WP1

DIGIT B1 - EP Pilot Project 645

Deliverable 6: Final Metrics Definition

Specific contract n°226 under Framework Contract n° DI/07172 – ABCIII

March 2016

DIGIT Fossa WP1 – Governance and Quality of Software Code – Auditing of Free and Open Source Software.

Deliverable 6: Final Metrics Definition

Author: XYZ

Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Commission. The content, conclusions and recommendations set out in this publication are elaborated in the specific context of the EU – FOSSA project.

The Commission does not guarantee the accuracy of the data included in this study. All representations, warranties, undertakings and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products. Neither the Commission nor any person acting on the Commission's behalf may be held responsible for the use that may be made of the information contained herein.

Contents

List of tables

List of Figures

Acronyms and abbreviations

EUI	European Institutions
EC	European Commission
EP	European Parliament
DG	Directorate General
FOSS	Free and Open Source Software
FOSSA	Free and Open Source Software Auditing
OS	Operating System
SDLC	System Development Life Cycle
WP	Work Package

1. Introduction

1.1. Objective of this Document and Intended Audience

This document represents the deliverable 6 included within TASK-04: Final metrics definition.

The objectives of this document are:

- To identify and categorise the aspects that can affect the sustainability of FOSS projects;
- To provide a list of the most relevant metrics that can be used to evaluate the sustainability of FOSS projects;
- To provide a tool to measure these metrics.

This document is addressed to the DIGIT areas interested in the use of these metrics to evaluate the sustainability of FOSS projects.

1.2. Document Structure

This document consists of the following sections:

- Section 1: **Introduction**, which describes the objectives of this deliverable and the intended audience, the structure of the document and the key success factors.
- Section 2: **Metrics to analyse the sustainability of FOSS projects,** which identifies and describes the metrics and respective categories that can be used to evaluate the sustainability of these projects.
- Section 3: **Metric Measurement Approach, which** describes the process for measuring the metrics.

1.3. Key Success Factors

All the steps described in Section 2 – Metrics to analyse the sustainability of FOSS projects, will ensure the fulfilment of the key success factors related to this deliverable:

- FOSSA outcomes provide new tools for CISO to measure the risk level of open source components.

1.4. Deliverables

1 *Deliverable 4: Analysis of Software Development Methodologies Used in FOSS communities*

2. Metrics to Analyse the Sustainability of FOSS Projects

If you are going to rely on a FOSS community contribution-based project for your own project, you want to ensure that the community will continue to support it throughout the lifecycle of your project. For any FOSS project, the sustainability of its communities is fundamental for its long term success.

There are many different aspects of a FOSS project that can affect the community sustainability: Good project management, an effective structure of governance, fair licensing, leadership, community activity and performance, and support from external entities are key for healthy and sustainable FOSS communities.

In this section, we will identify the aspects that can affect the sustainability of FOSS projects, and we will design a set of measurable metrics that can be used to evaluate the sustainability of these projects

2.1. Identification and Analysis of the Complete Set of Aspects that Can Affect the Sustainability of the FOSS Projects

In order to identify and analyse the complete set of aspects that can affect the sustainability of the FOSS projects, we researched and gathered information from several sources:

1 Everis FOSS expert team
2 The websites of the communities that were analysed in Deliverable 4
3 Relevant websites and research papers (see Section 4. Bibliographical References)

The information gathered was analysed and, as a result, we defined six categories of metrics, as follows:

1. Community Activity

The overall activity of the community and how it evolves over time is a useful metric category for all open source communities.

The Community Activity provides a first view into how much the community is doing, and it can be used to track the different activities that the community conducts, such as:

1. How many people took part in a relevant amount of a particular activity, like code development, code review, bug fixing?
2. Number of commits, releases, tickets
3. Communications activity (Mailing list, posts, forums, chat history)

4. Number of adoptions/implementations by external organisations / communities
5. Software evolution in terms of code, architecture and bug resolution, which is an indicator of the maturity of the project

2. Performance

Performance allows you to analyse how processes and people are completing their tasks. For example, you can measure:

1. How long processes take to finish, like implementing a new feature, fixing a bug, or conducting code review.
2. The time that it takes to resolve or close tickets
3. The time spent conducting code review

3. Quality and Security

Quality and security are two very important factors to evaluate for the sustainability of a project, for two main reasons:

1. A methodology that checks the quality of the code and ensures that different types of testing are conducted, which will also help the project to be of greater interest to the communities.
2. A project that has included security from the design stage, and implements it throughout its lifecycle, has a much better chance to live longer, because the identified security risks will be mitigated.

4. Demographics and Diversity

Demographics give us an overview of the developers and users around a project, and the companies that engage in it. This includes hosting and support providers, consultancy and customisation services, and companies that integrate the software with other products as part of solutions.

The number of companies involved in a project is an important indicator, since such companies will clearly have a strong interest in the sustainability of the software.

A sustainable project accumulates partners and providers of increasing specialisation. Likewise, if there are signs of service companies moving away from supporting the project this may be an indicator of underlying problems. As a result, projects that have been in production for a long time have a better chance to stay in the long run.

Another factor to take into consideration is the existing knowledge in the external market, regarding the language and platforms used in the project. This factor is extremely important because a project based on a very specific piece of knowledge that is not easily found or not of interest to the outside community of developers may find it difficult to stay in the long term, therefore directly affecting the sustainability of the project as a whole.

Diversity is an important factor in the resilience of communities. In general, the more diverse communities are—in terms of people or organisations that participate—the more resilient they are. For example, when a company decides to leave a FOSS community, the potential problems that the departure may cause are much smaller if its employees were contributing 5% of the work rather than 85%.

For the organisations that support the project, it is quite useful to look at their diversity in several ways:

1. Do they operate only in one country, or are they geographically spread out? And if so, in different continents?
2. Are they a mix of small and large companies?
3. Do they target a single sector or multiple industry sectors?

5. Governance

Governance is essential for the sustainability and evolution of a FOSS project and its associated communities.

It gives information on:

1. How the project is organised
2. Who is who in the project
3. If a roadmap exists
4. How well documented the project is
5. The licensing structure

6. FOSS Support

Support, either financial, tangible assets or workforce, is needed to ensure the sustainability of the FOSS project and its associated communities. This support can take various forms:

1 Financial

2 Infrastructure assets

3 Human Resources

2.2. Design of a Set Of Metrics

The objective of this task is to define a set of metrics with detailed aspects that will make it easy to measure the sustainability of the FOSS projects.

After the information gathering and the analysis conducted in task *2.1 Identification and analysis of the complete set of aspects that can affect the sustainability of FOSS projects*, a total of 34 metrics were defined and grouped in the six categories identified. Table 1 shows the categories with their corresponding metrics.

