Principled Evaluation of Strengths and Weaknesses in FLOSS Communities: A Systematic Mixed Methods Maturity Model Approach

Sandro Andrade, Filipe Saraiva

To cite this version:

Sandro Andrade, Filipe Saraiva. Principled Evaluation of Strengths and Weaknesses in FLOSS Communities: A Systematic Mixed Methods Maturity Model Approach. 13th IFIP International Conference on Open Source Systems (OSS), May 2017, Buenos Aires, Argentina. pp.34-46, 10.1007/978-3-319-57735-7_4 . hal-01776303

HAL Id: hal-01776303
https://hal.inria.fr/hal-01776303

Submitted on 24 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Principled Evaluation of Strengths and Weaknesses in FLOSS Communities: A Systematic Mixed Methods Maturity Model Approach

Sandro Andrade[1] and Filipe Saraiva[2]

[1] Federal Institute of Education, Science, and Technology of Bahia (IFBA)
GSORT Distributed Systems Group – Salvador – Bahia – Brazil
`sandroandrade@ifba.edu.br`
[2] Federal University of Pará (UFPA)
Institute of Exact and Natural Sciences – Belém – Pará – Brazil
`saraiva@ufpa.br`

Abstract. *Context:* Free and Open Source Software usually results from intricate socio-technical dynamics operating in a diverse and geographically dispersed community. Understanding the fundamental underpinnings of healthy and thriving communities is of paramount importance to evaluate existing efforts and identify improvement opportunities. *Objective:* This paper presents a novel reference model for evaluating the maturity of FLOSS communities by mixing quantitative and qualitative methods. *Method:* We build upon established guidelines for Design Science research in order to devise a well-informed and expressive maturity model, describing how those methods and procedures were used in the design and development of such a model. *Results:* We present the model structure and functions, as well as instructions on how to instantiate it as evaluations of FLOSS communities. The use of the proposed maturity model is demonstrated in four FLOSS communities. *Conclusion:* Whilst instantiating the model may be burdensome if aiming at sketchy evaluations, results indicate our model effectively captures the maturity regardless aspects such as community size and lifetime.

Keywords: FLOSS communities evaluation, discourse communities, maturity models, design science research, mixed methods research.

1 Introduction

Free/Libre Open Source Software (FLOSS) has been recognized, over the past years, as a promising socio-technical approach to deliver high quality technology in spite of being usually developed by a diverse, often decentralized, and geographically distributed community [7,37]. Understanding the social dynamics, work practices, methods, and tools adopted by such communities has been of interest not only to the canonical software industry but also to researchers from distinct fields [12], such as Management Science [20], Information Systems [10], Economics [6], and Social Sciences [17], just to mention a few.

Healthy and thriving FLOSS communities rely on effective communication [22], powerful socio-technical infrastructure [30], and refined hacker culture [23] to manage decentralization issues, attenuate contribution barriers, and delivery quality software in a timely way to a usually vast userbase. Capturing such refined socio-technical practices in systematic models for evaluating FLOSS communities enables a more rigorous and well-informed basis to compare distinct FLOSS efforts, reveal improvement opportunities for a given FLOSS community, and support researchers in figuring out the characteristics of the FLOSS community they are investigating. Nowadays, such evaluations are hampered by the lack of expressive and systematic maturity models particularly designed to take into account the subtleties of FLOSS development culture.

The scarcity of such maturity models hinders the thorough realization of many activities. First, when undertaking empirical observations of FLOSS communities, researchers may want to insulate confounding factors by ensuring that observed communities have similar maturity. Second, although some incipient FLOSS projects with low maturity may be seen as promising opportunities, FLOSS investors usually prefer to sponsor seasoned communities, where risks are lower and ROI is more likely. Third, it usually takes some time to young community managers start understanding what makes a thriving FLOSS community. Knowing beforehand the many facets that FLOSS community maturity encompasses, how to measure them, and possible improvement opportunities is quite useful to shorten the time required to reach enhanced maturity.

Existing software development maturity models, while effective in capturing general process areas and key practices, are usually inapt or limited in coping with and assessing idiosyncrasies commonly found in FLOSS projects [32,36]. Most of the available methods for evaluating FLOSS communities entail the use of quantitative approaches usually expressed as metrics for aspects such as community activeness, size, diversity, and performance [9,11,21,27,28,31]. Whilst useful to investigate how community performance has been evolving over time, metrics by themselves provide no guidance regarding improvement opportunities and may be tricky to be applied across different FLOSS communities.

