeholders"*.

B. Social

In this category we group attributes related to how maintainers deal with other contributors, who might not speak the same language or might not know how to get started.

Communication, mentioned by 18 out of 33 interviewees, is the ability to exchange information in a sensible way. Having a good communication channel with the community is important in several ways. For instance, *"it is important to look for external feedback after any major changes in the project"* (P9). In addition, P3 highlighted that *"it is important to have a good communication with other maintainers"*.

Empathy. By sharing the problems of others, maintainers can deal better with contributors with different backgrounds. Among the 16 respondents who reported this, P2 reported that it is important to be "*extremely careful with the way you write and say things so you don't look aggressive or impolite"*. P15 complemented by saying that *"we should discuss ideas, and not judge who presented it"*.

Community building. Nine respondents mentioned that it is important to build a collaborative culture. This involves helping newcomers to get on board and motivating existing contributors to finish their tasks. As P7 stated: *"[...] no issue is more crucial to the project's future than the development of a welcoming environment for new members."* P25 mentioned that "*some maintainers forget that many contributors are volunteers [...], if a person has already contributed to a certain part of the code, why not direct that same person to make changes in other parts? This motivates the contributor.*".

C. Technical

This is about technical skills that are used to run an OSS project (e.g., fix bugs or add new features). This category only had three attributes, which we discuss next.

Technical excellence. Mentioned by 24 out of 33 interviewees, this was the most mentioned attribute. P19 summarized this attribute with a provocation: *"If you don't have a high technical knowledge, how will you ask the contributors to do things with quality?"*. However, technical excellence goes beyond coding activities; P2 reported that maintainers should also perform code review to find bugs and give feedback. P32 reported that technical excellence is something that maintainers "*should look for regularly; the greater the knowledge in technologies, the easier it is to make decisions*".

Quality assurance. Being the quality gatekeeper—raising the gate when potential defects arise and keeping it closed while defects do not vanish from the codebase—is also an important attribute of great maintainers. P3 summarized this attribute as: *"the maintainer is responsible for making sure everything is working, i.e., make sure 1) that the pull request is following the contribution guidelines, 2) that the tests are passing, and 3) that they are not 'breaking' anything so they will not release an inadequate version. The maintainer should guarantee that this process is being conducted properly."* This attribute is not only about testing and code reviews. P31 highlighted that good documentation is also a strong quality indicator.

Domain experience. Having domain experience might help maintainers to understand idiosyncrasies in the codebase. P12 reported that *"knowing the problem domain makes me more engaged in the project"*, which is somehow aligned with the "scratch your own itch" [23] rationale.

D. Personality

When maintainers express how they think, feel, and behave in their interaction with the community, they refer to personality aspects. The personality attributes are the following.

Motivation. This relates to factors that encourage maintainers to do something with enthusiasm, as mentioned by P17 "*enthusiasm is in the person who believes in the project, and also believes in the ability to transform things, to make things work, have the feeling that they can change the world with small changes*". Being motivated is a foundation for everything else, as explained by P19: "*if you are well engaged, you also infect other people in the community and create a feeling that we are in a community and we are doing something together.*".

Open minded. This is about being open to listen to new ideas. P1 stated that "*the maintainer needs to be innovative in the community. Innovators have an open mind, it is necessary to let go of the ego and become an apprentice again... Open*