Table 1: Categories with their corresponding metrics

Category	No.	Metric Name
Community Activity	1	Code Activity (contributions and contributors)
	2	Release History
	3	Number of Commits
	4	Number of Tickets
	5	Communications (Mailing list, posts, forums, chat history)
	6	Number of Adoptions/Implementations by External Organisations / Communities
	7	SW Evolution (code, architecture, bug/feature)
	8	Programming Language Used
	9	Project Domain (OS, Application SW, IDE, Application servers, Libraries, desktop Environments and frameworks). I.e. Apache, Linux, Eclipse, Mozilla, Ant, GNoME, KDE)
	10	Source Code (repositories like CVS/SVN for code base, GitHub, source forge).
Performance	11	Time to Resolve Tickets
	12	Time Spent in Code Reviews
	13	Pending Work
Quality and Security	14	Security Requirements
	15	Threat Modelling
	16	Security Code reviews
	17	Security Testing
	18	Vulnerability Management
	19	Software Development Methodologies
	20	SLA

Category	No.	Metric Name
Demographics and Diversity	21	Longevity
	22	Real Knowledge Existent in the market of the language and Platforms Used.
	23	People Participating
	24	Organisation Participating
	25	Geographically distributed user community
Governance	26	Project Management
	27	Project Roadmap
	28	Project Structure
	29	Documentation
	30	Licensing
	31	Training
FOSS Support	32	Funding - Monetary
	33	Work force
	34	Infrastructure assets

2.3. Define Metrics Criteria

In order to design the forms that will be used to compile all the information for each metric, we defined the following criteria:

1. **Metric Name**: Descriptive name of the metric.
2. **Description:** what the metric should accomplish.
3. **Unit of Measurement:** it refers to the way the metric will be measured: a number, a maturity level, etc.
4. **Method:** it defines how the metric will be measured.
5. **Measurement:** it defines the actual measurement of the metric, i.e. the maturity level.
6. **Result:** the formula applied to measure the metric.

All the information of each metric is documented in the following forms, grouped in one of the 6 categories defined in *Task 2.1 Identification and analysis the complete set of aspects that can affect the sustainability of FOSS projects*

2.3.1. Community Activity

M1	Metric Name	Code Activity (contributions and contributors)
Description	For a project to be sustainable it must have contributors, and its codebase needs to be evolving. One can track this by looking at the project's revision control system and looking at the pattern of contributions. This metric measures the amount of committers that contribute to a majority of the commits in the project.	
Unit of Measurement	Ratio of contributors	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the pattern of contributions, to identify the number of contributors who submitted 80% of the total contributions in a specific period of time (mostActiveContributors80). Formula to calculate the ratio of contributors: **Contributors ratio = (mostActiveContributors80 / (mostActiveContributors80 + 1% x totalContributors)) x (totalContributors/ totalContributors + 10)**	
Measurement	**1. Very split:** Ratio value within the upper 20% of the maximum ratio **2. Split: Ratio** value ranked between 79% and 60% of the maximum ratio **3. Average:** Ratio value ranked between 59% and 40% of the maximum ratio **4. Dependant:** Ratio value ranked between 39% and 21% of the maximum ratio **5. Very dependant**: Ratio value within the lowest 20% of the maximum ratio	

<table>
<tr><th>M2</th><th>Metric Name</th><th>Release History</th></tr>
<tr><td></td><td colspan="2"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">This metric measures the approach followed for releases that provide information on the update frequency
1. Regular releases (disruption in the cycle might indicate sustainability or governance issues, in which case the best way to find out is to go into the project communications area and see if there is an issue)
2. Releases on a “need to have" basis. Some projects make releases as and when they feel ready, so they do not follow an established frequency.
3. When do releases occur? On the weekends (suggesting a hobby) or during the week (suggesting a business)?</td></tr>
<tr><td><b>Unit of Measurement</b></td><td colspan="2">Release frequency</td></tr>
<tr><td><b>Method</b></td><td colspan="2">Look at the release pattern for a certain period of time</td></tr>
<tr><td><b>Measurement</b></td><td colspan="2"><b>1 Optimised:</b> formal approach, regular releases are planned and delivered periodically, with the exception of security fixes.
<b>2 Managed:</b> informal approach, release is published when development objectives are achieved.
<b>3 Initial</b>: informal approach, release is published without clear definition criteria.</td></tr>
</table>

M3	Metric Name	Number Of Commits
Description	The number of commits gives a general idea about the volume of the development effort.	
Unit of Measurement	Number of commits	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the number of code commits done by contributors during - last year. The number of most active contributors will be those that submitted 50% of the total contributions Formula to calculate the ratio: **Commits Ratio = (nCommitsLastYear / nNumberCommitsLastYearTopPopularGitHubRepository) *100**	
Measurement	1 **Very active:** Ratio value within the upper 51% of the maximum ratio 2 **Active**: Ratio value ranked between 26% and 50% of the maximum ratio 3 **Average:** Ratio value ranked between 6% and 25% of the maximum ratio 4 **Inactive**: Ratio value ranked between 1% and 5% of the maximum ratio 5 **Very Inactive:** Ratio value within the lowest 1% of the maximum ratio	

M4	Metric Name	Number Of Tickets
Description	The number of tickets opened provides information about how many bugs are reported or the new functionalities that are proposed.	
Unit of Measurement	Ratio of tickets created	
Method	This analysis will be carried out by checking the community's main tasks or ticket repository. The information to look for will be when the tickets are created	
Measurement	**1** **Very active**: there are, at least, 10 tickets created in the last week. **2** **Active**: there are, at least, 10 tickets created in the last two weeks. **3** .**Average:** there are, at least, 10 tickets created in the last month. **4** **Inactive**: there are, at least, 10 tickets created in the last three months. **5** **Very Inactive:** rest of the values	

M5	**Metric Name**	**Communications (Mailing list, posts, forums, chat history)**
Description	The number of messages in mailing lists or posts in forums gives an idea of how many discussions are being held in public. However, this metric needs to differentiate the types of activities that are conducted in the communications, which can range from some serious discussions to unnecessary flame wars (in this case, the communication channel should not be accounted for).	
Unit of Measurement	Number of active communication channels	
Method	This analysis will be carried out by checking official communication channels provided by the community. The information to look for will be the number of active communication channels used by the community.	
Measurement	**1** **Optimised:** More than three communication channels are used (different mailing lists, IRC, wiki, user forums and web post are used for the project). **2** **Managed:** At least three communication channels are used in the project. **3** **Initial:** less than three channels are used for exchanging information.	