The work presented in this paper has been driven by two major long-term research questions: RQ1) to which extent refined socio-technical practices of thriving FLOSS communities can be captured as a maturity model? and RQ2) is such a maturity model of any help when investigating the achievements of different FLOSS communities or identifying opportunities for enhanced maturity? We believe existing approaches are still ineffectual in addressing the particularities of FLOSS development, rigorously supporting the replication of evaluations, and providing a well-informed artifact for revealing improvement opportunities.

This paper presents a novel maturity model for the systematic evaluation of FLOSS communities, particularly designed to capture those – sometimes ingenious – socio-technical activities that induce healthiness, long-term sustainability, and quality of delivered technologies. The model is slightly influenced by CMMI [34] and its structure encompasses nine maturity domains (named *improvement areas*). Each improvement area, in its turn, brings together a set of socio-technical activities (named *disciplines*) expected to be found in thriving communities. The realization degree of a discipline is evaluated by its corresponding (quantitative or qualitative) *metric*. Finally, our model specifies six distinct *maturity levels* and which disciplines are required by each level. A FLOSS community is said to be on a given maturity level when it exhibits all level's disciplines, each one with a metric value that satisfies a particular condition (named *acceptance criteria*) for such a discipline in that particular maturity level.

The process of instantiating the maturity model as a new FLOSS community evaluation is described and, afterwards, demonstrated for four communities (KDE Plasma, Apache HTTP, Poppler, and Inkscape) with different sizes and lifetimes. Results show that the model is able to identify maturity in spite of such differences and that the adopted mixed methods (quantitative/qualitative) approach helps mitigating issues presented by existing evaluation techniques.

The remainder of this paper is organized as follows. Section 2 details the Design Science and mixed methods approaches that guided this research. Section 3 explains the notion of maturity of FLOSS communities, the problem of evaluating such communities, and the requirements for the proposed maturity model. Section 4 discusses related work, while Section 5 explains the design (structure and functions) of our model. In Section 6, we demonstrate how our maturity model was used to evaluate four distinct FLOSS communities. In Section 7, we discuss the strengths and weaknesses of our proposal and present venues for future research. Finally, Section 8 draws the concluding remarks.

2 Method

This work was carried out in accordance with the research framework presented by Johannesson and Perjons in [16] and supported by the guidelines for mixed methods research introduced by Creswell in [8].

Johannesson&Perjons' (JP) framework aims at creating an artifact and producing knowledge about how such an artifact helps sorting out a problem that recurrently appears in a given intended practice. Their framework defines five main activities (with associated input and output) and provides guidelines for carrying out such activities, selecting suitable research strategies, and relating the research to an existing knowledge base. The five activities are: *explicate problem* (described, for this work, in Section 3), *define requirements* (also in Section 3), *design and develop artifact* (Section 5), *demonstrate artifact* (Section 6), and *evaluate artifact* (not addressed in this paper and subject of future work).

The structure of the maturity model we propose herein entails a set of quantitative and qualitative metrics which evaluate different aspects of a given community. Therefore, using our model to create a new evaluation of a particular community can be seen as conducting a mixed methods research, where both quantitative and qualitative data collection and analysis techniques are simultaneously applied. We build on the guidelines provided by Creswell in [8] in order to define how the proposed model should be used to evaluate a community.

3 Problem and Requirements

In consonance with the JP framework, we carried out the *explicate problem* activity by undertaking two sub-activities: *define precisely* and *position and justify.*

Define precisely. Before defining the problem of assessing FLOSS communities maturity, it is crucial to explain what we mean by *maturity* in this work and its impact on community's healthiness. By maturity we mean the degree of accomplishment a FLOSS community exhibits in reaching the goals of delivering high quality technology and becoming an ecosystem which supports socio-technical advances for all of its constituent actors. We elaborate this concept of maturity around three major underpinnings: *i)* userbase management; *ii)* long-term sustainability; and *iii)* inner/outer communication.