TABLE II
ATTRIBUTES OF A GREAT OSS MAINTAINER

	Attribute and definition	Quote
Management	**Availability**: to be available to answer questions whenever possible.	"*Availability is about how much time you can devote to the project. You can't become a core maintainer and then leave because you're busy. You should know how to dose your availability with your dedication to keep something you would like to see still working and plan it according to your schedule with your life so that you don't leave the project behind.*" (P21).
	Discipline: to make sure that the process, the guidelines, and the code of conduct are being followed as close as possible.	"*I will not be able to solve everything immediately, naturally, I just need to have organization to know what I need to do now and in the future.)*" (P30).
	Vision: to depict how a project could develop in the future, and plan for this.	"*If you don't have a macro view of the project, you might end up incorporating features that don't have a coherence between them, generating a behavior in one part of the software and a contradictory behavior in another part.*" (P5).
	Documentation: to reinforce the importance of documenting the software	"*It is really important that the project has updated documents. We document everything: from problems to source code. Therefore, it can be easy for newcomers to understand the requirements that the maintainer might not know in detail, etc.*" (P6)
	Project management: to be able to lead the work of a team to achieve bigger goals.	"*The project maintainer needs to be a good facilitator of reaching a decision. This means greater managerial responsibility for the project.*" (P32).
	Sustainability: to promote the project to fulfill its goals, to survive changes, and to incorporate new demands over time.	"*It is essential that you have some notion of marketing and that you know how to promote this project in some way.*" (P19).
	Transparency: to make available and disseminate all information about the project.	"*If you want people to work on your project, you need to be very transparent about what the project is, the repository of everything about the project, if the project is for a company, what is the company, if the company is supporting the project.*" (P6).
Social	**Communication**: to have the ability to exchange information in a sensible way.	"*It is about how to explain a problem or an answer that is understandable to anyone, regardless of the level of experience and context; this helps a lot. Knowing how to express yourself well, formatting your thinking in a clear way facilitates this asynchronous interaction.*" (P12)
	Community Building: to help and incentivize new members to keep participating into the project.	"*Knowing how to encourage new team members, that is, knowing how to assign challenges and tasks to encourage new team members*" (P23)
	Empathy: to have the ability to understand and share the feelings of another.	"*When you have the empathy, you have the ability to think from their point of view, and then you could understand and experience the pain that they went through. If somebody has gone through and used your project and fill an issue, that is a gift; that is not just somebody complaining.*" (P1)
	Leadership: to provide guidance to the project and the community, empowering others to contribute and make decisions.	"*There are maintainers who are very attached to the code and do not accept external contributions. The maintainer needs to be democratic with the community.*." (P5)
	Pedagogy: to teach others the basic contribution steps, e.g., how to get involved, how to walk through, or how to find an issue.	"*When someone contributes to your project, you have to provide technical comments, show that sometimes the code is wrong, etc. You have to do this in a non-aggressive way and you have to efficiently and effectively write the information to the person, in the way you expect the code to be written. You have to be a teacher to do this.*" (P13)
	Relationship: to sustain a good relationship with stakeholders (e.g., upstream and downstream developers).	"*The closer you are to your stakeholders, the better. You have to see what is happening, to be able to react quickly. [...] We have some insights about someone if they has already worked for an OSS community, but we also need time to build trust in this kind of relationship.*" (P4)
Technical	**Domain Experience**— to have experience in the domain of the project.	"*The maintainer has to be evolving not only technically, but also in the domain of that project; they must master more and more the software business.*" (P30).
	Quality assurance— to maintain a desired level of quality on the project, paying attention to every stage of the development cycle.	"*The maintainer is responsible for making sure everything is working, i.e. they have to make sure 1) that the pull request is following the contribution guidelines, 2) that the tests are passing, and 3) that they are not "breaking" anything so they will not release an inadequate version. The maintainer should guarantee that this process is being conducted properly.*" (P3)
	Technical excellence— to continuously learn new skills while trying to enhance the quality of her code.	"*Technical skill is important. If you don't have that skill, you won't be able to work things out on the project. You will not be able to give the final say in the face of a problem.*" (P6)
Personality	**Motivation**: to be energetic and eager to work on the project.	"*If motivation doesn't happen, they can't do anything else; they can't be be a good programmer, can't be available, can't have good communication, can't develop technically in the project domain. Being motivated is a basis for everything else.*" (P18).
	Open Minded: to be willing to listen and consider new ideas	"*Have some kind of vision for where the project is going. But being open about that, so open to other ideas or open to things not quite working out as expected*" (P20).
	Patience: to tolerate delay/trouble without getting upset.	"*Sometimes the maintainer wants a lot of things at once, so the contributor might ending up getting lost. It is important to be calm, and walk one step at a time.*" (P25).
	Confidence: to be certain about her own abilities or qualities.	"*Among several solutions, there must be a confidence to choose one and execute it. Trust leads to a good acceptance of the maintainer by the community.*" (P17).
	Diligence: to work carefully and persistently towards a goal.	"*I think in order to maintain a project well, you have to be very diligent about working through the communication overhead, dealing with bug reports, etc.*" (P8).
	Responsibility: to have a commitment and a consciously dedication to the project.	"*The person in the role of maintainer needs to be responsible for not being afraid and making a commitment to provide a non-hostile environment for the community.* " (P17).

your mind to discuss matters that you assume are 'truths' and to hear opinions different from yours". The maintainer might want to be receptive to a wide variety of ideas, arguments, and information: "*it's mostly about when someone suggests something that you didn't think about before. Instead of being defensive and trying to criticize, you have to think that the person is trying to help you*" (P17).

Patience. To accept delay without becoming angry. According to P20, "*patience enables [maintainers] to take stock of the situation, to understand what is required, and wait while they build the capacity to take appropriate and effective action*". P27 complemented by saying that: "*patience leads to clarity about the future of the project; it allows the maintainer to see the root of the problems more clearly.*".

IV. How these attributes relate to each other?

Figure 3 presents the conceptual framework for all categories.[1] Each conceptual framework has a core concept, highlighted in yellow. A core concept is a concept related to at least three different attributes. We also highlight in bold with underline the concepts that appear in multiple categories.

A. Management

Figure 3(a) illustrates the conceptual framework of the "Management" category, and its attributes: *Discipline*, *Project Management*, *Availability*, *Documentation*, *Transparency*, *Sustainability*, and *Vision*. We identified 18 concepts and their relationship with our attributes.

"Sustain a long term vision of the project" was identified as a core concept, mentioned by seven participants when describing four attributes: *Discipline*, *Project Management*, *Vision* and *Sustainability*. This concept helps maintainers to find and establish project goals, which could also be the motivators for future actions. When talking about discipline, P11 said that *"sometimes a person is taking over the project and they say 'I saw that this thing was missing, so I spent the whole week developing it'; most of the time this was not in the scope of the project. This is something to be highlighted in OSS projects. It is very easy to just make a contribution that is not relevant; it is a community and not a client, the person thinks they can do what they want. It is easy to lose focus"*. On the other hand, in the context of project management, according to P6, *"setting expectations up front is important. For example, what are the project's expectations for 10 years from now? Will the scope change? Will there be a drastic change?"*

It brings us to another recurring concept: **"Define a roadmap"**. It was mentioned by three participants when describing *Discipline* and *Project Management*. Defining a roadmap was considered important to manage expectations and organize the goals so the project can evolve within the scope. In that sense, P23 reported that *"having a well-defined schedule, knowing what to do not to leave tasks open [...]"*. Later on, the same participant mentioned that maintainers need to *"determine roadmaps, things that can be done and that can be incremental, or evolve"*.