M6	Metric Name	**Number of Adoptions/Implementations by External Organisations / Communities**

Description	Software downloads provide information about the global interest in the project Each distribution platform provides its own metrics to describe popularity. For example, on GitHub, watchers, stars, and forks are the strongest indicators of a project's popularity and use. On WordPress.org, you can see the number of downloads a plugin receives, as well as its average user rating. If distributed via package manager (e.g., Rubygems, NPM), you can see the number of installs. These indicators show how much the project is used.
Unit of Measurement	Interest level
Method	This analysis will be carried out by checking distribution platforms. The information to look for will be the identification and measurement of the interest, in order to rank it within the levels defined. This level of interest will be measured by means of doing the following assessment: Taking the 5 most downloaded/popular projects, an average will be assessed (Av). The level of popularity (using the Alexa ranking) of the project or the number of downloads (P) will be divided by that average. The result is the adoptions ratio (Ra). **Ra = P / Av**
Measurement	1 **Very Interesting**: The ratio value is larger than 1 2 **Interesting**: The ratio value is between 1 and 0,51 3 **Normal** The ratio value is between 0,50 and 0,26 4 **Disappointing**: The ratio value is between 0,25 and 0,11 5 **Very disappointing:** The ratio value is smaller than 0,10

M7	Metric Name	SW Evolution (code, architecture, bug/feature)
Description	This metric evaluates the evolution level of the software development cycle: 1 Code development follows a methodology 2 Improvements were made to the architecture supporting the software development 3 Improvements were made to the bug fixing process	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the project's development lifecycle and the evaluation of these three parameters: 1 Code development follows a methodology 2 Architecture Improvements 3 Improvements bug fixing process	
Measurement	**1** **Optimised:** The community applies all three parameters **2** **Addressed:** They accomplish two of the three parameters analysed **3** **Partially Addressed:** They accomplish one of the parameters **4** **Initial:** They don't address any of the parameters analysed	

M8	Metric Name	Programming Language Used
Description	This metric evaluates the use of a stable and widely used programming language	
Unit of Measurement	Use of the programming language	
Method	This analysis will be carried out by checking the community website and wiki. The goal is to measure the maturity of the programming language used using TIOBE Index as indicator. http://www.tiobe.com/tiobe_index	
Measurement	1 **Very popular:** First 5 entries from TIOBE 2 **Popular**: Languages ranked from 6 to 15 from TIOBE 3 **Average**: Languages ranked from 16 to 20 from TIOBE 4 **Unusual:** Rest of the languages from TIOBE	

M9	Metric Name	**Project Domain (OS, Application SW, IDE, Application servers, Libraries, desktop Environments and frameworks. I.e. Apache, Linux, Eclipse, Mozilla, Ant, GNoME, KDE…)**
Description	The sustainability of the projects increases if they belong to the most common domains: Operating Systems (OS), Application Software, Integrated Development Environments (IDE), Application Servers, Libraries, Desktop Environments and Frameworks. Examples of projects in these domains include Linux, Eclipse, Apache, Ant, Mozilla, GNOME, KDE, and ArgoUML This metric will evaluate if the project belongs to one of these domains.	
Unit of Measurement	Domain type	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the project's domain: 1. Common: Operating Systems (OS), Application Software, Integrated Development Environments (IDE), Application Servers, Libraries, Desktop Environments and Frameworks. Example projects under these domains include Linux, Eclipse, Apache, Ant, Mozilla, GNOME, KDE, and ArgoUML. 2. Not common	
Measurement	**1** Common Domain **2** Not common domain	

M10	Metric Name	Source Code (repositories like CVS/SVN for code base, GitHub, source forge).

Description	This metrics measures if the developer uses existing repositories to produce quality code. 1. Repositories maintaining the code base (e.g., CVS/SVN, change log) are data sources that contain information on the underlying software and its development process, ensuring that everything is commented. Comments are clear and free of misspellings, and the project includes extensive tests. 2. External sources, like SourceForge.net, repositories hosting thousands of FOSS projects
Unit of Measurement	Position in Alexa ranking
Method	This analysis will be carried out by checking the Alexa ranking for open source project hosting: http://www.alexa.com/topsites/category/Computers/Open_Source/Project_Hosting
Measurement	**1** **Popular Repository**: 1st, 2nd, 3rd positions **2** **Common Repository:** 4th, 5th, 6th positions. **3** **Independent Repository:** From 7th up to 15th positions. **4** **Marginal Repository**: Not ranked in the first 15 positions in Alexa ranking.

2.3.2. Performance

M11	Metric Name	Time to Resolve Tickets
Description	This metric measure the Time it takes to resolve or close tickets. This metric shows how the project is reacting to new information that requires another action, such as fixing a reported bug or implementing a requested new feature.	
Unit of Measurement	Average period to resolve a ticket	
Method	This analysis will be done by looking at the software development statistics during a certain period of time (for example, 6 months) The formula to calculate the average time is as follows: **Average time = sum(ticket solving time)/number of tickets**	
Measurement	1 **Optimised:** Average_time < 5 days 2 **Defined:** 10 days > Average_time >= 5 days 3 **Managed:** 15days > Average_time >= 10 days 4 **Basic**: 15days <= Average_time 5 **No data about this**	

M12	Metric Name	Time Spent in Code Reviews
Description	These metric measures the Time spent in code reviews —from the moment a change to the code is proposed, to the moment it is accepted—, and it shows how long it takes to upgrade a proposed change to the quality standards expected by the community. Other metrics deal with how well the project is coping with pending work, such as the ratio of new to closed tickets, or the backlog of still incomplete code reviews. Those parameters tell us, for example, whether or not the resources put into solving issues are enough.	
Unit of Measurement	Average time to do code reviews. (Considering the minimum number of code reviews before being accepted or rejected)	
Method	This analysis will be done by looking at the annual community reports. The formula to calculate the average time is as follows: **Average time = sum(code review acceptance time)/number of code reviews**	
Measurement	**1** **Optimised:** Average_time <= 3 days **2** **Defined:** 7days>= Average_time > 3 days **3** **Managed:** 15days>= Average_time > 8 days **4** **Basic:** Average_time > 15 days **5** **No data about this**	

<table>
<tr><th>M13</th><th>Metric Name</th><th>Pending Work</th></tr>
<tr><td colspan="3"></td></tr>
<tr><td><b>Description</b></td><td colspan="2">This metric measures the ratio of new to closed tickets, or the backlog of incomplete code reviews
This parameter is also an indicator of whether or not the resources put into solving issues are enough.</td></tr>
<tr><td><b>Unit of Measurement</b></td><td colspan="2">Ratio of new and closed tickets</td></tr>
<tr><td><b>Method</b></td><td colspan="2">The ratio between closed tickets (issues) and new ones will be done, if possible, taking a month as timeframe.
The formula to calculate this ratio is as follows:
<b>SolvingRatio = NewTickets/ClosedTickets * 100</b></td></tr>
<tr><td><b>Measurement</b></td><td colspan="2"><b>1</b> <b>Optimised:</b> SolvingRate <=33%
<b>2</b> <b>Controlled:</b> 33% < SolvingRate <= 66%
<b>3</b> <b>Managed:</b> 66% < SolvingRate <= 100%
<b>4</b> <b>Overloaded:</b> 100% > SolvingRate</td></tr>
</table>

2.3.3. Quality and Security

M14	Metric Name	Security Requirements
Description	This metric measures the existence and maturity level of the definition of security requirements in the early stages of the SDLC	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the definition of security requirements. If possible, the information will be verified by contacting the community.	
Measurement	**1** **Optimised:** Specific requirements (defined at the initial phases) **2** **Defined**: Within business requirements **3** **Managed:** Security requirements defined as needed **4** **Initial:** No Security Requirements	