Userbase management encloses all practices which ease the delivery of high quality technology to an ever increasing userbase. This includes, for instance, quality assurance, software localization, and the availability of user-friendly installers or binary packages. *Long-term sustainability* refers to those practices which try to keep the community's health in spite of disturbances in the underlying socio-technical infrastructure, such as contributors leaving the project, disruptive strategical changes, or internal conflicts. Finally, *inner/outer communication* includes all practices which enable the effective communication inside the community and between the community and other ecosystem actors. Such maturity underpinnings represent overarching views of maturity and are further detailed as a set of disciplines that make up the model we propose herein.

With that view of maturity in mind, the problem addressed in this work can be defined: how should one proceed to assess the maturity degree of a given FLOSS community? How to make such evaluations more systematic and replicable across different researchers? How to identify improvement opportunities in a given FLOSS community to enhance its maturity degree?

Position and justify. We believe that assessing FLOSS communities maturity is a significant and challenging problem of general interest. It may appear in quite diverse practices. For example, knowing about a community maturity is an important decision-making information for a FLOSS investor selecting a community to support. Community managers may benefit from such a maturity model by carrying out self-assessments and identifying improvement opportunities. Social sciences researchers doing ethnographic studies may want to deliberately choose a mature community or an incipient one depending on their research goals. FLOSS educators may select communities with established mentoring programs (and, therefore, with increased maturity) to push their students into initial contributions. End users or companies may want to assess maturity to decide about the adoption of competing FLOSS solutions. Software Engineering researchers may undertake maturity assessment as an early exploratory/descriptive study and then, afterwards, decide about future research.

The *define requirements* activity was carried out by undertaking the two subactivities proposed in JP framework: *outline artifact* and *elicit requirements.*

Outline artifact. The artifact proposed herein is a model which captures prescriptive knowledge about how to create new community evaluations.

Elicit requirements. On the basis of the aforementioned drawbacks of existing approaches, we defined the following requirements for the proposed maturity model: R1) *model generality*: the model must be effective in evaluating the maturity of FLOSS communities regardless their size and lifetime; R2) *model expressiveness*: the model must encompass socio-technical activities particularly relevant to FLOSS; R3) *model predictability*: the model must support, as much as possible, systematic and replicable evaluations.

4 Related Work

In [25], authors studied 40 successful and 40 unsuccessful FLOSS projects from SourceForge and analyzed their respective maturities by verifying the (lack of) adoption of some important processes/tools like mailing lists, version control systems, documentation, and more. The successful projects exhibited the adoption and continuous use of such processes, as opposed to those unsuccessful projects.

The work presented in [21] performs a study on maturity models for FLOSS communities in order to decide whether a software can be used for commercial solutions. In such a work, they assume there is a correlation between the maturity level of a FLOSS community and the products (software systems) delivered by this community: if the community has a high maturity level then the software created by them will exhibit a high maturity as well. Their approach is heavily based on CMMI [34], using the same levels and descriptions.

Sony Mobile has been developing studies on FLOSS maturity models for a similar context. Their model, described in [3], defines five levels of maturity for adoption of open source technologies: from "accidental" and "repetitive" – when FLOSS use is driven by individual initiatives, through "directed" – when FLOSS adoption has gained support from executive management, to "collaborate" and "prevail" – when a full-fledged FLOSS culture is already in place. This model initially supported the decision of Sony Mobile about the use of Android OS in their smart-phone products. Later, they have extended such model into a general tool for evaluation of FLOSS projects.

In [28], the authors propose an assessment model for FLOSS development processes named *Open Source Maturity Model.* Such a model resembles CMMI in many aspects and it is aimed to be used by both companies and FLOSS communities. In their work, a given FLOSS can be classified in one of three levels and the classification process is carried out by analyzing whether some elements of the management and development process are used by the community. For instance, they verify whether the community yields product documentation, the popularity of the software, contributions from companies, and more.

The QualOSS model, described in [32], evaluates qualitative aspects of software products and their communities in order to verify software quality and maintainability. In [15], a qualitative model based on metrics extracted from source code repositories, mailing lists, and issues tracking tools is proposed. QSOS (Qualification and Selection of Open Source Software) [2] is a maturity assessment methodology that aims to compare and select FLOSS projects to be used by companies. QSOS evaluations are carried out in four steps. In the "define" step, users must define different elements to be evaluated (e.g. features, license, and community). The "evaluate" step assigns scores to each one of such elements. In the "qualify" step, users apply filters/weights to scores in order to verify whether a project is good enough to be adopted. Finally, the "select" step verifies whether the software fulfill the needs defined by the user.

