Measuring the Quality of Open Source Software Ecosystems using QuESo

Oscar Franco-Bedoya[1,2], David Ameller[1], Dolors Costal[1], and Xavier Franch[1]

[1] Group of Software and Service Engineering (GESSI)
Universitat Politècnica de Catalunya, Barcelona, Spain
[2] Universidad de Caldas, Manizales, Colombia
{ohernan,dameller,dolors,franch}@essi.upc.edu
http://www.essi.upc.edu/~gessi/

Abstract. Open source software has witnessed an exponential growth in the last two decades and it is playing an increasingly important role in many companies and organizations leading to the formation of open source software ecosystems. In this paper we present a quality model that will allow the evaluation of those ecosystems in terms of their relevant quality characteristics such as health or activeness. To design this quality model we started by analysing the quality measures found during the execution of a systematic literature review on open source software ecosystems and, then, we classified and reorganized the set of measures in order to build a solid quality model. Finally, we test the suitability of the constructed quality model using the GNOME ecosystem.

Keywords: Quality Model, Software Ecosystem, Quality Measures, Open Source Software

1 Introduction

Software ecosystems are emerging in the last years as a new way to understand the relationships between software projects, products, and organizations. There are two widespread definitions:

– A software ecosystem is *"a set of actors functioning as a unit and interacting with a shared market for software and services. A software ecosystem consists of actors such as independent software vendors (ISV), outsourcers, and customers. A software ecosystem typically is interconnected with institutions such as standardization organizations, open source software communities, research communities, and related ecosystems"* [1].
– A software ecosystem is *"a collection of software projects which are developed and evolve together in the same environment"* [2].

In the first definition software ecosystems are understood from a holistic business oriented perspective as a network of actors, organizations and companies, while the second definition focuses on technical and social aspects of a set of software projects and their communities. In this paper we try to reconcile both visions and consider the business oriented perspective together with the technical and social perspectives in order to assess software ecosystem quality in its broader sense.

We focus on a particular kind of software ecosystems, i.e., those that are built around an Open Source Software (OSS) initiative (e.g., Android, GNOME, and Eclipse ecosystems), namely OSS ecosystems. We have identified three dimensions of quality in OSS ecosystems: the first dimension is the quality of the software platform in which the projects of the ecosystem are built upon (e.g., the Android ecosystem provides the Android platform used by all the Android mobile apps); the second dimension, as mentioned Jansen and Cusumano [1], is the quality of the OSS communities that grow inside the ecosystem and ecosystem's projects (e.g., the *GNOME* community itself, i.e., the community of the platform, but also the communities of the projects that belong to the ecosystem such as Anjuta, Banshee, and Abi Word communities); the third dimension of quality is inherent to the ecosystems themselves, i.e., the quality derived from the understanding of the OSS ecosystem as a network of interrelated elements (e.g., the number of Eclipse plug-ins and their dependencies between them can be used to assess the ecosystem's interrelatedness).

Assessing the quality of OSS ecosystems is of vital importance because quality assurance is a way to prevent bad decisions, avoid problems, and it allows to verify the compliance with the requirements and the business goals. It can also be used for quality systematic monitoring to provide feedback and execute preventive actions. For example, before deciding to integrate a project into an established OSS ecosystem it is crucial to perform a good quality assessment to avoid problems such as inactive user communities, low level of community cohesion, or even synergetic evolution problems, i.e., lack of collaboration between the key developers.

One way to ensure that the quality assessment has covered the most important characteristics of the ecosystem is to use a quality model, *"the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality"* [3]. Unfortunately, currently there is not any quality model for OSS ecosystems available in the literature, except from some measures distributed among many papers.

To fill this gap, in this paper we present QuESo, a quality model for the quality assessment of OSS ecosystems. To obtain this quality model we used design science methodology [4], first, we searched in the literature for all available measures related to OSS ecosystems, second, we designed the quality model using both a bottom-up strategy by classifying the measures found, and a top-down strategy by analysing the relationships in the quality characteristics that can be assessed by the measures (e.g., to assess the community activeness we can count the number of changes in the source repository or the number of messages in the mailing lists in a recent period of time), and finally, we maded a preliminary solution evaluation.

The rest of the paper is structured as follows: in Section 2 we review the related work; in Section 3 we explain the research methodology; in Section 4 we explain the QuESo quality model; in Section 5 we provide examples of real measures and their meaning; in Section 6 we provide an initial validation of the model; in Section 7 we discuss some observations made in this work; and finally, in Section 8 we provide the conclusions and the future work.

2 Related Work

When talking about quality models in the software domain it is inevitable to mention the ISO quality model [5]. This quality model targets the quality of a software product, from three perspectives: internal, external, and quality of use. The specific quality characteristics of the ISO quality model do not cover the important dimensions of OSS ecosystems such as the ones related to the community or the ones related to the *health* of the ecosystem.

The QualOSS quality model [6] gives a good representation for one of the three dimensions covered by QuESo, the OSS community. However we had to extend it with new characteristics that are relevant in the context of OSS ecosystems (see Section 4.2).

As we will explain in Section 3, we have found many papers that, although do not provide a quality model, they propose a good set of measures to evaluate some aspects of OSS ecosystems. We would like to mention the works that provided the most interesting measures.

- Hartigh et al. [7] developed a concrete measure tool for evaluating business ecosystems based on the classification made by Iansiti and and Levien [8]. They conceptualized the business ecosystem health as financial health and network health based on a set of eight measures.
- Mens and Goeminne. [9] provided a set of measures (e.g., number of commits, total bugs, mailing list), by studying the developer community, including the way developers work, cooperate, communicate and share information.
- Neu et al. [10] presented a web application for ecosystem analysis by means of interactive visualizations. The authors used the application to analysis the *GNOME* ecosystem study case.
- Kilamo et al. [11] studied the problem of building open source communities for industrial software that was originally developed as closed source.

Finally we remark the existence of two other secondary studies about software ecosystems [12, 13], but in both cases the studies did not have a research question about quality metrics or quality assessment for software ecosystems. Also, it is worth mentioning that as a way to complete our SLR we included the results of these two studies to our SLR (see Section 3.1).

3 Research Methodology

We structure our research in terms of design science since it involves creating new artefacts and acquiring new knowledge. Our research methodology follows the engineering cycle as described by [4]. We have performed a first engineering cycle that includes steps for (1) problem investigation, (2) solution design and (3) a preliminary solution evaluation. In the rest of these sections we describe the strategies followed in each step.