In another concept, **"Delegate tasks"**, was mentioned by two participants when describing *Discipline* and *Project Management*. P23 and P25 reported that the maintainer should not concentrate tasks but instead delegate them. This increases the team productivity, and the planned tasks are more likely to be completed. From the discipline perspective, delegation was related to contributors' commitment to the project. P25 explained that *"maintainers should not do the contributors' work. This is extremely bad in terms of community health. Defining the tasks for the contributors is a strategy to establish their commitment to the project."*.

"Take care of the docs" was mentioned by six participants when describing *Transparency* and *Documentation* attributes. Regarding documentation promoting the *transparency* of the project, P6 mentioned that *"when you want people to work on your project, you need to be very transparent about what the project is, [...] if the project is for a company, what is the company, if the company is supporting the project. These things must be documented appropriately."*. In a more general sense, P28 said that *"[...] it is very important that maintainers ask the contributors to keep the project documentation updated; this helps disseminate the project"*.

To manage all these demands, it is important to **"balance work and life"**, a concept related to *Project Management* and *Availability*. In terms of *Availability*, P21 mentioned that *"you can't become the main maintainer of the project and then leave because you were busy. You need to know how to measure your availability to keep something you would like to work on and plan it according to your life, so that you don't leave the project in trouble"*. While harmonizing the personal life with the projects' activities, P2 said that the maintainer needs to establish priorities, organize an agenda with contributors, be flexible with days and times dedicated to the project.

> ***Relationship 1:*** to **sustain a long term vision of the project**, maintainers ultimately should **define a roadmap**, **delegate tasks**, and **take care of the docs**. In a volunteer-based work, these activities must be carefully thought in order to not impact **work and life balance**.

B. Social

Figure 3(b) illustrates the conceptual framework of the "Social" category and its attributes, *Empathy*, *Leadership*, *Communication*, *Pedagogy*, *Community building*, and *Relationship*. According to our analysis, we discovered 17 concepts. Out of these, we observed three core concepts in this category.

"Extremely careful/polite" is the most recurring core concept mentioned by seven participants when describing *Empathy*, *Communication*, *Leadership*,*Pedagogy*, and *Community Building*. In terms of Empathy, P13 argues that *"it is absolutely essential that you make technical comments, but you can do it in a welcoming way, always kind and using the appropriate language, not using any kind of aggressive communication."*P25 suggested that, to become a better maintainer,

[1]All conceptual frameworks (as well as their different revisions) are available as supplementary material (uploaded along with this submission).