In spite of such a variety of models already available, we believe that some capabilities, not tackled by previous work, are important enough to justify the need for a new model. First, most of existing models provide no systematic instruments for measuring activities performance and, therefore, are highly dependent of a skilled evaluator to succeed. Second, social aspects are usually not addressed (except in People-CMM). Third, such models usually evaluate very specific subjects, such as product's quality or benefits of FLOSS adoption, being helpful only for those particular goals. Finally, mixing qualitative and quantitative metrics to improve model's accuracy is rarely adopted in existing work.

5 The Maturity Model

The sub-activities defined in JP framework for the *design and develop artifact* activity are: *imagine and brainstorm*, *assess and select*, *sketch and build*, and *justify and reflect.*

Table 1. Maturity model's Improvement Areas and corresponding Categories

Acronym	Improvement Area (IA)	Category
CA	Community Activeness	Inner/Outer Communication
FM	Financial Management	Long-Term Sustainability
OG	Open Governance	Inner/Outer Communication
QA	Process and Product Quality Assurance	Userbase Management
PR	Public Relations	Inner/Outer Communication
SI	Social Infrastructure	Long-Term Sustainability
SM	Strategic Management	Long-Term Sustainability
UR	User Reachability	Userbase Management

Imagine and brainstorm. In this work, the maturity model requirements and the activities it encompasses were elicited by using the participant observation [33] method. The authors of this paper have been contributing to communities such as Qt [35], KDE [18], and Mageia [24] for nearly ten years, working on activities that span coding, promotion, artwork, finances, community management, and strategic management. In order to elicit the socio-technical activities captured in our model, we carried out a brainstorm which resulted in an extensive enumeration of practices we think highly impacts community maturity.

Assess and select. After having a list of prospective socio-technical activities, we ranked each activity regarding its impact on community maturity and its likelihood of being generically carried out in any FLOSS community (global practice). We systematically discarded those ones with minor impact on maturity or those whose adoption makes sense only in particular scenarios. Whilst brainstorming is obviously amenable to research bias, we consider that the rank-discard operation alleviates some threats and makes it acceptable for this paper's purposes. The adoption of more rigorous techniques for selecting those socio-technical activities are subject of future work.

Sketch and build. We carried out this sub-activity in many iterations, where the model constructs, structure, and functions were incrementally refined as we gained knowledge about the proposed artifact. The model is slightly influenced by CMMI (in particular, by CMMI-Dev [34] and People-CMM [4] models), since it also defines levels of maturity and expected disciplines for each level. The following paragraphs explain the final model's structure and behavior, as well as the decisions we took in order to fulfill the requirements mentioned in Section 3.

The proposed maturity model is defined as a tuple $MM = \langle C, IA, D, L\rangle$; where $C = \{$*userbase management*, *long-term sustainability*, *inner/outer communication*$\}$ is a set of *categories* representing the three maturity underpinning presented in Section 3, IA is a set of *improvement areas*, D is a set of *disciplines*, and L is a set of *levels* of increasingly maturity. An improvement area $ia_i \in IA$ is defined as a tuple $ia_i = \langle a, n, c\rangle$; where a is the improvement area's *acronym*, n is the improvement area's *name*, and $c \in C$ is the improvement area's associated *category*. Table 1 presents the categories and improvement areas of our model.

A discipline $d_i \in D$ is defined as a tuple $d_i = \langle a, n, t, \mu, pv, ia\rangle$; where a is the discipline's *acronym*, n is the discipline's *name*, $t \in \{$*T–technical*, *S–social*, *S/T–socio-technical*$\}$ is the discipline's *type*, μ is the discipline's *metric*, pv is the discipline's *preferable value*, and ia is the discipline's associated *improvement area*. Each metric $d_i.\mu$ is defined in a quantitative (QT) or qualitative (QL) way. A preferable value $d_i.pv$ specify whether maturity increases with greater ($\uparrow$) or smaller ($\downarrow$) values, for quantitative metrics, or with which specific values (e.g. Yes/No), for qualitative metrics. Table 2 presents the model's disciplines.