M15	Metric Name	Threat Modelling

Description	This metric measures the existence and maturity level of threat modelling
Unit of Measurement	Maturity level
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the definition of the approach to threat modelling. If possible, the information will be verified by contacting the community.
Measurement	**1** **Optimised:** They have threat modelling and countermeasures are implemented or in the process of being implemented (managed) **2** **Managed**: No formal threat modelling, however some countermeasures are implemented (from previous experiences) **3** **Initial:** No threat modelling

M16	Metric Name	Security Code Reviews

Description	This metric measures the existence and maturity level of security procedures such as code reviews
Unit of Measurement	Maturity level
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the definition of the security code review process (security code reviews is being responsibly conducted). If possible, the information will be verified by contacting the community.
Measurement	**1** **Formal:** Security code reviews conducted by a specific team **2** **Informal:** Security code reviews conducted by community members **3** **No** security code reviews conducted

M17	Metric Name	Security Testing
Description	This metric measures the existence and maturity level of security procedures such as security testing (white box /black box)	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be if the definition of the security testing process (security testing is being conducted, specifying in which SDLC phase). If possible, the information will be verified by contacting the community.	
Measurement	**1** **Optimised:** Security testing conducted during development **2** **Defined:** Security testing conducted during testing **3** **Managed**: Security testing conducted before release **4** **Basic:** No security testing or conducted after release (user finds a vulnerability)	

M18	Metric Name	Vulnerability Management
Description	This metric measures the existence and maturity level of vulnerability management.	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the definition of the vulnerability management process. If possible, the information will be verified by contacting the community.	
Measurement	1 **Optimised:** Vulnerability management conducted by a dedicated team 2 **Defined:** Vulnerability management conducted as part of the security team´s responsibilities 3 **Managed:** Vulnerability management conducted by a closed group (community leaders, vulnerability stakeholders, trusted members)	

M19	Metric Name	Software Development Methodology
Description	This metric measures the existence and maturity level of the software development methodologies used	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the software development methodology used in the project. If possible, the information will be verified by contacting the community.	
Measurement	1 **Optimised:** Use of a standard methodology (i.e. Scrum, Agile, Kanban, Waterfall) 2 **Managed:** Use of their own documented methodology 3 **Basic:** Random, individual contributions	

M20	Metric Name	SLA
Description	An SLA that defines the parameters for ticket resolution, bug fixing, etc… This metric measures the existence and maturity level of an SLA	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the definition of an SLA in the project. If possible, the information will be verified by contacting the community.	
Measurement	**1** **Formal:** An SLA exists and is managed **2** **Informal:** An SLA does not exist, however, there is an informal procedure to resolve the issues	

2.3.4. Demographics and Diversity

M21	Metric Name	Longevity
Description	This metric measure how long the project has been in a "live" or production status. Some open source projects are long-lived, leading more conservative organisations to adopt the software, and maintain its use for longer, and resulting in a longer-term investment in its sustainability. If a project has survived long enough to undergo several technology replacement cycles, this is a good indication that it is going to be around for years to come. The warning signs appear when there seems to be subsequent migrations from one project community to another. Eventually, even a large, mature project will start to suffer if this happens.	
Unit of Measurement	Start year of the project	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the starting date of the project. If possible, the information will be verified by contacting the community.	
Measurement	**1** **Reference Project in FOSS environment:** Project started before 2000 **2** **Veteran Project:** Project started between 2000 and 2005 **3** **Experimented Project:** Project started between 2005 and 2010 **4** **Adult Project:** Project started between 2010 and 2015 **5** **Beginner Project:** Project started after 2015	

M22	Metric Name	Real Knowledge Existent in the Market about the Language and Platforms Used.

Description	The PYPL PopularitY of Programming Language Index is created by analysing how often language tutorials are searched on Google: the more a language tutorial is searched, the more popular the language is assumed to be. It is a leading indicator. The raw data comes from Google Trends.
Unit of Measurement	PYPL index
Method	This analysis will be carried out by checking the website: http://pypl.github.io
Measurement	1 **Popular programming language**: PYPL share >10% 2 **Common programming language**: 10% >= PYPL share >5% 3 **Specialised programming language**: 5%>= PYPL share

M23	Metric Name	People Participating
Description	This metric evaluates the different groups and number of active members that are participating as contributors or supporters of this community. Having a diversity of contributors indicates that there's a community of users who rely on and care about improving the software. Contributors need not be only technical. Look for those contributing to documentation processes, posting on support forums, or filing issues and feature requests. They can be grouped as: 1 Developers 2 Documenters 3 Supporters	
Unit of Measurement	Number of active groups	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the number of working groups or teams within the community. If possible, the information will be verified by contacting the community.	
Measurement	**1** **High:** Three or more groups **2** **Medium**: Two groups **3** **Low:** One group	

M24	Metric Name	Organisations Participating

Description	This metric evaluates the number of different organisations that are participating as contributors or supporters of this community. There are many open source projects that can meet the above mentioned criteria, but if none of the peers are using the project (or haven't even heard of it), that could be a major red flag. Many companies proudly showcase the open source projects they're built on, and Google searches can often reveal those that don't.
Unit of Measurement	Levels, indicating the number and relevance of supporting organisations
Method	This analysis will be carried out by checking community website and wiki. The information to look for will be the organisations that support the project. If possible, the information will be verified by contacting the community.
Measurement	**1** **Level 1**: Several big technological organisations participate in the project **2** **Level 2:** Only one big technological organisation participates in the project **3** **Level 3:** Several organisations participate in the project **4** **Level 4:** One organisation participates in the project **5** **Level 5:** No participating organisations

M25	Metric Name	Geographically Distributed User Community
Description	This metric evaluates how geographically spread out the user community is.	
Unit of Measurement	Number of continents	
Method	This analysis will be carried out by checking the community website and wiki. Identify the home country/continent of the current top contributors (100).	
Measurement	**1** **Geographically widely spread:** more than 4 continents **2** **Geographically spread:** Between 2 and 4 continents **3** **Geographically concentrated:** Less than 2 continents	

2.3.5. Governance

M26	Metric Name	Project Management
Description	This metric measures the existence and maturity level of the project management cycle	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the project's management cycle conducted by the community. If possible, the information will be verified by contacting the community.	
Measurement	**1** **Optimised:** Project Management is defined and implemented **2** **Defined:** Project Management is defined and documented, but does not completely follow the agreed methodology **3** **Managed:** Project management is conducted in an informal way **4** **Initial:** Project management is conducted as needed	

M27	Metric Name	Project Roadmap
Description	This metric evaluates the existence and maturity level of a project roadmap	
Unit of Measurement	Maturity level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the community's project roadmap. If possible, the information will be verified by contacting the community.	
Measurement	**1** **Optimised:** Project roadmap is defined and implemented **2** **Defined:** Project roadmap is defined and documented, but does not completely follow the agreed methodology **3** **No** project roadmap	

M28	Metric Name	Project Structure
Description	This metric evaluates if there is a formal structure for the project. 1 How is the project organised? 2 Who is behind the project, in terms of number of people? 3 Are they fully committed to the project or is it a partial assignment, done on a voluntary basis?	
Unit of Measurement	Documentation coverage defined in 3 levels	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the project structure (organogram). If possible, the information will be verified by contacting the community.	
Measurement	**1** **Optimised:** A formal structure with roles and responsibilities is defined, following an enterprise approach **2** **Managed:** An informal structure, with roles and responsibilities defined, although it may not be complete (i.e. no security roles) **3** **Initial**: Only leader and contributor roles are defined.	