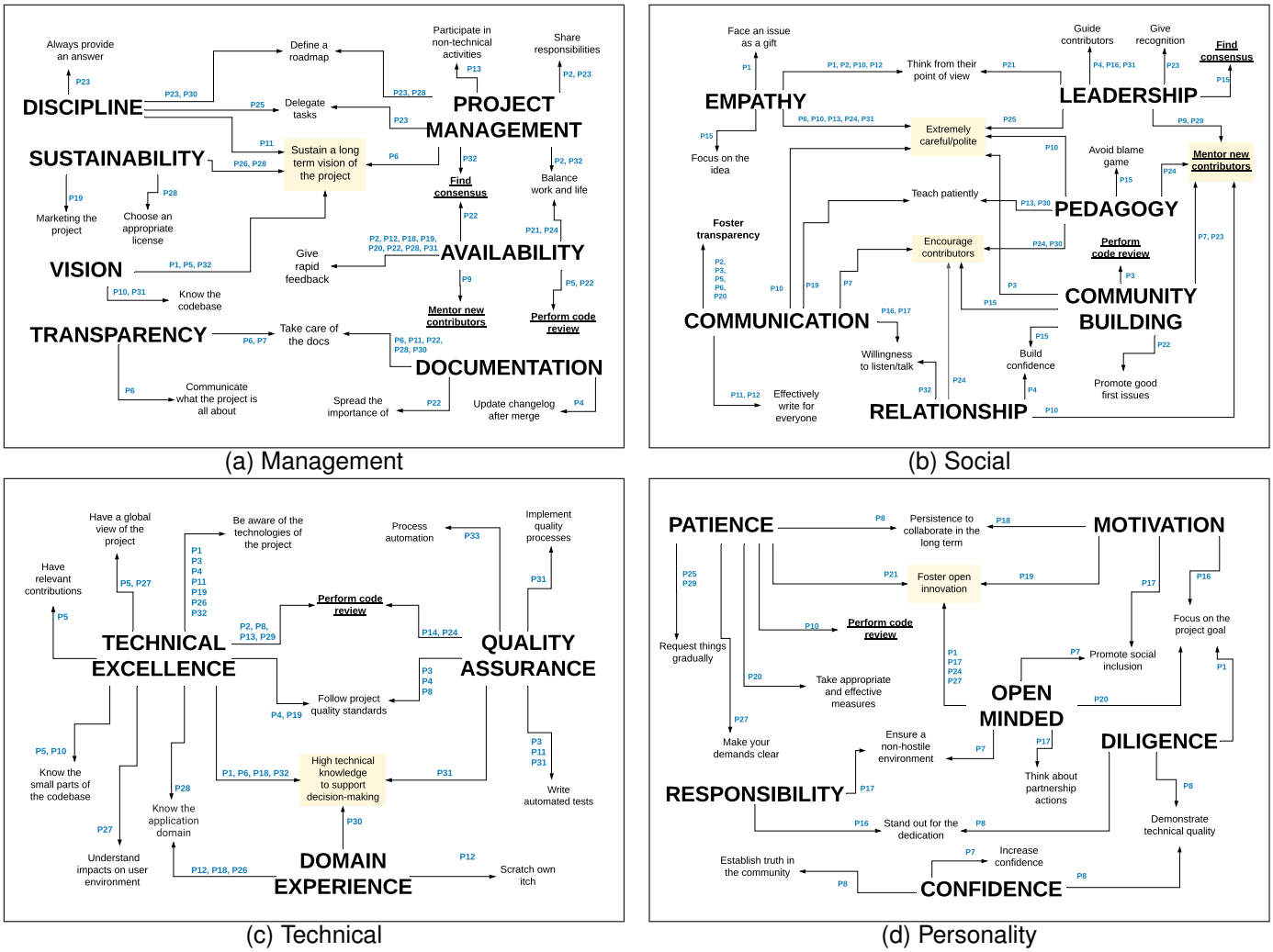

Fig. 3. Conceptual frameworks of our four categories. Core concepts are highlighted in yellow. Concepts in bold and underlined are those that appear in multiple categories.

"I am always very careful; I should thank the contribution and then make the considerations". Being polite also helps to build a community, since *"there are people sending code, people wanting to contribute, so [the maintainer] needs to pay attention revising that code, being didactic to teach the best way. The project can get better from these contributions."* (P3).

"Encourage contributors" and **"Mentor new contributors"** also appear as core concepts. Although they might sound similar, they have their differences. **Encourage contributors** is about empowering contributors to finish a task and engage in other activities. As a community-building strategy, P15 suggested giving commit rights to active contributors, since it will further motivate them, fostering a collaborative culture. In the context of pedagogy, P24 said that some maintainers carefully explain how a patch can be improved, instead of rejecting it: *"they carefully explain to encourage future improvements."* A less hierarchical and more decentralized community increases everyone's engagement, incentivizing the contributors to engage with the project. *"More collaboration, more creation, less hierarchy. Encourage contributors, you'll see them more engaged [...] this generates an emotional commitment to the project"* (P15). Different from **"Encourage contributors"**, **"Mentor new contributors"** guiding and helping newcomers to the project. When mentoring new contributors, P23 reported that it is important to delegate suitable tasks for newcomers as a community-building strategy.

Other recurring concepts include: **"Think from their point of view"**, **"Build confidence"**, and **"Willingness to listen/talk"**. According to P1, it is important *"when you have the empathy you have the ability to think from their point of view, and understand and experience the pain that they went through"*. In terms of leadership, P21 shared that she *"was a little traumatized by this kind of 'review', thinking that I was right without willing to understand one's opinion. For me, it is very important to know that one made a mistake and understand different opinions; it is something that I bring to my life"*. In the context of **"Build confidence"**, P4 mentioned that, to create good relationships, one must acknowledge that

"if a person has already worked in a software community, they have some credibility, but we can only be sure when we build trust". Therefore, it is necessary to create ways to strengthen the ties and build trust among team members. Finally, **"Willing to listen/talk"** is about finding the time to sit with someone and talk. P16 said that *"direct communication helps to humanize the community environment."*

> ***Relationship 2:*** Being **extremely careful/polite** seems to be the bedrock to **encourage** and **mentor new contributors**. To **build their confidence**, maintainers should be **willing to listen and talk**, which in turn will allow them to **think from another person's point of view**.

C. Technical

Figure 3(c) illustrates the conceptual framework of the "Technical" category and its attributes: *Technical Excellence*, *Quality Assurance*, and *Domain Experience*. **"High technical knowledge to support decision-making"** was identified as a core concept, was mentioned by six participants describing *Quality Assurance*, *Domain Experience* and *Technical Excellence*. In the context of technical excellence, P18 said that *"it is important that the maintainer knows the topic technically well to help the developers. The technical capacity must be much more comprehensive as the ability to master the subjects of Computer Science."* Also, P31 mentioned that *"tests and documentation are indicative of quality, OSS has to provide that for the community, the quality."*

"Follow projects quality standards" was mentioned by five participants describing *Quality Assurance* and *Technical Excellence*. The maintainer needs to follow the standards to develop roadmaps for configuration and project management. Describing technical excellence, P19 said: *"[...] if you do not have technical quality, it is difficult to keep the quality of the project. If you don't have a high level of knowledge of project technologies, how are you going to ask contributors to do things with quality?"*. P3, about quality assurance, mentioned that *"[the maintainer] has to ensure that the contributors are following the standard, [...] the automated tests are passing and that the contribution is not 'breaking' the code."*

The third concept for this category is **"Know the application domain"**. It describes that understanding the application domain is also important. P18 said that *"the maintainer has to be evolving not only technically, but also in the project domain, they must increasingly dominate the software business."*

> ***Relationship 3:*** To have the **high technical knowledge to support decision-making**, maintainers should **know the application domain**, **be aware of the technologies of the project**, and have the experience to **implement a quality process** to **perform code review** and **follow project quality standards**.