3.1 Problem Investigation: Systematic Literature Review

Our problem investigation step was devoted to search in the literature for all available measures related to OSS ecosystems and establish criteria to judge wich measures to consider for developing QuESo. A Systematic Literature Review (SLR) is a method to identify, evaluate, and interpret the available research relevant to a particular topic, research question, or phenomenon of interest [14].

The literature review protocol is part of a wider SLR that we are conducting with the goal of identifying the primary studies related to OSS ecosystems. A more detailed explanation of the protocol can be found in Franco-Bedoya et al. [15].

The research question that addresses the measures and indicators related to the ecosystem quality is: *What measures or attributes are defined to assess or evaluate open source software ecosystems?*

We defined a search string based on keywords derived from all the SLR research questions:*"(OSS OR FOSS OR FLOSS OR Open Source OR Free Software OR Libre Software) AND ecosystem"*.

The search strategy used was a combination of sources: digital libraries, manual searches, the inclusion of specific papers from the two secondary studies mentioned in Section 2 and the chapters in a recently published book about software ecosystems [16].

As a result of the SLR, 53 primary studies were selected, from them we identified 17 related to the identification of measures to evaluate the quality of OSS ecosystems. Figure 1 illustrates the SLR selection process.

Once we had collected the measures from the selected papers, we used the following criteria from Hartigh et al. (2013) and Neu et al. [7] and [10] to include them in QuESo:

1. *User-friendly and operationalizable:* measures should be logical, easy to use and operationalizable into a measurable entity.
2. *Non-redundant:* when we identified similar measures we selected only one of them, but we kept all the sources for traceability.

After excluding non-operationalizable and merging the similar measures with the previous criteria, we finally selected 68 different measures for the QuESo quality model (note that some of the measures are used to assess more than one characteristic of the quality model).

3.2 Solution Design: Quality Model Construction

There exist several proposals for quality model construction that focus on software quality. Most of them follow top-down strategies [17, 18]. In short, they take as a basis a reference quality model such as the ISO quality model [5], take their quality characteristics as departing point and refine them till they end up with a hierarchy with specific measures at its lower level. Remarkably, the proposal in Radulovic and Garcia-Castro [19] is mainly bottom-up oriented, i.e., it takes a set of measures as departing point to build the model. For our purposes, a bottom-up approach is the most adequate because: (1) a well-established reference quality model (or even, in its defect, a complete and systematic body of knowledge) for software ecosystems is still missing [20], and (2) there already exist a myriad of specific measures that can be applied to OSS ecosystems and that have been identified in our SLR. Furthermore, although it focuses on the construction of software quality models, we can easily use it to the construction of a quality model for OSS ecosystems.

Radulovic and Garcia-Castro [19] proposal has a clearly defined sequence of steps:

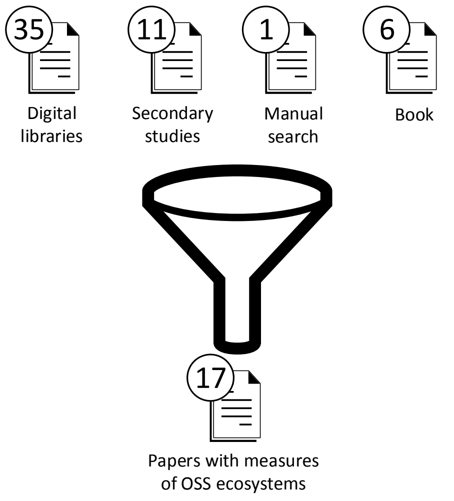

Fig. 1: Selection of primary studies.

1. To identify basic measures.
2. To identify derived measures.
3. To identify quality measures (by aggregation of basic and derived measures).
4. To identify relationships between measures.
5. To specify relationships between measures.
6. To define domain-specific quality sub-characteristics.
7. To align quality sub-characteristics with a quality model.

Note that the alignment in the seventh step partly implies top-down reasoning. Quality sub-characteristics that have been previously defined are related to others already specified in the existing model. If needed, some new quality sub-characteristics can be specified, or existing ones can be modified or excluded.

We have followed all the steps of the proposal. In particular, for steps 1 and 2, devoted to identify measures, we have based our work on the SLR described in Section 3.1. The application of step 7 requires the use of a reference quality model. Since, to our knowledge, a quality model for the whole scope of OSS ecosystems is still missing, we have decided to use QualOSS [6] which measures the performance of open source communities. Clearly, new quality sub-characteristics emerging from measures related to the ecosystem considered as a whole will have to be specified, since they are not addressed by QualOSS.

3.3 Solution Evaluation

As a first step for the solution evaluation we have validated the feasibility of obtaining the measures included in our QuESo quality model and, consequently, the feasibility of evaluating the characteristics and subcharacteristics proposed in QuESo. We have done this by taking the ecosystem around GNOME projects as a case study and have analyzed the literature related to the GNOME ecosystem to extract for which measures there are available values documented in the literature. The rest of details of the evaluation process are described in Section 6.

4 QuESo Quality Model

In this section we describe the QuESo quality model obtained as a result of the application of the procedure described in Section 3.2. The model is composed of two types of interrelated elements: quality characteristics and measures. Quality characteristics correspond to the attributes of an open source software ecosystem that are considered relevant for evaluation. The quality characteristics are organized in a hierarchy of levels that is described in the rest of this section. For the lack of space, in the tables presented in this paper we have omitted the descriptions. The whole set of measures with their definitions is available in the Appendix A. Also, note that we opted to keep the measure names that appear in the primary studies, even that in some cases the name given is not the most appropriate, we discuss about this topic in Section 7.

The quality characteristics in QuESo have been organized in three dimensions: (1) those that relate to the *platform* around which the ecosystem is built, (2) those that relate to the *community* (or set of communities) of the ecosystem and (3) those that are related to the *ecosystem as a network* of elements, such as projects or companies (see Figure 2).

