MSWL Project Evaluation: OpenNebula Project Analysis

Sergio Arroutbi Braojos

January 5, 2014

Contents

List of Tables

1 Executive Summary

This paper tries to perform an estimation of quality for a particular Cloud Computing Open Source Project, in this case, OpenNebula. Apart from that, a comparison of this project to other similar projects which are its competitors, such as CloudStack, Eucalyptus and OpenStack, is performed, considering the same quality model and how these projects score on it.

In particular, by means of using a role play, the selection of a Cloud Computing System by a company offering cloud computing, storage and network services for other companies, from the perspective of a member of the Board of Directors of the company.

By means of definition of a custom quality model, focused on the necessities of an invented role, OpenNebula project is analyzed, scored, and compared to the other projects. The final aim of this document is to take a decision of which is the best alternative to use, taking into account the most important factors in terms of quality that a Cloud Computing System must have for the invented role.

2 Introduction

This document is an approach to demonstrate the different issues and evaluations that can be taken into consideration in order to make a decision on the selection of an Open Source Project. A complete analysis of the quality of a particular project, in this case, OpenNebula, will be performed, by applying a self-defined quality model.

2.1 Document Objectives

Apart from previously approach, a comparison with the score of other projects into the same quality model will be performed, so that decision making will be structured by following next steps:

- Defining a role that justifies why some aspects are more or less important from a quality perspective, hence which factors are more valuable to consider in a quality model, in such way that allows to select or discard a particular project.
- Defining a quality model that has been initially agreed and drafted between different students, that allows to obtain a quality score so that comparison between other Cloud Computing Open Source Projects can be performed.
- Defining the different values in terms of scoring (weights) that all of the statements included in the quality model have, according to the role selected.

- Analyzing a particular Cloud Computing Open Source Project, OpenNebula in this case, by obtaining metrics of it and sharing with the rest of the students.
- Calculating the final scoring of this particular project, OpenNebula, according to the weights decided according to the role.
- And finally, by comparing the final score to the rest of the projects and justifying the selection of the "Winner Project".

2.2 Document Structure

This document follows next structure, in order to accomplish the different phases that must be considered before making the final decision on the Cloud Computing Open Source Project selected:

- **Introduction**. This chapter describes the final aim of this document, how it is structured and the role being taken in order to take into consideration the most important factors to make the final decision on the Cloud Computing Open Source Project.
- **Methodology**. On this chapter, the Quality Model will be described, justifying the selection of the different attributes to consider, the metrics derived from them and the weights applied taking into consideration the role selected.
- **Analysis**. Analysis chapter will justify the application of the model, which tools will be used in order to obtain and analyze the different representative metrics and the data sources used to retrieve them. A Goal-Question-Metric (GQM) model will be implemented to describe how OpenNebula project will be analyzed. Chapter final aim is to calculate OpenNebula score obtained by the application of the Quality Model by using the different tools and source data, all of them described on the previous sources.
- **Results**. In parallel to this document, and analysis of other Cloud Computing Open Source Projects have been performed with the same quality model. However, on those documents the particular weights applied are related to what the other students invented roles have considered to be appropriate. In this and those documents the metrics obtained will be shared. This chapter will perform the score of the rest of the projects, taking into account the same weights that this document have taken into consideration. With OpenNebula and the other Cloud Computing Open Source Projects score, the final decision will be taken.
- **Conclusion**. Conclusion chapter will justify if results and Open Source Project selected does really fit into requirements specified. It will also summarize the different aspects collected on this document.

- **Bibliography and References**. This chapter collects the references and bibliography followed to perform this analysis.

2.3 Introduction to the Quality Model

This document considers a Quality Model agreed between some students and a professor in order to have a common starting point. In section Methodology a complete description of the Quality Model will be performed. However, an introduction to it is appropriate to later justify the most important Quality Model Attributes that the role will consider.

The Quality Model Main Attributes agreed are described below:

1. **Efficiency**. From the perspective of performance of the software. E.g: Do benchmarks of the product exist?
2. **Documentation**. Quality of documentation of the project. E.g: Is the project documentation updated frequently?
3. **Functionality**. Quality of the project in terms of representative functionality. E.g: Does this project have a web browser to configure and administrate the resources under control?
4. **Professional Support**. Amount of companies providing professional support. E.g: How many companies provide professional support?
5. **Community**. Community Health as a way of measuring quality. E.g: Has the number of committers grown or decreased in the last year?