M29	**Metric Name**	**Documentation**
Description	This metric will indicate the level of the documentation existent in the project. 1 Is it a readme file or a dedicated documentation site? 2 Does it have technical documentation that covers how to install, and specifies requirements, dependencies? 3 Does it have a user manual? 4 Does it have general documentation?	
Unit of Measurement	Documentation coverage defined in 3 levels	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the documentation of the project. If possible, the information will be verified by contacting the community.	
Measurement	**1 Full documentation:** a) developer guides (code style, code review, security review, development environment), b) user manual, c) technical manual (for system administrator), d) support wikis. **2 Partial documentation:** Only main documentation is developed, user-oriented and for developers **3 Basic documentation**: Only two types of documentation are developed, mainly user-oriented	

M30	Metric Name	Licensing
Description	This metric will indicate how serious the project is in terms of providing intellectual property. 1 Is the project properly licensed? 2 What type of license is provided? 3 Does it contain a license file or just a reference to a license in the readme? 4 Do files contain the proper headings, where required?	
Unit of Measurement	Intellectual property level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the license file of the project. If possible, the information will be verified by contacting the community.	
Measurement	**1** **Optimised:** Project has a license history, up-to-date license that contains proper headings **2** **Defined**: Project incorporates a license file with proper headings. **3** **Managed:** Project incorporates a license file without proper headings.	

M31	Metric Name	Training
Description	This metric measures if the project has provisions for regular training to ensure the quality of project deliverables	
Unit of Measurement	Training programmes coverage defined in 3 levels	
Method	Identification of the regular training provided by the project	
Measurement	**1** **Optimised:** Project has a complete set of documentation for newcomers (How to contribute, how community works, tools), and a mentor is assigned to help them to get started. **2** **Managed:** Project has a complete set of documentation for newcomers (How to contribute, how community works, tools) **3** **Basic:** Project has some informal information for newcomers (How to contribute, how community works, tools)	

2.3.6. FOSS Support

M32	Metric Name	Funding - Monetary
Description	This metric measures if the project is being supported by some kind of monetary funding from an external source	
Unit of Measurement	Funding level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the "Thanks" or "acknowledgment" part in the project/community website. If possible, the information will be verified by contacting the community.	
Measurement	1 **Optimised:** Different external organisations fund the project directly, or it is funded from a private organisation that does business with the FOSS 2 **Managed:** Different external organisations fund different projects in the same community. 3 **Basic:** No funding by third-party organisations, just individual donations.	

M33	Metric Name	Workforce
Description	This metric measures if the project is being supported by external volunteers who provide support in development, documentation or issue management tasks	
Unit of Measurement	Workforce level	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the "Thanks" or "acknowledgment" part in the project/community website. If possible, the information will be verified by contacting the community.	
Measurement	1 **Optimised:** there are paid human resources in all areas of the project, working exclusively in that area. Volunteers can also be part of the project 2 **Dedicated:** there are paid human resources working in one or more areas of the project. Volunteers can also be part of the project 3 **Volunteering:** There are only volunteers in the project.	

M34	Metric Name	Infrastructure Assets
Description	This metric measures if the project is being supported by the provision of equipment or software licenses from an external source This provision can come from a monetary donation or an actual asset donation	
Unit of Measurement	Type of infrastructure	
Method	This analysis will be carried out by checking the community website and wiki. The information to look for will be the "Thanks" or "acknowledgment" part in the project/community website. If possible, the information will be verified by contacting the community.	
Measurement	1 **Dedicated:** Community is the infrastructure owner 2 **Mixed:** Dedicated and shared infrastructure. 3 **Shared:** Infrastructure assets are shared with other communities	

3. Metrics measurement approach

Following the criteria defined and agreed upon in Section *2.3 Define Metrics Criteria*, we conducted the following activities to measure the metrics designed in Section *2.2 Design of a Set of Metrics*:

3.1. Tool to measure the metrics

1. Development of an Excel sheet, with all the metrics that were defined in Section *2.2 Design of a Set of Metrics* and all the metrics criteria defined in Section *2.3 Define Metrics Criteria*
2. Definition of a unit of measurement for each metric
3. Development of method to measure each metric. This method could be a formula to calculate the ratio of two values, or data obtained from the project website.
4. Each measurement is normalised, so all the metrics can be analysed on the same scale, in a quantitative way
5. To show the results in a graphic way, easy to understand, a set of example graphs are produced, to represent the results in a graphical way.

   To view the measurement tool, click on the icon below:

Metrics measurement
tool

3.2. Frequency of the measurement

Bitergia, a company focused on software development analytics, indicates in the article ‘On the Importance of Quarterly Reports: OPNFV and OpenStack as use cases’, that measurement of all the metrics should be conducted at least on a quarterly basis.

3.3. Responsible for the measurement

A team should be appointed to conduct the metric measurement of the selected FOSS projects.

For successful measurements, the team should have a suitable level of relevant skills and experience.

These skills include:

- Analytical thinking, to notice discrepancies and inconsistencies in available information.
- Communication skills, oral and written, to ensure that important information is shared with others appropriately and to communicate results
- Specific knowledge for particular categories, e.g. project management knowledge for the governance category, security knowledge for the Quality and Security category, etc.
- Experience in conducting metrics evaluations
- Teamwork

3.4. Results

Once the measurement is conducted, 8 types of graphs can be produced, as follows:

1. One for each of the categories defined in Section 2.1 Identification and Analysis of the Complete Set of Aspects that Can Affect the Sustainability of the FOSS Projects
2. A graph comparing each community against all 6 categories.