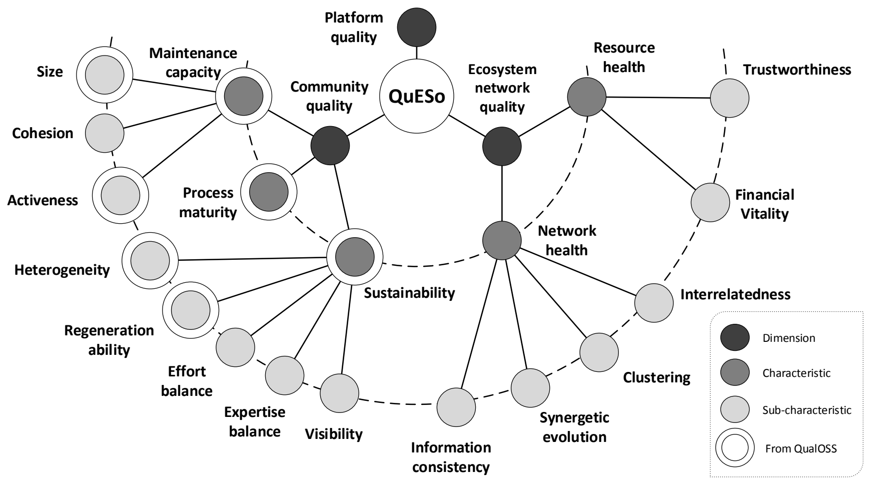

Fig. 2: QuESo quality model.

4.1 Platform-related Quality Characteristics

Platform-related quality characteristics consist of the set of attributes that are relevant for the evaluation of the software platform.

As a result of our SLR, we have observed that the literature do not provide measures for evaluating open source platform-related quality characteristics. This fact may indicate that there are not significant differential issues for open source software quality with respect to generic software quality that motivates the need of specific measures.

Then, similarly as done in the QualOSS model, since a mature proposal such as ISO 25000-SQuaRE [5] focuses on generic software quality, QuESo adopts directly the characteristics and sub-characteristics proposed by ISO 25000-SQuaRE and this part of the quality model is omitted in the paper.

4.2 Community-related Quality Characteristics

Following the procedure described in Section 3.2, the QuESo proposal for community-related quality characteristics is based both on the set of measures identified in our SLR and on the QualOSS quality model [6] (see Figure 2).

QualOSS specifies three community characteristics, namely, *maintenance capacity*, *sustainability* and *process maturity*.

Maintenance Capacity Soto et al. define *maintenance capacity* as the ability of a community to provide the resources necessary for maintaining its products and mention that aspects relevant to it are the number of contributors to a project and the amount of time they contribute to the development effort. In order to align maintenance capacity with our identified measures it is refined in three sub-characteristics: *size, cohesion* and *activeness*. The *size* of the community influences its maintenance capacity and can be evaluated by measures such as *number of core developers* and *number of committers*. The ability of the community to collaborate defined by its *cohesion* is also relevant. A measure that can be used to evaluate cohesion is the *ecosystem connectedness* in the community social network. Finally, the *activeness* of the community can be evaluated by measures such as *bug tracking activity* and *number of commits*. We have identified 26 measures that can be used to measure the maintenance capacity (see Table 1).

Sustainability *Sustainability* is the likelihood that a community remains able to maintain the products it develops over an extended period of time. According to Soto et al. it is affected by *heterogeneity* and *regeneration ability* and, as a result of our measure analysis, we have specified additional sub-characteristics besides them: *effort balance, expertise balance* and *visibility*.

The *heterogeneity* of a community contributes to its sustainability. For instance, if a community is mainly composed of employees of a particular company, there is the risk of the company cutting its financial support. *Heterogeneity* can be evaluated by measures such as *geographical distribution* of community members.

Regeneration ability also enhances sustainability since a community that has been able to grow in the past increases its chances of not declining in the future. A measure that we have identified for it is for instance, *new members* which counts the number of new members of the community at any point of time.

The *effort balance* is relevant for sustainability i.e., if most of the contribution effort comes from one or a small number of members of the community and it is not uniformly

Table 1: List of measures for *maintenance capacity*.

Subcharacteristic	Measure
Size	Number of contributors
Size	Number of members
Size	Number of authors
Size	Number of bug fixers
Size	Number of committers
Size	Number of core developers
Size	Number of nodes and edges
Cohesion	Betweenness centrality
Cohesion	Cluster of collaborating developers
Cohesion	Ecosystem connectedness
Cohesion	Outdegree of keystone actors
Activeness	Bug tracking activity
Activeness	Buildup of assets
Activeness	Community effort
Activeness	Date of last commit
Activeness	Files changed
Activeness	Files per version
Activeness	Lines added
Activeness	Lines changed
Activeness	Lines removed
Activeness	Mailing list
Activeness	Number of commits
Activeness	Contributor commit rate
Activeness	Developer activity diagrams
Activeness	Temporal community effort
Activeness	Number of event people

distributed, then its continuity is highly dependent on that small set of members. On the other hand, a balanced effort distribution among all members facilitates its continuity over time. Some measures for effort balance are: *number of developer projects* and *number of developer releases*.

In a similar way, the *expertise balance* among most members of a community is again a way to guarantee its sustainability. A community highly dependent on the expertise of one or a few members suffers from a risky situation. A measure for this is, for instance, *expertise view contributor* which calculates a contributor expertise based on the number and type of files he changed within a month.

The *visibility* of a community gives it the capacity of attracting people to contribute and support it if needed. Examples of measures identified for visibility are: *number of downloads, social media hits* and *web page requests*.

QuESo has 28 measures that can be used to measure the sustainability (see Table 2).

Process Maturity *Process maturity* is the ability of a developer community to consistently achieve development-related goals by following established processes. It can

Table 2: List of measures for *sustainability*.

Subcharacteristic	Measure
Heterogeneity	Geographical distribution
Regeneration ability	Temporal community effort
Regeneration ability	New members
Effort balance	Contributor commit rate
Effort balance	Developer activity diagrams
Effort balance	Maximum number of commits of a developer
Effort balance	Member effort
Effort balance	Member activity rate
Effort balance	Number of activity communities
Effort balance	Number of developer releases
Effort balance	Number of developer projects
Effort balance	Project developer experience
Effort balance	Temporal community effort
Effort balance	Total effort of members
Expertise balance	Expertise view contributor
Expertise balance	Principal member activity
Expertise balance	Relation between categorical event and developer participation
Visibility	Number of event people
Visibility	Amount of inquires or feature requests
Visibility	Job advertisements
Visibility	Number of downloads
Visibility	Number of mailing list users
Visibility	Number of passive user
Visibility	Number of readers
Visibility	Number of scientific publications
Visibility	Social media hits
Visibility	Visibility
Visibility	Web page requests

be assessed for specific software development tasks with the answers of questions such as: *is there a documented process for the task?* [6]. Apparently, this characteristic requires qualitative assessment more than quantitative measures. This is consistent with the results of our SLR since we have not identified measures devoted to evaluate process maturity aspects. The absence of measures for process maturity hampers the application of the bottom-up process to further refine this characteristic.