D. Personality

Figure 3(d) illustrates the conceptual framework of the "Personality" category, and its attributes: *Patience*, *Motivation*, *Responsibility*, *Open-minded*, *Diligence* and *Confidence*. The core concept **"Foster open innovation"** was mentioned by six participants when describing three different attributes: *Patience*, *Motivation*, and *Open-minded*. In the context of motivation, P19 described that the maintainer *"must be engaged, it motivates other people in the community and creates a feeling that we are in a community and we are doing something together [...] and the growth of the project depends on different ideas."* In the same sense, but talking about being open-minded, P27 said that *"the maintainer must have a more open and innovative mind [...] The way the maintainer thinks should not be 'the absolute truth'."*

Finally, P7 mentioned that the maintainer can create a friendly environment and show that the product can help in the social environment. These actions can give rise to perspectives that include people to be part of something. Thus, the concept **"Promote social inclusion"** emerged. This is observed in a quote from P17, *"[...] enthusiasm is in the person who believes in the project, and also believes in the ability to transform things, to make things work, have the feeling that they can change the world with small changes"*.

> ***Relationship 4:*** To **foster open innovation**, keeping the **focus on the project goal** and **establish truth in the community**, maintainers should **stand out for the dedication**, have the **persistence to collaborate in the long term**, and **ensure a non-hostile environment**.

E. Intra- (and inter-) relationships

The concepts and relationships discovered here are (what we called) *intra*-category, since they connect attributes within the same categories. However, we also found *inter*-category relationships: when the concepts are connecting attributes among different categories. We observed three concepts with this characteristic, highlighted in gray in Figure 3. The **"Perform good code review"** concept was the most comprehensive one: it appears as part of all the four categories: social, management, technical, and personality. The other two concepts that crosscut the categories are **"Find consensus"** and **"mentor new contributors"**, which appeared in the management and social categories. Since we could not present all conceptual frameworks in this paper, we leave for future work a holistic observation of these relationships.

V. How do contributors perceive the importance of these attributes?

Table III presents the values of the three metrics we computed based on the survey responses (Section II-C3). We start by examining, for each attribute, the $\%Essential$ metric, which highlights the attributes that stand above the others, even considering that our interviewees mentioned all of them. For this metric, we have a clear winner: Communication. This attribute is considered Essential by 76.67% of the respondents and does not have a single vote for Unimportant or Unwise. The second attribute with the highest percentage of votes

TABLE III
RESULTS OF THE RATING SURVEY.

Attributes	% Essential	% Good	% Not good
Communication	76.67	100.00	0.00
Quality Assurance	57.78	97.78	2.22
Community Building	50.00	94.44	5.56
Empathy	43.33	96.67	3.33
Vision	43.18	89.77	10.23
Open Mindedness	41.11	96.67	3.33
Motivation	41.11	85.56	14.44
Patience	40.91	94.32	5.68
Technical Expertise	38.89	91.11	8.89
Domain Experience	36.36	82.95	17.05
Discipline	34.83	88.76	11.24
Availability	24.72	88.76	11.24

for Essential is Quality Assurance, with 57.78%, followed by Community Building (50.00%)

The $\%Good$ metric exhibits consistently high values for all the attributes. The lowest percentage we observe for this metric is 82.95%, for the Domain Experience attribute. The Worthwhile rating worked as a baseline, since all the attributes not only were mentioned during the interviews but were the 12 most frequently cited. A corollary of this high value of $\%Good$ is that the percentages for the $\%Nogood$ metric are consistently low. The attributes with the highest values for $\%Nogood$ are Domain Experience (17.05%), Motivation (14.44%), Availability and Discipline (11.24%), and Vision (10.23%). Availability is the one with the lowest value for $\%Essential$. It is the attribute with the lowest difference between $\%Essential$ and $\%Nogood$; the former is higher than the latter by a factor of 2.2.

We calculated the odds ratio to quantify the strength of the association between respondent experience and an attribute being rated Essential. For Patience, the odds ratio is 0.232, i.e., a not experienced respondent is 4.3 times more likely to consider this attribute Essential than an experienced one. The confidence interval at the 95% confidence level is (0.083,0.645), with a p-value of 0.005. This confidence interval suggests that, with 95% confidence, a not experienced respondent is between 1.55 and 12.05 times more likely to consider this attribute Essential. Applying the Bonferroni correction, i.e., dividing the target significance level of 0.05 by 12 (the number of statistical tests we perform) we obtain a significance level of 0.004167. Since the p-value of 0.005126 is greater than this, we cannot ascertain statistical significance. For Communication, the odds ratio is 0.533, i.e., a not experienced respondent is almost twice more likely to consider it Essential. At the 95% confidence level, the confidence interval is (0.182, 1.562), although the p-value does not indicate statistical significance. The odds ratios for these two attributes suggest that less experienced developers perceive a stronger need for communication and patience from other project contributors. This reinforces Relationship 2 (Section IV-B) with quantitative data.