A sample of the graphs is shown in Figures 1 through 7

Figure 1: Activity

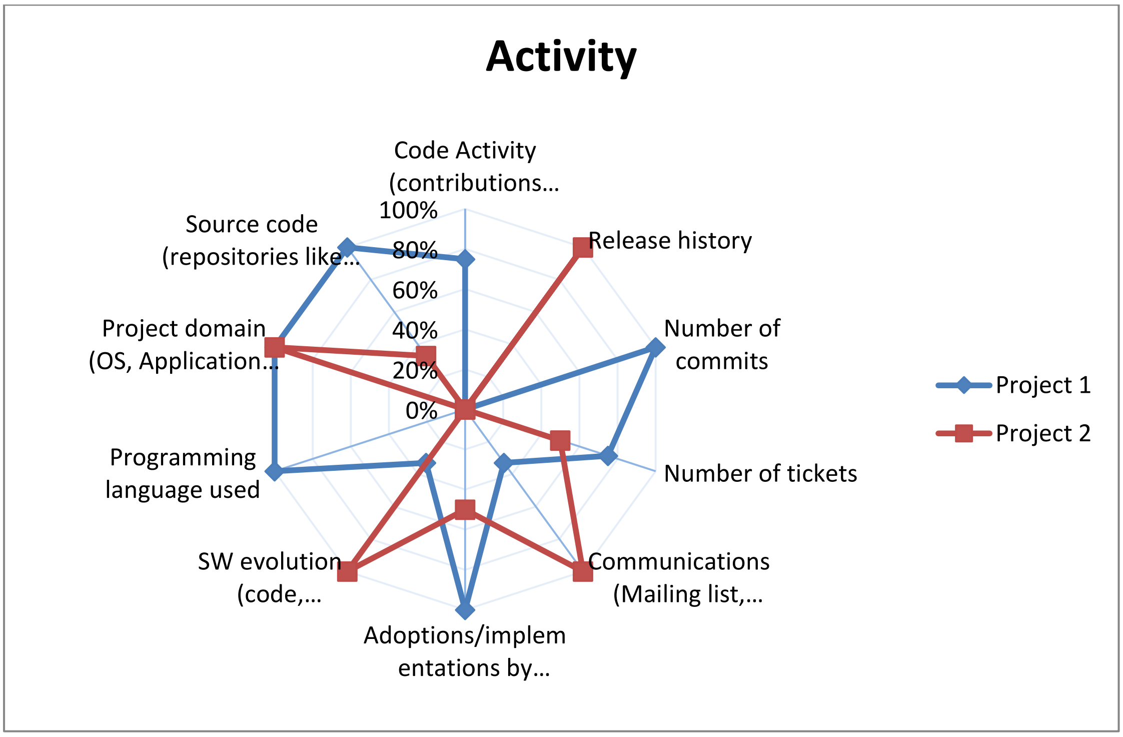


Figure 2: Performance

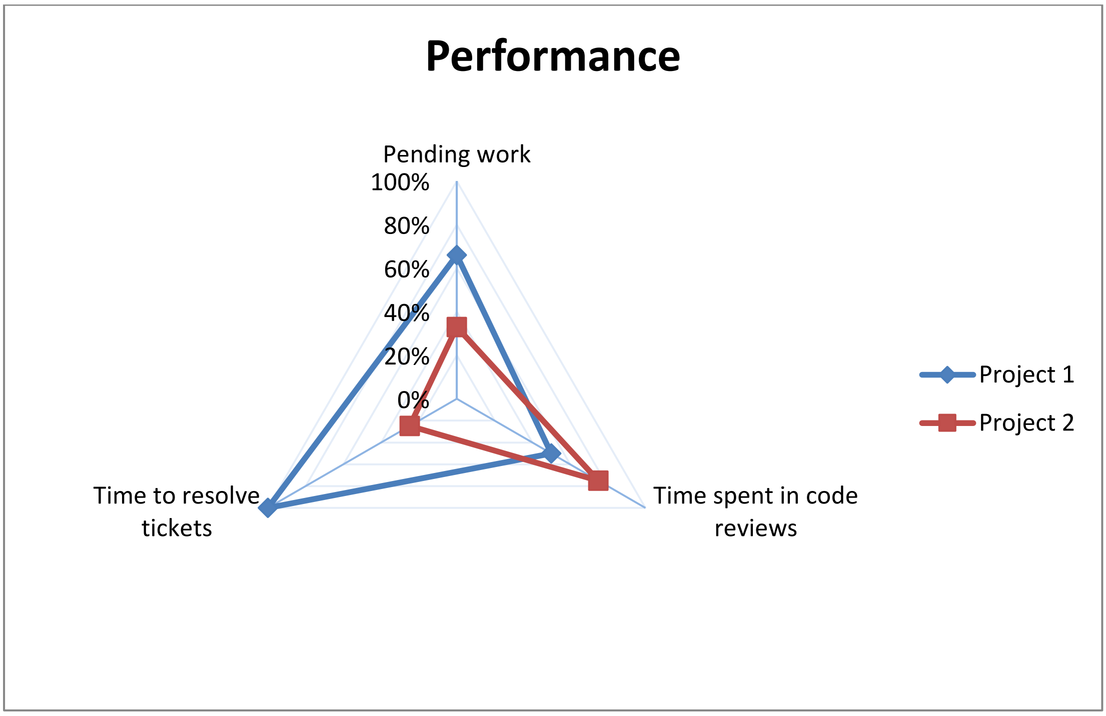


Figure 3. Quality and Security

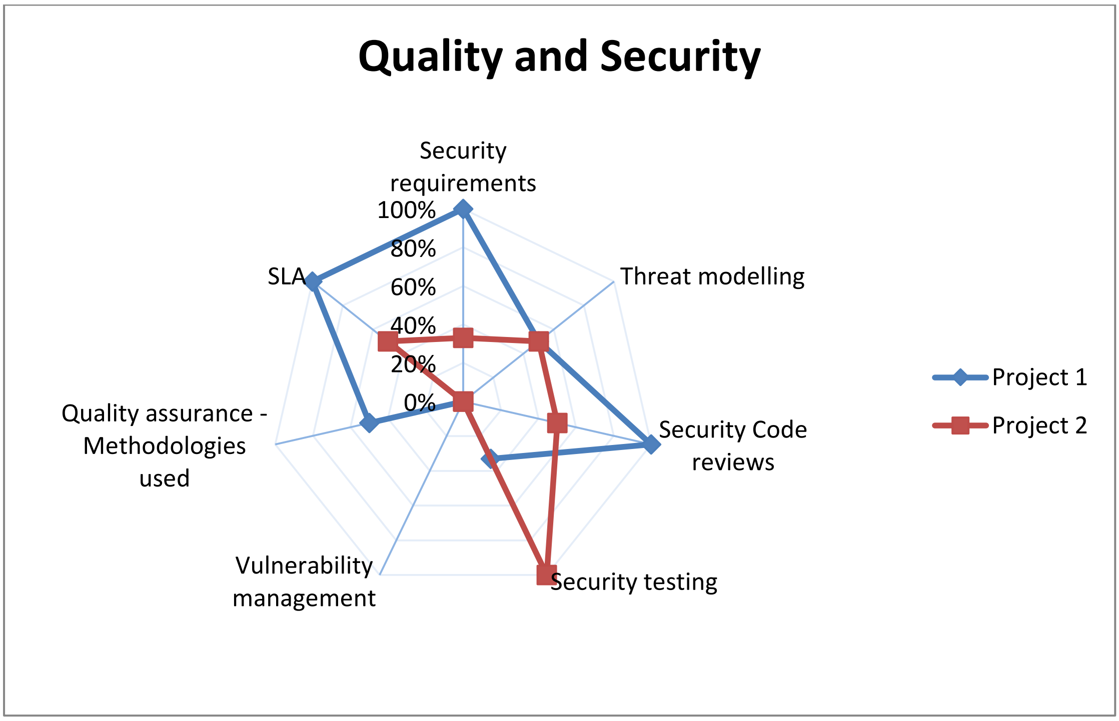


Figure 4: Governance

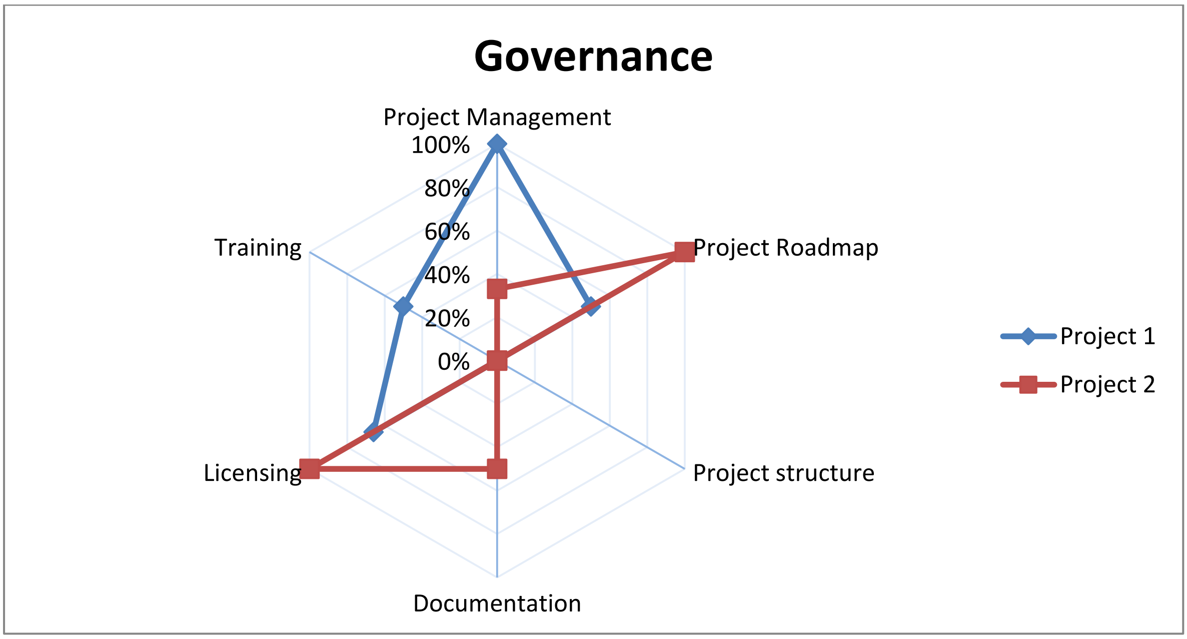


Figure 5. Demographics and Diversity

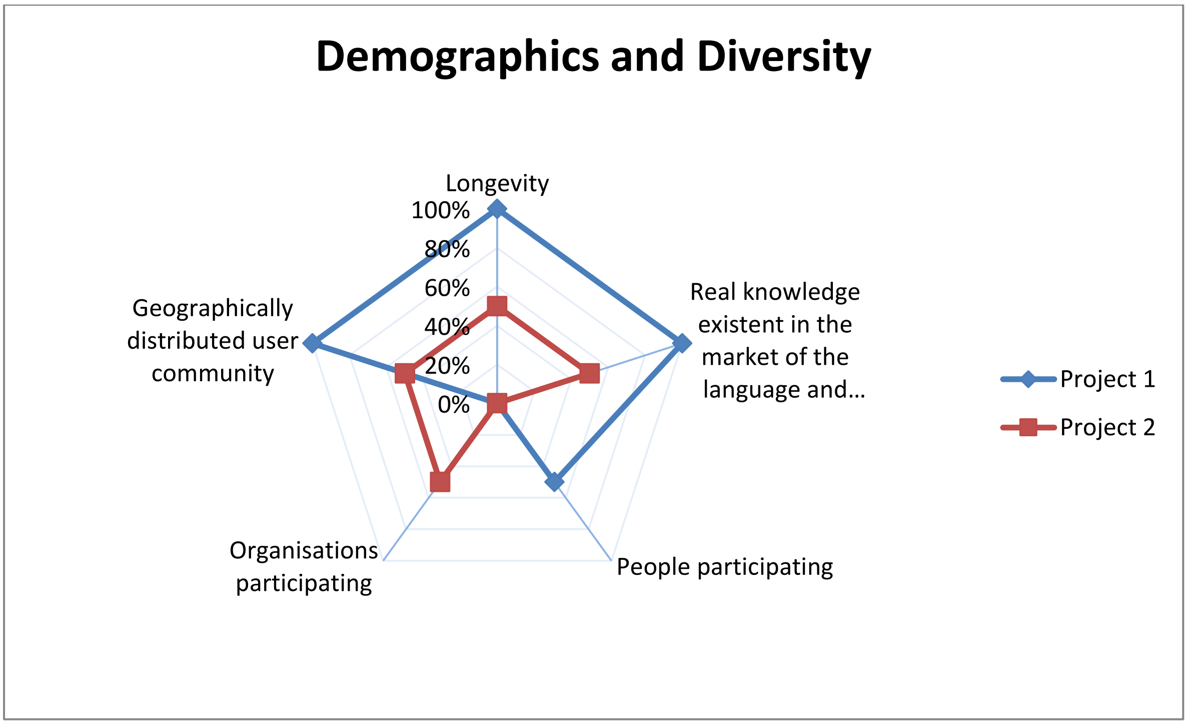


Figure 6. FOSS Support

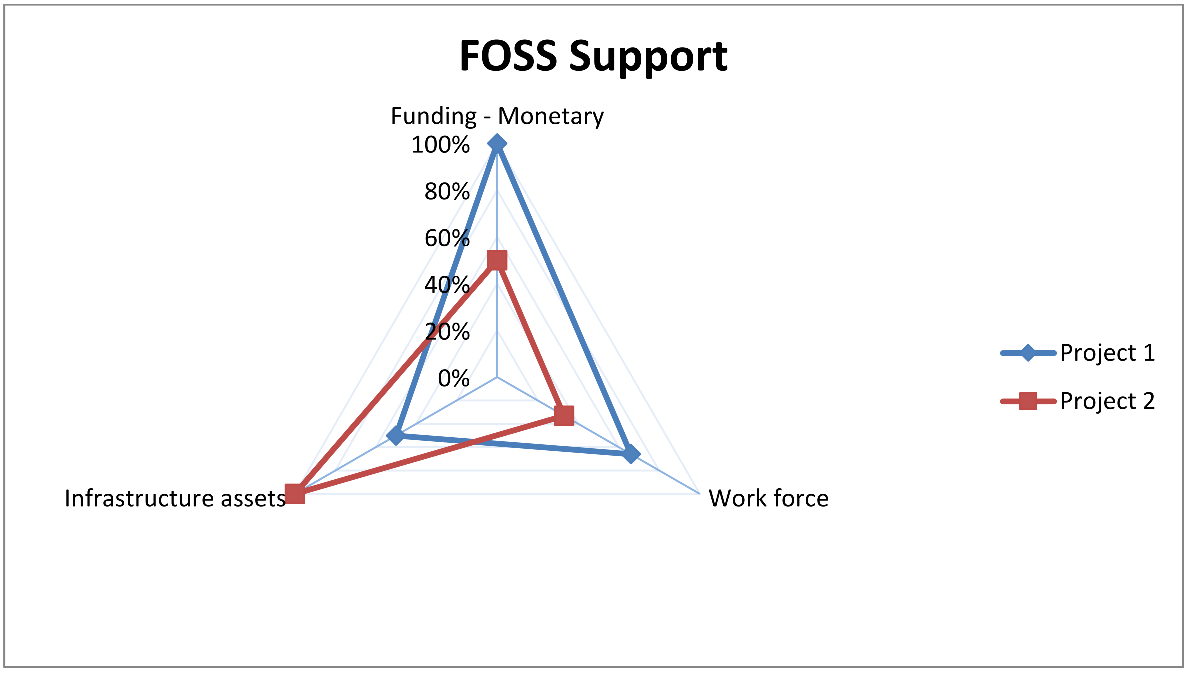