4.3 Ecosystem Network Quality Characteristics

Since QualOSS does not address the network-related quality, this part of QuESo is exclusively based on the analysis of measures identified in our SLR.

QuESo proposes two ecosystem network-related characteristics: *resource health* and *network health*. In this paper we take as definition for *health* applied to software ecosystems: *longevity and a propensity for growth* [21, 22].

Resource Health *Resource health* facilitates the combination of value activities from multiple actors to obtain value-creating end products [23]. It is related to the financial health concept defined by Hartigh et al. [7]: *"The financial health is a long-term financially based reflection of a partner's strength of management and of its competences to exploit opportunities that arise within the ecosystem and is directly related to the capability of an ecosystem to face and survive disruptions"*. In the OSS ecosystem case this means that there is a set of partners or actors functioning as a unit and interacting among them. Their relationships are frequently operated through the exchange of information and resources. Two sub-characteristics, particularly relevant to resource health, are the *financial vitality* and the *trustworthiness* of the ecosystem.

The *financial vitality* is the viability and the ability to expand (i.e., robustness, ability to increase size and strength) of the ecosystem [24]. Two examples of financial measures that evaluate it are *liquidity* and *solvency*. They can be obtained directly, e.g., using balance sheet data of partners, but also indirectly, through the network relations.

Trustworthiness is the ability to establish a trusted partnership of shared responsibility in building an overall open source ecosystem [25]. Operational financial measures obtained from bankruptcy models (e.g., *Z-score* and *Zeta model*) are adequate to measure it because they take short-term and long-term survival into account [7].

QuESo has 5 measures that can be used to measure the resource health (see Table 3).

Table 3: List of measures for *resource health*.

Subcharacteristic	Measure
Trustworthiness	Zeta model
Trustworthiness	Z-score
Financial vitality	Liquidity
Financial vitality	Solvency
Financial vitality	Network resources

Network Health [7] define *network health* as a representation of how well partners are connected in the ecosystem and the impact that each partner has in its local network. Healthy ecosystems show many relations and subsystems of different types of elements that are intensely related [26]. Furthermore, in a healthy OSS ecosystem network, these relations are mutualistic [27]. Van der Linden et al. [28] proposed to evaluate the network health of an OSS ecosystem before its adoption. To align network health with the identified measures we have refined it into four sub-characteristics: *interrelatedness*, *clustering*, *synergetic evolution* and *information consistency*.

Interrelatedness is the ability of the nodes of an OSS ecosystem to establish connections between them. It can be evaluated by measures such as *centrality* i.e., the number of network relations of a node, and *project activity diagrams* that allows to obtain the kind of project evolution.

Clustering is the capacity of the species (or nodes) in the entire ecosystem to be classified around its projects. It also enables small OSS projects to come together as a large social network with a critical mass [29]. Basic measures as *number community projects*, *number of files* and *variety in products* can be used to identify clusters using social network analysis techniques [30].

Synergetic evolution is the ability of the subsystems that constitute the whole ecosystem to form a dynamic and stable space-time structure [31, 24]. Measures such as *ecosystem entropy* and *ecosystem reciprocity* can be used to evaluate synergetic evolution. The *ecosystem entropy* measure is based on the definition of software system entropy from Jacobson [32] who states that it is a measure for the disorder that always increases when a system is modified. *Ecosystem reciprocity* measures direct and active collaboration between the company and its customers in creating value propositions (e.g., through collaboration with key developers in an OSS community and other companies within the ecosystem) [33].

Information consistency is the consistency of the core information elements across the breadth of an ecosystem. The *code vocabulary map* measure evaluates this sub-characteristic. It consists of a summary of terms used in the source code that can be used to obtain a general overview of the domain language of the project's network.

QuESo has 15 measures that can measure the network health (see Table 4).

5 Examples of Measures

In this section we provide several examples extracted from the papers selected in the SLR. In particular we have selected the examples that belong to the *GNOME* software ecosystem. Our intention is to clarify the type of measures that are mentioned in this paper with examples.

In the following we present the selected *GNOME* examples of measure values organized by the characteristics of the QuESo quality model. We omit *process maturity* because we have not found quantitative measures to evaluate it (see explanation in Section 4.2). We also omit *resource health* measures because examples for them are not reported in the SLR papers for the *GNOME* ecosystem.

- The *maintenance capacity* can be evaluated from the *number of authors* measure which gives the amount of people that change files in a project. According to Goeminne and Mens [34] data, for the *GNOME* ecosystem there have been 3.500 different people having contributed at least once to at least one of the *GNOME* projects between 1997 and 2012. The *number of commits* measure is also relevant. Each commit corresponds to the action of making a set of changes permanent. According to Jergensen and Sarma [35] Jergensen and Sarma (2011) approximately 480.000 commits were made in *GNOME* from 1997 to 2007.
- A measure for *sustainability* is the *member activity rate* which gives a value between 0 and 1 that helps to analyse the effort balance, i.e., a zero value indicates a uniform distribution of the work, which means that each person has the same activity rate while a value of 1 means that a single person carries out all the work. The *member activity rate* for the *GNOME Evince* project has had a value between 0,7 and 0,8 from 1999 to 2009 according to Mens and Goeminne [34].
- The *network health* of an ecosystem can be evaluated by measures such as *number community projects* and *number of active projects*. For the *GNOME* ecosystem,

Table 4: List of measures for *network health*.

Subcharacteristic	Measure
Interrelatedness	Contributor activity graph
Interrelatedness	Project activity diagrams
Interrelatedness	Networks node connection
Interrelatedness	Ecosystem connectedness
Interrelatedness	Ecosystem cohesion
Interrelatedness	Centrality
Interrelatedness	Variety of partners
Clustering	Variety in products
Clustering	Number community projects
Clustering	Number of active projects
Clustering	Number of files
Synergetic evolution	Distribution over the species
Synergetic evolution	Ecosystem entropy
Synergetic evolution	Ecosystem reciprocity
Information consistency	Code vocabulary map

there were more than 1.300 projects between 1997 and 2012 and more than 25% of them had been active for more than six and a half years. At the lower side of the spectrum, more than 25% of all projects had been active less than one year [36]. Another measure for network health is the *contributor activity graph*. According to Neu et al. [10], one of the contributors of the *GNOME* ecosystem has been working in 499 projects and has more than 15.000 changes between 1998 and 2011.