Finally, a level $l_i \in L$ is defined as a tuple $l_i = \langle n, CP\rangle$; where n is the level's name and CP is the level's set of *compliance points*. A compliance point $cp_{ij} \in l_i.CP$, in its turn, is defined as a tuple $\langle rd, ac(rd.\mu)\rangle$; where $rd \in D$ is the compliance point's *requested discipline* and $ac(rd.\mu)$ is a corresponding predicate named *acceptance criteria* that, when evaluated as true, denotes that the discipline rd is carried out with a maturity degree that is good enough for the level l_i. The model's levels and disciplines are presented in Table 3.

The maturity of a given FLOSS community FC is defined as the name of the maximum level l_m that has all its compliance points satisfied in FC:

$$M(FC) = l_m.n;\text{ where } m = \max_{i:l_i \in L \wedge \forall cp \in l_i.CP:cp.ac(cp.rd.\mu)} i$$

Justify and reflect. In order to cope with the subtleties of our definition of maturity and leverage the fulfillment of requirement R2 (model expressiveness), we adopted a mixed methods approach where all quantitative and qualitative data collection/analysis steps are carried out simultaneously. A last analysis

Table 2. Some maturity model's Disciplines for each Improvement Area (IA)

IA	Acronym	Discipline	Type	Metric ($\mu_{Acronym}$)	Type
QA	QA1	Static Code Analysis	T	% codebase analyzed	QT (↑)
	QA2	Code Review	T	% codebase reviewed	QT (↑)
	QA3	Continuous Integration	T	% codebase under CI	QT (↑)
	QA4	Documentation Policy	T	qualitative evidence	QL (Yes)
	QA5	Release Schedule	T	qualitative evidence	QL (Yes)
CA	CA1	Mailing List Activity	S/T	norm. threads/month	QT (↑)
	CA2	Mailing List Resp. Rate	S/T	% threads answered	QT (↑)
	CA3	Bug Fixing	T	% bugs fixed/month	QT (↑)
	CA4	Activity Diversity	S/T	norm. Pony Factor	QT (↑)
OG	OG1	Public Repository	T	qualitative evidence	QL (Yes)
	OG2	Public Roadmap	T	qualitative evidence	QL (Yes)
	OG3	Open Release Process	T	qualitative evidence	QL (Yes)
	OG4	Open QA Process	T	qualitative evidence	QL (Yes)
UR	UR1	Localized Content	T	qualitative evidence	QL (Yes)
	UR2	Binary Packages	T	qualitative evidence	QL (Yes)
	UR3	Cross-Platf. Support	T	qualitative evidence	QL (Yes)
PR	PR1	Website	S	completeness degree	QT (↑)
	PR2	Release Announces	S	qualitative evidence	QL (Yes)
SI	SI1	Regular Sprints	S/T	#sprints/year	QT (↑)
	SI2	Newcomers Program	S/T	qualitative evidence	QL (Yes)
FM	FM1	NGO/Foundation	S	qualitative evidence	QL (Yes)
	FM2	Sponsorship Strategy	S/T	qualitative evidence	QL (Yes)
SM	SM1	Board of Directors	S/T	qualitative evidence	QL (Yes)
	SM2	Advisory Board	S/T	qualitative evidence	QL (Yes)

step converges the partial results and come up with an ultimate outcome. In our model, this is implemented by a set of acceptance criteria defined for each level. We address requirement R1 (model generality) by ensuring all metrics use normalized values, insulating the effect of community size/lifetime. For example, the normalized Pony Factor is defined as the canonical Pony Factor [26] divided by the total number of core developers. Most of metrics presented in Table 2 are straightforward and may be evaluated by mining data sources such as version control systems, mail archives, code review platforms, and bug tracking systems. The metric *completeness degree* assigns an integer value in $[0, 5]$ to the community's website, depending on how many information is made available.

6 Artifact Demonstration

As mentioned in Section 2, in this work we evaluated the proposed maturity model by demonstration. JP framework defines two sub-activities for the *demonstrate artifact* activity: *choose or design cases* and *apply artifact*.