Figure 7. Comparison of Projects and Categories

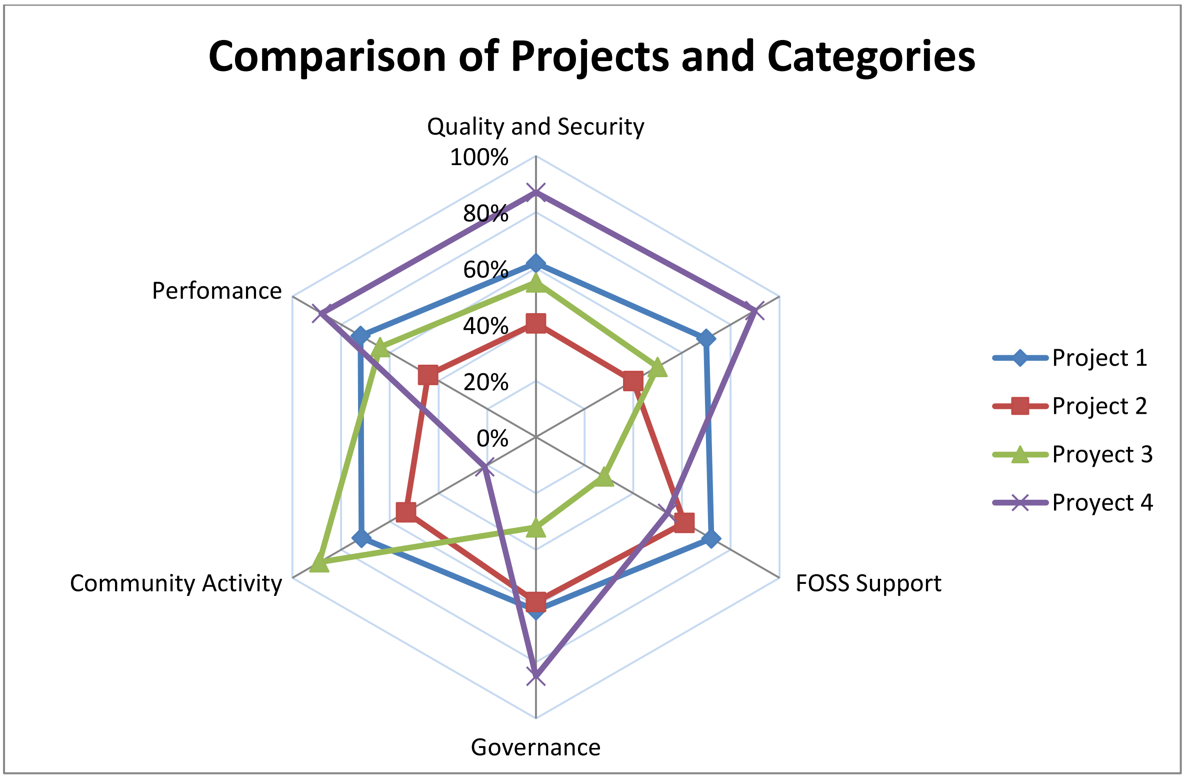
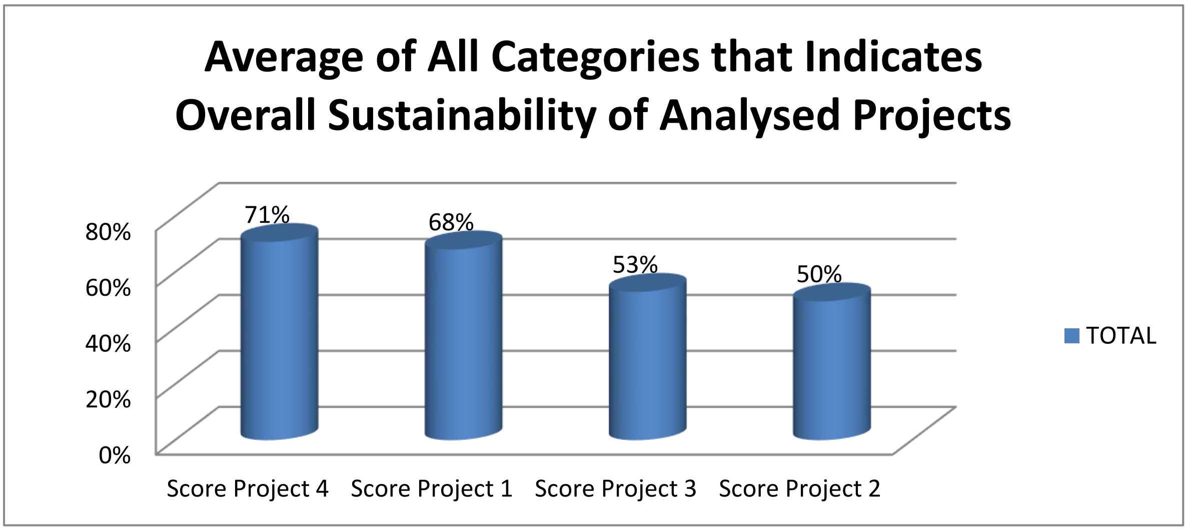


Figure 8. Average of All Categories that Indicates Overall Sustainability of Analysed Projects



4. Bibliographical references

Bibliographical references detail
https://github.com/jgbarah/evaluating-foss-projects
https://github.com/jgbarah/evaluating-foss-projects/blob/master/ evaluation_models.md
http://hbtlabs.github.io/oss-checklist/2014/06/24/checklist/
http://radar.oreilly.com/2014/10/measure-your-open-source-communitys-age-to-keep-it-healthy.html
https://www.openhub.net/ Black Duck
http://ben.balter.com/2014/06/02/how-to-identify-a-strong-open-source-project/
https://opensource.com/life/14/1/evaluate-sustainability-open-source-project
https://opensource.com/business/15/12/top-5-open-source-community-metrics-track
http://blog.bitergia.com/2016/03/15/quarterly-reports/