6 Validation: GNOME Case

In this section we present an early version of our quality model validation. The goal of this validation is to provide evidence of the feasibility to obtain the measures, and consequently, the feasibility to evaluate the corresponding characteristics and sub characteristics proposed in QuESo. We hope to validate this feasibility using the QuESo measures identified in the literature related to the *GNOME* ecosystem

6.1 Quality Model Validation

The validation of a quality model is very important and very difficult activity [37]. It is not practically possible to specify or measure all subcharacteristics for all parts of a OSS ecosystem. It also requires long period of time. Similarly it is not usually practical to specify or measure quality in use for all possible stakeholders scenarios [5]. The model should be tailored before use to identify those characteristics and subcharacteristics that are most important, and the different types of measure depending on the stakeholder goals and also to provide some evidence of the feasibility to obtain these measures, as mentioned in [21], one of the most habitual problems is the *absence of data* to calculate the measures. However, in order to gain confidence in the quality of the work, an initial feasibility and availability validation of the QuESo measures will be done.

6.2 GNOME Ecosystem Case

In this section we present an early version of our quality model validation. The goal of this validation is to provide evidence of the feasibility to obtain the measures, and consequently, the feasibility to evaluate the corresponding characteristics and sub characteristics proposed in QuESo. We hope to validate this feasibility using the QuESo measures identified in the literature related to the *GNOME* ecosystem.

We divided the process in two phases, similar to Samoladas [38]: the identification of the literature related to *GNOME* ecosystem and the specification of the QuESO measures that are available for the *GNOME* ecosystem.

In the first phase, we have identified several papers that have measures for analysing the *GNOME* ecosystem. In the second phase the selected works were analysed as follows: first the measures with available values were extracted, secondly the measures were classified according to the QuESo measure classification, and finally analysed the situation for each quality aspects of QuESo.

Figure 3 shows the QuESo graphical model with the percentage of measures found in the literature for each quality aspect and Table 5 shows the number of measures found for each subcharacteristisc. In the Appendix B we show a detailed table with the *GNOME* measures and the papers associated. The column source of this table lists the papers with values for each measure. The *community quality* dimension has measures associated with each subcharacteristic, and some of them have values for all the associated measures. However, in the case of the *ecosystem network quality* dimension we have not found values for the measures related to *resource health*, and only the 40% of measures of *network health* have values.

Table 5: List of measures by QuESo subcharacteristic

Sub-characteristic	Number of measures
Size	5
Cohesion	1
Activeness	13
Heterogeneity	1
Regeneration ability	2
Effort balance.	11
Expertice Balance	2
Visibility	1
Information consistency	0
Synergetic evolution	1
Clustering	2
Interrelatedness	3
Trustworthiness	0
Financial vitality	0

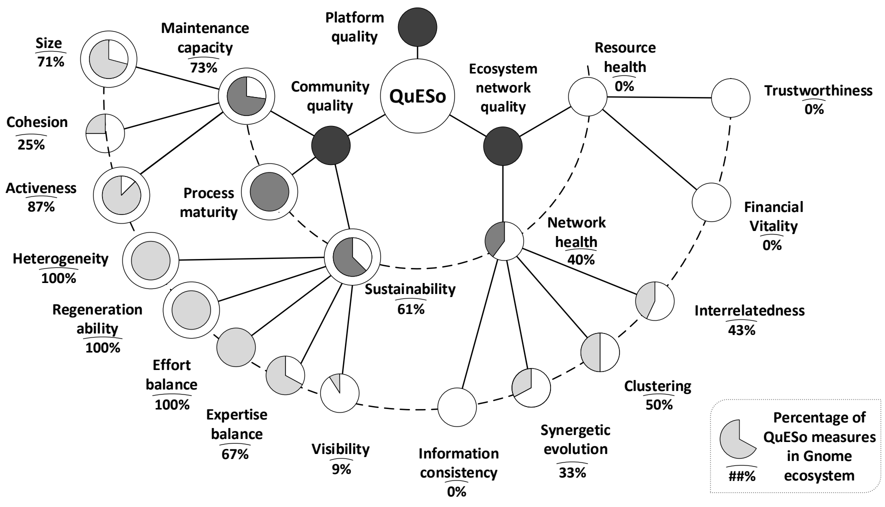

Fig. 3: *GNOME* ecosystem evaluation with QuESo

6.3 Observations

From the results shown in Figure 3 and Table 5 we made some observations:

– There are many works with measure values for the community while there are few works with measure values related to the ecosystem network.
– The papers found do not cover the whole set measures in the QuESo quality model.

The first observation is also mentioned by other authors, for example, Jansen et al. [21] wrote that there is little literature that studied OSS from an ecosystem perspective, while Manikas et al. [39] wrote that most of the works studied OSS from a project or community perspective.

For the second observation, we cannot state that there is full availability of measure values, however it worth to mention that in this case we limited the sources to the ones published in the literature. Other methods can include direct access to the *GNOME* data sources (e.g., the number of files, number of downloads, web page request, and code vocabulary map). Other measures such as z-score, liquidity, and solvency can be obtained using surveys or public data.

7 Discussion

Some observations were made during the design of this quality model. In the following, the most interesting ones are discussed:

– *Completeness:* since we followed a mainly bottom-up strategy, the completeness of the quality model depends on how complete the set of measures found in the literature is. In this sense, we would like to remark that our quality model may be not complete by one or more of the following reasons: there may be some papers with relevant measures not included in the SLR because they were not present in digital libraries or because our search string did not find them; another reason could be that some important measures are not yet reported in the literature. In this work, our intention was not to invent new measures but to organize the existing ones into a quality model.