Table 3. Maturity model's Levels with Required Disciplines and Acceptance Criteria

Level	**Disciplines**	**Acceptance Criteria**
1 (operational)	CA[1-3]	$\mu_{CA1} \geq 0.1 \wedge \mu_{CA2} \geq 0.5 \wedge \mu_{CA3} \geq 0.1$
2 (proactive)	QA2, OG1	$\mu_{QA2} \geq 0.1 \wedge \mu_{OG1} = Yes$
	PR[1-2]	$\mu_{PR1} \geq 2 \wedge \mu_{PR2} = Yes$
3 (established)	QA[2,4,5]	$\mu_{QA2} \geq 0.5 \wedge \mu_{QA4} = Yes \wedge \mu_{QA5} = Yes$
	CA[1-4]	$\mu_{CA1} \geq 0.5 \wedge \mu_{CA2} \geq 0.8 \wedge \mu_{CA3} \geq 0.6 \wedge \mu_{CA4} \geq 0.05$
	UR[1-2], PR[1-2]	$\mu_{UR1} = \mu_{UR2} = \mu_{PR2} = Yes \wedge \mu_{PR1} \geq 3$
4 (managed)	QA[1-3], SI2	$\mu_{QA1} \geq 0.8 \wedge \mu_{QA2} \geq 0.75 \wedge \mu_{QA3} \geq 0.5 \wedge \mu_{SI2} = Yes$
	OG[2], UR[3]	$\mu_{OG2} = \mu_{UR3} = Yes$
5 (sustainable)	OG[3-4], SM[1-2]	$\mu_{OG3} = \mu_{OG4} = \mu_{SM1} = \mu_{SM2} = Yes$
	FM[1-2]	$\mu_{FM1} = \mu_{FM2} = Yes$

Choose or design cases. We chose four representative FLOSS communities to demonstrate to use of the proposed maturity model: KDE Plasma [19], Apache HTTP [1], Poppler [29], and Inkscape [14]. Such communities were selected because they exhibit different characteristics regarding size and lifetime, which makes it possible a more thorough investigation on how successful our model is in addressing requirement R1 (model generality).

Apply artifact. All quantitative metrics defined for our maturity model's disciplines have been evaluated, in the aforementioned FLOSS communities, manually, by using tools such as `git_stats` [13] or by scripts developed as part of this work. We collected evidence for the qualitative metrics by observing documents such as project's website, wiki pages, and mailing list archives. Table 4 presents the evaluations of some maturity model's metrics.

Table 4. Some metric values for the four FLOSS communities evaluated

Metric	**KDE Plasma**	**Apache HTTP**	**Poppler**	**Inkscape**
μ_{QA1}	0.82	0.00	0.00	0.87
μ_{QA2}	0.93	0.96	0.12	0.77
μ_{QA3}	1.00	1.00	1.00	0.52
μ_{CA4}	0.19	0.16	0.83	0.21
μ_{OG3}	Yes	Yes	Yes	No
μ_{OG4}	Yes	No	Yes	No
μ_{PR1}	3	5	2	5
μ_{SI1}	2	1	0	1
μ_{FM1}	Yes (KDE e.V.)	Yes (ASF)	No	Yes (Inkscape)
μ_{FM3}	No	Yes	No	Yes

7 Discussion and Future Work

The maturity evaluation data presented in Table 4 allows one to classify the investigated FLOSS communities in the following maturity levels: KDE Plasma (4: managed), Apache HTTP (3: established), Poppler (2: proactive), and Inkscape (4: managed). In spite of having nice politics for code review ($\mu_{QA2} = 0.96$) and continuous integration ($\mu_{QA3} = 1.00$), Apache HTTP fails in reaching higher maturity mostly because of the lack of static code analysis ($\mu_{QA1} = 0.00$) – important to reduce bugs density. Poppler got a low maturity level because of the lack of quality assurance practices such as code review, documentation policy, and release schedule – probably a consequence of its small size (biggest normalized Pony Factor: $\mu_{CA4} = 0.83$). Finally, KDE Plasma and Inkscape fail in having increased maturity because of the lack of sponsorship strategy ($\mu_{FM3} = No$) and open release and QA processes ($\mu_{OG3} = \mu_{OG4} = No$), respectively.

At this point, some important considerations can be drawn about the maturity model's limitations when evaluating FLOSS communities and threats to the validity of the demonstration presented herein. First, the model's acceptance criteria have been defined to assign different fulfillment degrees of disciplines into increasingly levels of maturity. Obviously, different acceptance criteria may lead to different results and are subject of further investigation. Second, the lack of more systematic information about how to carry out the qualitative evaluations may also imply in different results when replicating evaluations. Third, exceptional scenarios such as lack of community activeness because of heavy codebase stabilization (and, therefore, high maturity) are not currently addressed.