With previous quality attributes brought up, now an invention of the role, a description of it and a categorization of the most important attributes from her/his perspective can be performed.

2.4 Role Description

This document assumes a role play that allows to determine the more important factors from a quality perspective. In particular, the role of a member of the Board of Directors have been considered. So, the main question to answer is next one: **Which are the more relevant factors for this role?**

To answer this question, a first assumption of the personal background of this role must be performed. This member has been part of the leadership group of an important Open Source Project.

For this reason, his personal and professional background makes him aware of the **Importance of the Community** for an Open Source Project to succeed along the time.

Apart from being aware of the Community, this role considers that **Documentation** is as well an important factor, as it helps to strengthen and consolidate Community itself.

Meanwhile, role's professional background and experience also makes him/her to be aware of the **Importance of having professional support**.

Last, but not least, the board member considers that **having a full featured functionality and good efficiency/performance is not as important as the previous factors**, as having a strong community and a good professional support can help on improving both attributes in a relatively short period of time.

3 Methodology

In order to calculate what is the quality of a project, first of all the Quality Model must be defined. It is not the objective of this chapter to describe on a very detailed basis the different Software Quality Models and its characteristics. However, a very brief description and classification of them is required in order to evaluate them and select the most appropriate, or, as well, the one that can inspire the Final Quality Model.

There are many different type of Quality Models to evaluate Software Quality. Several classifications can be taken into consideration:

3.1 Software Generic Quality Models

There are Quality Models that have been designed to measure the Quality of the Software, or, at least, the process that is followed to produce software. Examples of this kind of Quality Models are:

- ISO 9126 [1, 2]. This model defines a set of six super-attributes that are considered the more relevant from Software Quality perspective:

  1. Functionality
  2. Reliability
  3. Usability
  4. Efficiency
  5. Maintainability

6. Portability

Each of the super-attributes defines a collection of sub-attributes as well. A collection of 27 sub-attributes derived from the super-attributes are defined. Taking into consideration all of the attributes and sub-attributes, a consideration on the quality of the Software of a particular project can be performed.

- CMMI [3]. Code Maturity Model Integration (CMMI) is applied more to the process of Software production rather than the Software itself. Defined in Carnegie Mellon University, this model claims that the model can be used to guide process improvement across a project, division, or an entire organization. It basically defines a set of levels that allow to identify the state of the development process up to day. The levels defined are as follow, taking into account that the higher the level is, the more quality the development process of the project has:

  - **Level 1. Initial.** Processes are unpredictable and unmeasurable, and no reactive.
  - **Level 2. Managed.** Processes are characterized for projects, and often reactive.
  - **Level 3. Defined.** Processes are characterized for the organization and proactive.
  - **Level 4. Quantitatively Managed.** Development Processes are Measured and Under Control.
  - **Level 5. Optimizing.** Focus on Process Improvement.

Both previously defined models, although being complete in terms of quality of the Software and its development process respectively, do not take into consideration additional aspects which are considered important only in the case of Open Source Software Projects. Examples of factors not being taken into consideration are, e.g.:

- Community commitment
- Community activity
- What kind of licensing is used? (Copyleft/Permissive)

Fortunately, there have been several approaches to define Quality Models which are more appropriate to be applied to an Open Source Project, and, which indeed, address previous factors, as well as other different ones that are also important to measure particular Quality aspects related to this kind of projects. Next section defines some of the most commonly used models.

3.2 Open Source Software Project Quality Models

As previously defined, some Open Source Software Project Quality Models have been defined. There are two basic kind of models, taking into account the complexity of its implementation:

- **Light-weight**: This kind of Quality Models are easy to accomplish. Examples of this kind of Quality Models are OpenBRR and QSoS:
  - **OpenBRR** [4]. Open Business Readiness Rating Quality model defines a Spreadsheet with a set of super-attributes. Each of the super-attributes (named Categories in the model) proposed by the model are:
    1. **Functionality**
    2. **Usability**
    3. **Quality**
    4. **Security**
    5. **Performance**
    6. **Scalability**
    7. **Architecture**
    8. **Support**
    9. **Documentation**
    10. **Adoption**
    11. **Community**
    12. **Professionalism**

    Each of the Categories define a set of subcategories (Metrics) with a description of the metric itself and the specification of the scores that apply to that particular metric. The user of the spreadsheet can assign different weights to both the category and the subcategories, in order to achieve a final score for the Open Source Project under study.

    The model gives, in the end, a Final Score for each project, what allows to the evaluator the possibility of quickly determining which is the most appropriate project according to her/his priorities in terms of Quality.