- *Quantitative vs. qualitative:* the measures found in the literature are mostly quantitative, but a quality assessment may also include qualitative evaluations. For example, we commented in Section 4.2 the lack of measures for process maturity because in this case the assessment needs to be done with qualitative evaluations of the community. Since we have focused on quantitative measures, there may be other characteristics of the quality model that require or that may be complemented with qualitative evaluations.
- *Unbalanced distribution of measures:* just by looking into the measure tables, it is easy to observe that the amount of measures for some characteristics is high (e.g., *activeness* with 17 measures, *visibility* with 11 measures) while for other is very low (e.g., *heterogeneity* with 1 measure, *information consistency* with 1 measure). This unbalanced situation could be an indicator that more research is needed for the characteristics with a low amount of measures.
- *Measure names:* we have named the measures included in the QuESo quality model with the same names they are referred to in the SLR papers from where they were extracted. The reason for this is to improve traceability. However, some of those measure names might be ambiguous or misleading because it is not evident from them how the measure is evaluated (e.g., *project activity diagrams*). To improve measure understandability we have listed their definitions in the Appendix A.
- *Assesment process:* It is worth mentioning that to perform a complete quality assessment of a software ecosystem we first would need to define the assessment process which is out of the scope of this paper. The *quality assessment process* will have to deal with, e.g., How are the values of each measure interpreted (i.e., defining what are the good and the bad values)?; How can the measures be merged to provide the assessment for a particular sub-characteristic of the quality model?; or What are the principles to perform the assessment with missing, incorrect, and/or inconsistent measure data? We are will provide the answer to these and other questions as part of our future work in this topic.

8 Conclusions

In this paper we have presented QuESo, a quality model for assessing the quality of OSS ecosystems. This quality model has been constructed following a bottom-up strategy that consisted in searching the available measures for OSS ecosystems in the literature and then organize them into several quality characteristics. The presented quality model covers three aspects of OSS ecosystems: the platform, the community, and the ecosystem network; which altogether are a good representation of the most important aspects of an OSS ecosystem.

This quality model can be used as a starting point for the quality assessment of an OSS ecosystem, and it is in our plans for the future work to define a complete quality assessment process (as described in Section 5) and to apply it in a real quality assessment. As consequence new measures may be needed for the assessment, but this is the best way to improve, and complete the quality model, and a way to prove its capabilities in quality assessment.

ACKNOWLEDGMENTS

This work is a result of the RISCOSS project, funded by the EC 7th Framework Programme FP7/2007-2013 under the agreement number 318249. We would also like to thank Carme Quer for her assistance and the contribution of EOSSAC project, founded by the Ministry of Economy and Competitiveness of the Spanish government (TIN2013-44641-P) is also acknowledged.

References

1. Jansen, S., Cusumano, M.: Defining software ecosystems: a survey of software platforms and business network governance. [20] 13–28
2. Lungu, M., Malnati, J., Lanza, M.: Visualizing gnome with the small project observatory. In: Proceedings of the 6th MSR, IEEE (2009) 103–106
3. ISO/IEC 9126: Product quality – Part 1: Quality model (2001)
4. Wieringa, R.: Design science as nested problem solving. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, New York, NY, USA, ACM (2009) 8:1–8:12
5. ISO/IEC 25000: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE (2014)
6. Soto, M., Ciolkowski, M.: The QualOSS open source assessment model measuring performance of open source communities. In: Proceedings of the 3rd ESEM. (2009) 498–501
7. Hartigh, E., Visscher, W., Tol, M., Salas, A.J.: Measuring the health of a business ecosystem. [20] 221–245
8. Iansiti, M., Levien, R.: Keystones and dominators: framing operating and technology strategy in a business ecosystem. Technical report, Harvard Business School (2004)
9. Mens, T., Goeminne, M.: Analysing the Evolution of Social Aspects of Open Source Software Ecosystems. In: Proceedings of the 3rd IWSECO. (2011) 1–14
10. Neu, S., Lanza, M., Hattori, L., D'Ambros, M.: Telling stories about GNOME with Complicity. In: Proceedings of the 6th VISSOFT. (2011) 1–8
11. Kilamo, T., Hammouda, I., Mikkonen, T., Aaltonen, T.: From proprietary to open source - Growing an open source ecosystem. Journal of Systems and Software **85** (2012) 1467–1478
12. Manikas, K., Hansen, K.M.: Software ecosystems A systematic literature review. Journal of Systems and Software **86** (2013) 1294–1306
13. Barbosa, O., Alves, C.: A Systematic Mapping Study on Software Ecosystems. In: Proceedings of the 3rd IWSECO. (2011) 15–26
14. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3. Technical report, Keele University EBSE (2007)
15. Franco-Bedoya, O., Ameller, D., Costal, D., Franch, X.: Protocol for a systematic literature review on open source-software ecosystems. Technical report, Universitat Politcnica de Catalunya (2014) Available online: `www.essi.upc.edu/~gessi/papers/queso-slrprotocol.pdf`.

16. Jansen, S., van Capelleveen, G.: Quality review and approval methods for extensions in software ecosystems. [20] 187–217
17. Franch, X., Carvallo, J.P.: Using quality models in software package selection. IEEE Software **20** (2003) 34–41
18. Behkamal, B., Kahani, M., Akbari, M.K.: Customizing ISO 9126 quality model for evaluation of B2B applications. Information and Software Technology **51** (2009) 599–609
19. Radulovic, F., Garcia-Castro, R.: Extending Software Quality Models - A Sample In The Domain of Semantic Technologies. In: Proceedings of the 23rd SEKE. (2011) 25–30
20. Jansen, S., Brinkkemper, S., Michael Cusumano: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar Publishing (2013)
21. Jansen, S.: Measuring the health of open source software ecosystems: Beyond the scope of project health. Information and Software Technology (2014)
22. Lucassen, G., Rooij, K., Jansen, S.: Ecosystem Health of Cloud PaaS Providers. In: Proceedings of the 4th ICSOB. (2013) 183–194
23. Anderson, J.C., Narus, J.A., Narayandas, D.: Business Market Management: Understanding, Creating, and Delivering Value (3rd Edition). Prentice Hall (2009)
24. Li, X., Jie, X., Li, Q., Zhang, Q.: Research on the Evaluation of Business Ecosystem Health. In: Proceedings of the 6th ICMSEM, Springer (2013) 1009–1020
25. Agerfalk, P.J., Fitzgerald, B.: Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy. Mis Quartely **32** (2008) 385–409
26. Gamalielsson, J., Lundell, B., Lings, B.: The Nagios community: An extended quantitative analysis. In: Proceedings of the 6th OSS, Springer (2010) 85–96
27. Lundell, B., Forssten, B.: Exploring health within OSS ecosystems. In: Proceedings of the 1st OSCOMM. (2009) 1–5
28. van der Linden, F., Lundell, B., Marttiin, P.: Commodification of Industrial Software: A Case for Open Source. IEEE Software **26** (2009) 77–83
29. Scacchi, W.: Free/Open Source Software Development: Recent Research Results and Emerging Opportunities. In: Proceedings of the 6th ESEC-FSE. (2007) 459–468
30. Lungu, M., Lanza, M., Gîrba, T., Robbes, R.: The Small Project Observatory: Visualizing software ecosystems. Science of Computer Programming **75** (2010) 264–275
31. Haken, H.: Synergetics. Naturwissenschaften **67** (1980) 121–128
32. Jacobson, I.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison Wesley (2004)
33. Glott, R., Haaland, K., Bannier, S.: D3.1 Draft Business Model Risk Requirements Report (2013) Deliverable of the RISCOSS FP7 project (grant 318249).
34. Goeminne, M., Claes, M., Mens, T.: A historical dataset for the gnome ecosystem. In: Proceedings of the 10th Working Conference on Mining Software Repositories. MSR ’13, ACM (2013) 225–228
35. Jergensen, C., Sarma, A.: The onion patch: migration in open source ecosystems. In: Proceedings of the 19th SIGSOFT and 13th ESEC/FSE, ACM (2011) 70–80
36. Goeminne, M., Mens, T.: Analyzing ecosystems for open source software developer communities. [20] 247–275
37. von Wangenheim, C., Hauck, J., Zoucas, A., Salviano, C., McCaffery, F., Shull, F.: Creating software process capability/maturity models. Software, IEEE **27** (2010) 92–94
38. Samoladas, I., Gousios, G., Spinellis, D., Stamelos, I.: The sqo-oss quality model: Measurement based open source software evaluation. In Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G., eds.: Open Source Development, Communities and Quality. Volume 275 of The International Federation for Information Processing. Springer US (2008) 237–248
39. Manikas, K., Hansen, K.M.: Reviewing the health of software ecosystems-a conceptual framework proposal. In: Proceedings of the International Workshop on Software Ecosystems, Citeseer (2013) 33–44