As for future work, a number of improvement venues may be identified. Creating new evaluations may be burdensome, specially for seasoned communities with a lot of data to be analyzed. We plan to provide better support by creating a new tool or extending initiatives like Bitergia's Grimoire Lab [5]. Advanced empirical studies are also planned, aimed at providing fundamental knowledge for enhancing the capture of socio-technical activities, refining the acceptance criteria, and evaluating maturity model quality attributes.

8 Conclusion

This paper presented a systematic mixed methods approach for evaluating the maturity of FLOSS communities. The novelty of our approach is the definition of a maturity model which captures socio-technical practices endemic to FLOSS ecosystems and mixes quantitative and qualitative metrics to systematically support the assessment of maturity and identification of improvement opportunities. We described the model structure and behavior, and demonstrated how to use the proposed artifact to evaluate four FLOSS communities with distinct characteristics. We believe this work advances the field of open source research by providing a generic artifact which enables the systematic capture of refined FLOSS-related socio-technical activities and supports researchers in carrying out replicable FLOSS communities evaluation experiments.

References

1. Apache Foundation: The Apache HTTP Server Project (2017), http://httpd.apache.org
2. Atos: Qualification and Selection of Open Source software (QSOS) (2013), http://backend.qsos.org/download/qsos-2.0_en.pdf
3. Bergman, O.: Report from the SCALARE Project – Sony Mobile's Involvement in Open Source. Tech. Rep. 1, Sony Mobile (2014), http://scalare.org/sony-mobiles-involvement-in-open-source/
4. Bill Curtis, William E. Hefley, S.A.M.: People Capability Maturity Model (P-CMM), version 2.0, second edition. Tech. Rep. CMU/SEI-2009-TR-003, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2010), http://cmmiinstitute.com/resources/people-capability-maturity-model-p-cmm
5. Bitergia: Grimoire Lab - OpenSource Development Analytics toolkit (2017), http://grimoirelab.github.io/
6. Bitzer, J., Schröder, P.: The Economics of Open Source Software Development. Elsevier (2006), https://books.google.com.br/books?id=obex6Fkrc18C
7. Capiluppi, A., Lago, P., Morisio, M.: Characteristics of open source projects. In: 7th European Conference on Software Maintenance and Reengineering (CSMR 2003), 26-28 March 2003, Benevento, Italy, Proceedings. p. 317. IEEE Computer Society (2003), http://dx.doi.org/10.1109/CSMR.2003.1192440
8. Creswell, J.W.: Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. SAGE Publications, Inc, 4 edn. (3 2013)
9. Crowston, K., Howison, J.: Assessing the health of open source communities. IEEE Computer 39(5), 89–91 (2006), http://dx.doi.org/10.1109/MC.2006.152
10. Fitzgerald, B.: The transformation of open source software. MIS Quarterly 30(3), 587–598 (2006), http://misq.org/the-transformation-of-open-source-software.html
11. Franco-Bedoya, O.: Open source software ecosystems: Towards a modelling framework. In: Damiani, E., Frati, F., Riehle, D., Wasserman, A.I. (eds.) Open Source Systems: Adoption and Impact - 11th IFIP WG 2.13 International Conference, OSS 2015, Florence, Italy, May 16-17, 2015, Proceedings. IFIP Advances in Information and Communication Technology, vol. 451, pp. 171–179. Springer (2015), http://dx.doi.org/10.1007/978-3-319-17837-0_16
12. Gacek, C., Arief, B.: The many meanings of open source. IEEE Software 21(1), 34–40 (2004), http://dx.doi.org/10.1109/MS.2004.1259206
13. Gieniusz, T.: GitStats is a git repository statistics generator (2017), https://github.com/tomgi/git_stats
14. Inkscape Community: Inkscape – Draw Freely (2017), http://inkscape.org
15. Izquierdo-Cortazar, D., González-Barahona, J.M., Dueñas, S., Robles, G.: Towards automated quality models for software development communities: The QualOSS and FLOSSMetrics case. In: e Abreu, F.B., Faria, J.P., Machado, R.J. (eds.) Quality of Information and Communications Technology, 7th International Conference on the Quality of Information and Communications Technology, QUATIC 2010, Porto, Portugal, 29 September - 2 October, 2010, Proceedings. pp. 364–369. IEEE Computer Society (2010), http://dx.doi.org/10.1109/QUATIC.2010.66
16. Johannesson, P., Perjons, E.: An Introduction to Design Science. Springer (2014), http://dx.doi.org/10.1007/978-3-319-10632-8
17. Karanović, J.: Free software and the politics of sharing. Digital Anthropology p. 185 (2013)