On the other hand, for the Motivation and Domain Experience attributes, the odds ratios are 2.154 and 1.75, respectively. According to the responses we received, this evidence means that an experienced respondent is more than twice more likely to consider Motivation Essential than a not experienced respondent and about 1.75 times more likely to consider Domain Experience Essential. The confidence intervals are (0.828, 5.599) and (0.660, 4.639), reinforcing this result, although it was not possible to ascertain statistical significance.

TABLE IV
RESULTS OF THE POST-HOC FRIEDMAN-NEMENYI TEST FOR THE COMMUNICATION ATTRIBUTE. A P-VALUE LOWER THAN 0.004545 INDICATES THAT IT IS POSSIBLE TO REJECT THE NULL HYPOTHESIS THAT COMMUNICATION WAS RATED SIMILARLY TO ANOTHER ATTRIBUTE.

Attribute	p-value
Communication	1.000000
Empathy	0.018169
Community Building	0.101391
Patience	**0.001672**
Open Mindedness	0.009216
Motivation	**0.001000**
Vision	**0.002479**
Discipline	**0.001000**
Availability	**0.001000**
Technical Expertise	**0.001000**
Quality Assurance	0.599034
Domain Experience	**0.001000**

We employed Friedman's test to determine whether any of the 12 attributes has been rated consistently higher than other attributes by the respondents. The classic example of the usefulness of Friedman's test is wine tasting[2]: *"N wine judges each rate K different wines. Are any of the K wines ranked consistently higher or lower than the others?"*. This test yielded a p-value of 0.000475, indicating that there are attributes rated consistently higher than others. To further investigate this, we employ the posthoc Friedman-Nemenyi test. For every possible pair of attributes, this test verifies whether it is possible to reject the null hypothesis that the two attributes are rated similarly by the respondents. This test reveals that Communication is rated higher than seven out of the 11 other attributes, after applying the Bonferroni correction[3], as shown in Table IV. Besides, Quality Assurance, the attribute with the second-highest value of $\%Essential$, is rated consistently higher than Availability (p-value = 0.001602).

VI. LIMITATIONS

A threat to the results' validity in qualitative work relates to the data classification subjectivity. To alleviate this threat, we used multiple methods. More than one researcher conducted every qualitative analysis step; when in doubt, other researchers joined the discussion. These discussions along the process aimed to validate the interpretations through mutual agreement. We also used the constant comparison technique when coding the interview transcripts, comparing our findings with previous ones as they emerged from the data analysis.

Another potential threat in qualitative works refers to the saturation or the comprehensiveness of the results. To ensure that we reached a comprehensive set of attributes that represent

[2]https://en.wikipedia.org/wiki/Friedman_test

[3]The target significance level is 0.004545 = 0.05/11 since each attribute is compared to 11 others.

great maintainers, we interviewed 33 well-experienced OSS maintainers, who are diverse in terms of location, gender, years of experience, and domain. We kept interviewing participants until we could not find any new attributes for five consecutive interviews. Still, the number of participants seems to be adequate to understand the core attributes in any cultural domain or study of lived experience [5].

Finally, most of our interviewees work on infrastructure OSS (e.g., operating systems, browsers, programming languages). Our findings might not generalize to maintainers working on non-critical environments (e.g., trivial JavaScript libraries [1]). To reduce this threat, we made sure we interviewed maintainers of smaller projects. Yet, while conducting our interviews, we could identify several maintainers responsible for multiple projects, including smaller/less visible ones. During our interviews, we asked them to report their experience in whatever projects in which they were involved.

VII. Related Work

Software engineering researchers investigated the attributes that would make great software developers, maintainers, and software project managers. In 1995, Turley and Bieman [34] reported 38 competencies that characterize the skills required by software developers. They show that, in addition to the technical skills, great software developers have interpersonal and personality traces that lead them to better performance. Li et al. [18] uncovered a set of 54 attributes of great engineers. They found that software engineers excellence comes with "writing good code", but also relates to looking for future value and costs, practicing informed decision-making, and continuous learning.

With a specific focus on managers at Microsoft, Kalliamvakou et al. [13] they found that, although technical skills are relevant, they are not the sign of greatness for an engineering manager. The main attributes relate to keeping a positive environment and enabling talent growth and software engineering autonomy. Once again, the interpersonal attributes stood up as essential characteristics. However, it is possible to notice that OSS maintainers' attributes spam across community building and sustainability, which transcends the requirements in companies. Likewise, our results show that great maintainers need to have a set of attributes that go beyond technical skills like software engineers. We expand this by showing that maintainers need to have a special "Swiss Army Knife" to deal with specific points to OSS, including community building and project sustainability.