SLR References

R1 Gamalielsson, J., Lundell, B., and Lings, B. (2010). The Nagios community: An extended quantitative analysis. In *Proceedings of the 6th OSS*, pages 85–96. Springer.

R2 Goeminne, M. and Mens, T. (2010). A framework for analysing and visualising open source software ecosystems. In *Proceedings of IWPSE-EVOL*, pages 42–47.

R3 Goeminne, M. and Mens, T. (2013). *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*, chapter Analyzing ecosystems for open source software developer communities, pages 247–275. In [20].

R4 Hartigh, E., Visscher, W., Tol, M., and Salas, A. J. (2013). *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*, chapter Measuring the health of a business ecosystem, pages 221–245. In [20].

R5 Jansen, S., Souer, J., Luinenburg, L., and Brinkkemper, S. (2012). Shades of gray: Opening up a software producing organization with the open software enterprise model. *Journal of Systems and Software*, 85(7):1495–1510.

R6 Jergensen, C. and Sarma, A. (2011). The onion patch: migration in open source ecosystems. In *Proceedings of the 19th ACM-SIGSOFT*, pages 70–80.

R7 Kabbedijk, J. and Jansen, S. (2011). Steering insight: An exploration of the ruby software ecosystem. In *Proceedings of the 2nd ICSOB*, pages 44–55. Springer.

R8 Kilamo, T., Hammouda, I., Mikkonen, T., and Aaltonen, T. (2012). From proprietary to open source - Growing an open source ecosystem. *Journal of Systems and Software*, 85(7):1467–1478.

R9 Lungu, M., Lanza, M., Gîrba, T., and Robbes, R. (2010). The Small Project Observatory: Visualizing software ecosystems. *Science of Computer Programming*, 75(4):264–275.

R10 Lungu, M., Malnati, J., and Lanza, M. (2009). Visualizing gnome with the small project observatory. In *Proceedings of the 6th MSR*, pages 103–106. IEEE.

R11 Mens, T. and Goeminne, M. (2011). Analysing Evolution of Social Aspects of Open Source Software Ecosystems. In *Proceedings of the 3rd IWSECO*, pages 1–14.

R12 Neu, S., Lanza, M., Hattori, L., and D'Ambros, M. (2011). Telling stories about GNOME with Complicity. In *Proceedings of the 6th VISSOFT*, pages 1–8.

R13 Scacchi, W. and Alspaugh, T. A. (2012). Understanding the role of licenses and evolution in open architecture software ecosystems. *Journal of Systems and Software*, 85(7):1479–1494.

R14 Shao, J., Kuk, G., Anand, S., Morley, J. G., Jackson, M. J., and Mitchell, T. (2012). Mapping Collaboration in Open Source Geospatial Ecosystem. *Transactions in GIS*, 16(4):581–597.

R15 Ververs, E., van Bommel, R., and Jansen, S. (2011). Influences on developer participation in the Debian software ecosystem. In *Proceedings of the MEDES*, pages 89–93. ACM.

R16 Weiss, M. (2011). Economics of collectives. In *Proceedings of the 15th SPLC*, pages 39:1–39:8. ACM.

R17 Yu, L., Cawley, J., and Ramaswamy, S. (2012). Entropy-Based Study of Components in Open-Source Software Ecosystems. *INFOCOMP Journal of Computer Science*, 11(1):22–31.

Appendix A: Measure Definitions

Amount of inquires or feature requests Number of inquire or feedbacks received for the OSS community. Contributions could be corrective, adaptive, perfective or preventive. R8.

Betweenness centrality Reflects the number of shortest paths that pass through a specific node. R1.

Bug tracking activity Number of comments created in project bug tracker and total number of actions in the bug tracker. R2, R3, R6, R8, R15.

Buildup of assets Total factor productivity over time. Can be measured using individual company data. R4.

Centrality Number of relations clique memberships. Number of individual network relations of a partner. The more central partner is the most persistent. When the partners are in clique or cluster, its persistence is considered high. Because is regarded as a secure environment. R1, R4, R6, R7.

Cluster of collaborating developers The nodes are developers and the edges between them represent projects on which they collaborated. They both make modifications to the project for at least a certain number of times. R9, R10.

Code vocabulary map Summary of terms used in the source code of the project. The vocabulary map is a tool for the developer who wants to obtain a general overview of the domain language of a project. R9.

Community effort The combined effort of all members belonging to community. R3.

Contributor activity graph The contributor distribution at ecosystem level. R12.