18. KDE Community: KDE – Experience Freedom! (2017), http://www.kde.org
19. KDE Community: Plasma Desktop (2017), http://plasma-desktop.org
20. von Krogh, G., von Hippel, E.: The promise of research on open source software. Management Science 52(7), 975–983 (2006), http://dx.doi.org/10.1287/mnsc.1060.0560
21. Kuwata, Y., Takeda, K., Miura, H.: A study on maturity model of open source software community to estimate the quality of products. In: Jedrzejowicz, P., Jain, L.C., Howlett, R.J., Czarnowski, I. (eds.) 18th International Conference in Knowledge Based and Intelligent Information and Engineering Systems, KES 2014, Gdynia, Poland, 15-17 September 2014. Procedia Computer Science, vol. 35, pp. 1711–1717. Elsevier (2014), http://dx.doi.org/10.1016/j.procs.2014.08.264
22. Lanubile, F., Ebert, C., Prikladnicki, R., Vizcaíno, A.: Collaboration tools for global software engineering. IEEE Software 27(2), 52–55 (2010), http://dx.doi.org/10.1109/MS.2010.39
23. Lin, Y.: Hacker culture and the FLOSS innovation. IJOSSP 4(3), 26–37 (2012), http://dx.doi.org/10.4018/ijossp.2012070103
24. Mageia Community: Home of the Mageia project (2017), http://www.mageia.org
25. Michlmayr, M.: Software process maturity and the success of free software projects. In: Software Engineering: Evolution and Emerging Tech., pp. 3–14. IOS Press (2005), http://www.booksonline.iospress.nl/Content/View.aspx?piid=1139
26. Nalley, D.: The Tragedy of Open Source (2017), http://opensource.cioreview.com/cxoinsight/the-tragedy-of-open-source-nid-23375-cid-92.html
27. Ortega, F., González-Barahona, J.M.: Quantitative analysis of the wikipedia community of users. In: Désilets, A., Biddle, R. (eds.) Proceedings of the 2007 International Symposium on Wikis, 2007, Montreal, Quebec, Canada, October 21-25, 2007. pp. 75–86. ACM (2007), http://doi.acm.org/10.1145/1296951.1296960
28. Petrinja, E., Nambakam, R., Sillitti, A.: Introducing the opensource maturity model. In: Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. pp. 37–41. FLOSS '09, IEEE Computer Society, Washington, DC, USA (2009), http://dx.doi.org/10.1109/FLOSS.2009.5071358
29. Poppler Community: Poppler (2017), https://poppler.freedesktop.org
30. Scacchi, W.: Socio-technical interaction networks in free/open source software development processes. In: Software Process Modeling, pp. 1–27. Springer (2005)
31. Sigfridsson, A., Sheehan, A.: On qualitative methodologies and dispersed communities: Reflections on the process of investigating an open source community. Information & Software Technology 53(9), 981–993 (2011), http://dx.doi.org/10.1016/j.infsof.2011.01.012
32. Soto, M., Ciolkowski, M.: The QualOSS open source assessment model measuring the performance of open source communities. In: Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement, ESEM 2009, October 15-16, 2009, Lake Buena Vista, Florida, USA. pp. 498–501. ACM / IEEE Computer Society (2009), http://doi.acm.org/10.1145/1671248.1671316
33. Spradley, J.P.: Participant observation. Waveland Press (2016)
34. Team, C.P.: Cmmi for development, v1.3. Tech. Rep. CMU/SEI-2010-TR-033, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2010), http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661
35. The Qt Company: Qt — cross-platform software development for embedded & desktop (2017), http://www.qt.io

36. von Wangenheim, C.G., Hauck, J.C.R., von Wangenheim, A.: Enhancing open source software in alignment with CMMI-DEV. IEEE Software 26(2), 59–67 (2009), http://dx.doi.org/10.1109/MS.2009.34
37. Wilson, T.D.: Review of: Jodeph feller & brian fitzgerald. understanding open source software development. london: Addison-wesley, 2002. xi, 211 pp. ISBN 0-201-73496-6. Inf. Res. 7(2) (2002), http://informationr.net/ir/reviews/revs046.html