More generally speaking, research about OSS spams from what motivates contributors [35], what attracts and hinders newcomers [28, 2], how to become a core contributor [39, 20], challenges faced by mentors [2], and career pathways [33].These efforts contribute to understanding the activities of OSS contributors. However, they hardly touch the maintainers' work, who, as we showed, are responsible for managing not only people and projects as in traditional software projects but also to assure that the community is well-kept and that the project is sustainable.

Closely related to this work, Wang et al. [36] investigated the activities performed by *Elite developers*—those who "hold clearly defined project management privileges in a project." By analyzing repository data, they found that these developers manage the community tasks, parallel coordinate work, and discuss them with multiple stakeholders. Although the authors uncover a set of tasks performed by these elite developers, it is not clear whether they are maintainers and the attributes that would make these developers perform their work with excellence. Therefore, we understand that our work brings a complementary perspective to Wang et al.'s [36].

Still, previous work from Gousious and colleagues focused on understanding the perspective of contributors [10] and integrators [11] about the pull-based development model [9]. Gousios *et al.* [11] found that integrators focus on quality assurance when reviewing a contribution and that prioritization of the contributions involves a sense of urgency and the size of the contribution. The social aspects were mentioned as challenges of dealing with pull requests. Our work complements Gousios' work by focusing not only on integrators but showing the characteristics that great maintainers would have. We can see that, and while quality assurance ans social aspects are a common themes, we show that the set characteristics of a great maintainer is broader and includes attributes that are not solely related to code integration.

Finally, Nadia Eghbal [8] analyzes OSS maintenance and presents some of the "hidden costs" of maintaining OSS projects (e.g., physical infrastructure, user support, and community management). She mentions that maintainers are responsible for "being mindful of how they allocate their attention," given that their effort is a limited resource. Eghbal's discussion revolves around the Management and Social categories uncovered here, which show that maintainers need to manage multiple aspects of the project, mindfully, to keep contributors, community, and project working in the same frequency.

VIII. Conclusion

Maintainers perform an unquestionably central work to the success and long-term sustainability of communities. The responsibilities of these contributors encompass a diverse set of responsibilities, which require different competencies. While one can think of a maintainer as a solely technical developer responsible for keeping high-quality code, our results showed that a great maintainer has to master a diverse set of skills.

In fact, we found that Communication was considered the most important attribute of a great maintainer with *Quality Assurance* being the second-ranked according to the $\%Essential$ metric. This has been confirmed by a statistical analysis that involved pairwise comparisons between the ratings received by the attributes. The other three attributes that compose the top-5 are not related to technical competencies (Community Building, Empathy, and Vision). Therefore, it is clear that, although maintainers are usually highly technically-skilled contributors, social and management skills are essential to excel in the position.

Acknowledgments

We thank the reviewers for their helpful comments and the participants of our study for their input. This work is partially supported by CNPq (grants 313067/2020-1, 309032/2019-9 and 304220/2017-5), National Science Foundation (grant 1900903), FACEPE/Brazil (APQ-0839-1.03/14), and INES 2.0 (FACEPE grants PRONEX APQ 0388-1.03/14 and APQ-0399-1.03/17, and CNPq grant 465614/2014-0).

References

[1] R. Abdalkareem, O. Nourry, S. Wehaibi, S. Mujahid, and E. Shihab. Why do developers use trivial packages? an empirical case study on npm. In E. Bodden, W. Schäfer, A. van Deursen, and A. Zisman, editors, *11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017*, pages 385–395. ACM, 2017.

[2] S. Balali, I. Steinmacher, U. Annamalai, A. Sarma, and M. A. Gerosa. Newcomers' barriers... is that all? an analysis of mentors' and newcomers' barriers in oss projects. *Computer Supported Cooperative Work (CSCW)*, 27(3):679–714, 2018.

[3] S. Baltes and S. Diehl. Worse than spam: Issues in sampling software developers. In *10th International Symposium on Empirical Software Engineering and Measurement*, ESEM '16, New York, NY, USA, 2016. ACM.

[4] A. Begel and T. Zimmermann. Analyze this! 145 questions for data scientists in software engineering. In *36th International Conference on Software Engineering*, ICSE 2014, page 12–23, New York, NY, USA, 2014. ACM.

[5] H. R. Bernard. *Research methods in anthropology: Qualitative and quantitative approaches*. Rowman & Littlefield, 2017.

[6] M. Buckingham. What great managers do. *IEEE Engineering Management Review*, 33(2):3–10, 2005.

[7] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, and F. Ferrucci. Gender diversity and women in software teams: how do they affect community smells? In *41st International Conference on Software Engineering: Software Engineering in Society, ICSE 2019*, pages 11–20, 2019.

[8] N. Eghbal. *Working in Public: The Making and Maintenance of Open Source Software*. Stripe Press, 2020.

[9] G. Gousios, M. Pinzger, and A. v. Deursen. An exploratory study of the pull-based software development model. In *36th International Conference on Software Engineering*, ICSE, pages 345–355. ACM, 2014.

[10] G. Gousios, M.-A. Storey, and A. Bacchelli. Work practices and challenges in pull-based development: The contributor's perspective. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 285–296. ACM, 2016.

[11] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen. Work practices and challenges in pull-based development: The integrator's perspective. In *37th International Conference on Software Engineering*, ICSE, pages 358–368, 2015.