Contributor commit rate Average between first and last commit. R12.

Date of last commit Date of last commit of a project/community. R11.

Developer activity diagrams Give an overview of the contributors daily activity within an ecosystem. R10, R12.

Distribution over the species Variety measure for niche creation factor. The equality of the division of partners over the species. E.g., the distribution between numbers of resellers, number of system integrators, numbers of OEM’s. R4, R11.

Ecosystem cohesion Number of relations present in a subgroup/maximum possible of relation among all the nodes in the sub-group. R4.

Ecosystem connectedness Number of relations as a proportion of the theoretically maximum number of relations in all ecosystem. Is a metric of connectedness. Is a property that keeps communities structure safe from risks, guaranteeing their well-being and health. R4.

Ecosystem Entropy The second law of thermodynamics, in principle, states that a closed system’s disorder cannot be reduced, it can only remain unchanged or increase. A measure of this disorder is entropy. This law also seems plausible for software systems; as a system is modified, its disorder, or entropy, always increases. Can be viewed as being similar to the measurement of the existence of order or disorder among the participating software components, software products, or software organizations. R17.

Expertise view contributor Visualization about a contributor expertise based on file extensions (number and type of files changed within a month). R12.

Files changed Number of files that has been changed. R12.

Files per version Number of files per version. R6, R11, R12.
Geographical distribution Geographical distribution of community members. R2.
Job advertisements Number of job advertisements on the project/community. R8.
Lines added Lines added. R7, R12.
Lines changed Lines changed. R12.
Lines removed Lines removed. R12.
Liquidity Provide an indication whether a partner is able to meet its short-term obligations. Can be measured with: financial status of a partner; counting the number of new members in a business ecosystem. R4.
Mailing list Number of messages posted to project mailing lists and the number of responses obtained from those messages. R1, R2, R6, R11, R15.
Maximum number of commits of a developer The size and density of a contributor in a project. R3, R12.
Member activity rate Activity rate 1 means that a single person carries out all the work. R11.
Member effort The effort of member *m* in community *c*. R3, R10, R11.
Network resources Measure for delivery innovations factor of productivity. They can be measured directly, e.g., using balance of partners, but also indirectly, through the network relations. R4.
Networks node connection Connections between central and non-central species or partners. R4.
New members Counting the number of new members at any point in time. R4.
Number of active projects Number of active projects. R3, R10.
Number of authors Number of authors for projects. Author can change files in a project. R3, R11.
Number of bug fixers Number bug fixers in the community. R8.
Number committers Number of committers per project. R3, R9, R11.
Number of activity communities The number of activity communities in which member *m* is involved. R3, R7.
Number of commits Total number of commits containing source code, documentation, and translation. Average number of commits per week (project/community). R3, R6, R9, R10, R11, R12, R14, R15.
Number of community project Number of projects built on top of the platform of a community. R3, R8.
Number of contributors Total of contributors per project. R3, R8, R12.
Number of core developers Core developer contribute most of the code and oversee the design and evolution.
of the project. R6, R10.
Number of developer releases Number of releases that a developer has been active on a project. R6.
Number of developer projects Number of projects of a developer. R12.
Number of downloads Number of downloads from the official community portal or mirrors. R7, R8.
Number of event people The number of people participating in project events and meetings gives direct information on the activity in the community. R8.
Number of files Files during projects life. R14, R11.

Number of mailing list users Number of users subscribed to the project mailing list. R2, R6, R8, R11.

Number of members The number of activity members involved in community *c*. R3, R5, R6, R16.

Number of nodes and edge Number of nodes and edges. R1.

Number of passive user Passive users in the community. R8.

Number of readers Number of readers in the community. R8.

Number of scientific publications Number of scientific publications mentioning the community. R8.

Outdegree of keystone actors Is defined as someone who has a lot of developers he works with and also plays a large role in the software ecosystem. R7

Principal member activity The principal activity of a member *m* for a given time *t*. Community *c* for which *m* carried out the most effort. R3, R10, R11.

Project activity diagrams Allow identify the project evolution comparing six metrics; calculating the contributors involvement distribution. R2, R12.

Project developer experience Total number of releases in which the developer was active. R6.

Reciprocity of the ecosystem *(definition not provided)*. R7.

Relation between categorical event and developer participation Relation between categorical event and developer participation. R15.

Social media hits Number of hits the project gets in the social media. R7, R8.

Solvency Value creation measure for niche creation. Can be measured by standard metrics such as revenue share or profit share of newly introduced products or technologies. An alternative is to look at the build-up of partner equity. R4.

Temporal community effort The combined effort of all members belonging to community *c* during time period *t*. R3.

Total effort of members Total effort done by a particular community member *m* in a set of communities *C*. R3.

Variety in products Offered by the partner depends on alliances with other partners. Euclidean distances towards the overall mean of the business ecosystem can be used to measured most of these variety of scores. R4, R13.

Variety of partners Covariance with market indicates the variety of different partners a partner has. R4.

Visibility Tell us something about the centrality of a partner in the market. Popularity of the partner R4.

Web page requests Total request to OSS community web page. R8.

Zeta model Bankruptcy classification score model. R4.

Z-score Bankruptcy model to test the creditworthiness and solvency of partners. R4.

Appendix B: QuESo Measures in *GNOME* ecosystem

Measure	Sources
Bug tracking activity	R2, R3, R6, R11
Centrality	R6
Cluster of collaborating developers	R10
Community effort	R3
Contributor activity graph	R12
Contributor commit rate	R12
Date of last commit	R11
Developer activity diagrams	R10, R12
Distribution over the species	R11
Expertise view contributor	R12
Files changed	R12
Files per version	R6, R11, R12
Geographical distribution	R2
Lines added	R12
Lines changed	R12
Lines removed	R12
Mailing list	R2, R6, R11
Maximum Number of commits of a developer	R3, R12
Member effort	R10
Members activity rate	R11
New members	R6
Number community projects	R3
Number of active projects	R3, R10
Number of activity communities	R3, R7
Number of authors	R3, R11
Number of commiters	R3, R11
Number of commits	R3, R6, R10, R12
Number of contributors	R3, R12
Number of core developers	R6, R10
Number of developer projects	R12
Number of developer releases	R6
Number of mailing list users	R2, R6, R11
Number of members	R3, R6
Principal member activity	R3
Project activity diagrams	R2, R12
Project developer experience	R6
Temporal community effort	R3
Total effort of members	R3