[12] A. Hemon, B. Lyonnet, F. Rowe, and B. Fitzgerald. From agile to devops: Smart skills and collaborations. *Information Systems Frontiers*, 22(4):927–945, 2020.

[13] E. Kalliamvakou, C. Bird, T. Zimmermann, A. Begel, R. DeLine, and D. M. Germán. What makes a great manager of software engineers? *IEEE Trans. Software Eng.*, 45(1):87–106, 2019.

[14] E. Kalliamvakou, C. Bird, T. Zimmermann, A. Begel, R. DeLine, and D. M. German. What makes a great manager of software engineers? *IEEE Transactions on Software Engineering*, 45(1):87–106, 2019.

[15] N. Kano, N. Seraku, F. Takahashi, and S. Tsuji. Attractive quality and must-be quality. *Journal of the Japanese Society for Quality Control (in Japanese)*, 14(2):39–48, April 1984.

[16] B. A. Kitchenham and S. L. Pfleeger. Personal opinion surveys. In F. Shull, J. Singer, and D. I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 63–92. Springer, 2008.

[17] L. A. F. Leite, C. Rocha, F. Kon, D. S. Milojicic, and P. Meirelles. A survey of devops concepts and challenges. *ACM Comput. Surv.*, 52(6):127:1–127:35, 2020.

[18] P. L. Li, A. J. Ko, and A. Begel. What distinguishes great software engineers? *Empirical Software Engineering*, 25(1):322–352, 2020.

[19] P. L. Li, A. J. Ko, and J. Zhu. What makes a great software engineer? In *37th International Conference on Software Engineering, ICSE 2015*, pages 700–710, 2015.

[20] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye. Evolution patterns of open-source software systems and communities. In *International Workshop on Principles of Software Evolution*, pages 76–85. ACM, 2002.

[21] S. H. Padala, C. J. Mendez, L. F. Dias, I. Steinmacher, Z. S. Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. D. Simpson, M. Burnett, et al. How gender-biased tools shape newcomer experiences in oss projects. *IEEE Transactions on Software Engineering*, 2020.

[22] A. Rastogi, N. Nagappan, G. Gousios, and A. van der Hoek. Relationship between geographical location and evaluation of developer contributions in github. In *12th International Symposium on Empirical Software Engineering and Measurement*, ESEM '18, pages 22:1–22:8, New York, NY, USA, 2018. ACM.

[23] E. S. Raymond. *The cathedral and the bazaar - musings on Linux and Open Source by an accidental revolutionary*. O'Reilly, 1999.

[24] J. G. Rivera-Ibarra, J. Rodríguez-Jacobo, J. A. Fernández-Zepeda, and M. A. Serrano-Vargas. Competency framework for software engineers. In *2010 23rd IEEE Conference on Software Engineering Education and Training*, pages 33–40, 2010.

[25] C. B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Trans. Softw. Eng.*, 25(4):557–572, July 1999.

[26] E. Smith, R. Loftin, E. Murphy-Hill, C. Bird, and T. Zimmermann. Improving developer participation rates in surveys. In *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 89–92, 2013.

[27] D. Spencer. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.

[28] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles. Social barriers faced by newcomers placing their first contribution in open source software projects. In *18th ACM conference on Computer supported cooperative work & social computing*, pages 1379–1392, 2015.

[29] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa. Almost there: A study on quasi-contributors in open source software projects. In *40th International Conference on Software Engineering*, ICSE '18, page 256–266, New York, NY, USA, 2018. ACM.

[30] P. E. Strandberg. Ethical interviews in software engineering. In *13th International Symposium on Empirical Software Engineering and Measurement*, ESEM '19, 2019.

[31] A. L. Strauss and J. M. Corbin. *Basics of qualitative research : techniques and procedures for developing grounded theory*. Sage Publications, Thousand Oaks, 1998.

[32] M. Szumilas. Explaining odds ratios. *J Can Acad Child Adolesc Psychiatry*, 19(3):227–229, 2010.

[33] B. Trinkenreich, M. Guizani, I. Wiese, A. Sarma, and I. Steinmacher. Hidden figures: Roles and pathways of successful OSS contributors. *Computer Supported Cooperative Work (CSCW)*, 4(180), 2020.

[34] R. T. Turley and J. M. Bieman. Competencies of exceptional and nonexceptional software engineers. *Journal of Systems and Software*, 28(1):19 – 38, 1995.

[35] 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*, pages 649–676, 2012.

[36] Z. Wang, Y. Feng, Y. Wang, J. A. Jones, and D. Redmiles. Unveiling elite developers' activities in open source projects. *Transactions on Software Engineering and Methodology (TOSEM)*, 29(3):1–35, 2020.

[37] T. Whitaker. *What great teachers do differently: Seventeen things that matter most*. Eye on Education, 2012.

[38] M. Wood and F. Su. What makes an excellent lecturer? academics' perspectives on the discourse of 'teaching excellence'in higher education. *Teaching in higher education*, 22(4):451–466, 2017.

[39] M. Zhou and A. Mockus. What make long term contributors: Willingness and opportunity in oss community. In *34th International Conference on Software Engineering (ICSE)*, pages 518–528. IEEE, 2012.