    This model, although being very improvable, has the merit of being the most simple one among the ones studied. For this reason, our Quality Model will be derived from an even more simple version of this model.

  - **QSOS** [5]. Qualification and Selection of Opensource Software is a Quality Model, licensed under GNU Free Documentation License.

The quality model defines what is named as the "QSOS Manifesto", which establishes a list of statements for the objectives that this Quality Model tries to define, that can be summarized in:

  * Analyzing needs and limitations in software adoption
  * Evaluating functional and technical requirements
  * Formalizing a methodology
  * Results reusing
  * Providing a free method

Based on this manifesto, QSOS defines an iterative process that consist of four independent steps:

  * **Definition**. Define a frame of references, with software families, types of licenses, types of communities, etc.
  * **Evaluation**. An Evaluation sheet is defined to evaluate the functional coverage and the risks that are to be considered from both user and service provider perspectives.
  * **Qualification**. This step define filters that take into account needs and limitations on the specific context of the project.
  * **Selection**. Identify software meeting the needs and/or requirements. Compare software and target selection.

To summarize, this model, despite the fact that is a light-weight model, is, somehow, a more difficult model to implement compared to OpenBRR. Apart from that, OpenBRR, having the possibility to assign different weights makes OpenBRR a more flexible model, more malleable to the necessities and priorities of a particular role or a particular company.

- **Heavy-weight**: This kind of Quality Models are complex and not easily implementable. A good example of this kind of project is QualOSS.

  - **QualOSS** [6]: Quality of Open Source Software. Started on 2006 and funded mainly by the European Union, this project aimed to fill the gap in the state of the art of the Quality Models for Open Source Project Evaluation.

    It is considered a **heavy-weight** Quality Model, due to the progress and change of methodology that supposed compared to the already existing Quality Models, described above.

    One of its first goals consisted of semi-full automation of the analysis of the projects and metrics obtaining, by retrieving them using tools such as FLOSS-Metrics [7], together with several scripts provided for optimizing this purpose.

As this document will not use or even inspire on QualOSS, not very deep information will be provided. However, last but not least, methodology used by this Quality Model is exposed below:

* **Initial Steps**: Consisted of interviews with companies and identification of priorities for them when selecting an Open Source Project. One of the main priorities identified were the Community, and concepts such as **Community Side**.
* **Basic Quality Model**: The Methodology followed a GQM (Goal - Question - Metric) basis, where each goal is divided into several questions, and several metrics are defined for each question.
* **Community Side**: The main quality attributes identified for the community side were **Size and Regeneration Adequacy**, on the one hand, as well as **Interactivity and Workload Adequacy** on the other hand.
* **Metrics**: The metrics retrieved should be as objective as possible. One of more metrics are matched with a given question, and always Metrics were result of an average among some data.

3.3 Details of the Selected Quality Model

However, a particular Quality Model is not mandatory to be applied "AS IS". Some variations, limitations, trims, extensions or further considerations can be performed, according to the requirements and priorities defined for a particular Open Source Project.

This document does not pretend to be an extensive report or investigation on Quality, but rather an explanation on what a definition of a simple Quality Model for a particular type of Open Source Projects, in this case, related to an incipient and very active technology as Cloud Computing is, and how a particular project scores to that model.

For previous reason, in this case, a very simple Quality Model has been developed, taking some of the attributes of OpenBRR on the one hand, and on the other hand implementing and extending other attributes that have been considered to be important for this kind of Cloud Computing Open Source Projects.

3.3.1 Quality Model Description

The model is inspired on OpenBRR, but with a set of Categories and metrics which differ from the original model. Apart from that, some concepts from ISO 9126 [1, 2] have been taken into account.

The set of Categories agreed for the Quality Model is described below:

- **Efficiency**
- **Documentation**
- **Functionality**
- **Professional Support**
- **Community**

Besides this, each of the categories, have a set of questions, with answers that are, somehow, measurable. Apart from that, there are several scores applicable, depending on the metrics obtained. Next tables define, for each category, the metrics and scores applicable.

Metric	**Description**	**Score Specification**	**Score**
Billing System	Does The Cloud System provide a Billing System for User Accounting?	Yes, fully supported	5
		Yes, partially supported	3
		No	1
Multi-Platform Support	Does The Cloud System work on top of different Operating Systems??	Yes, more than two	5
		Yes, two	3
		No	1
Administration Configuration System	Does the Cloud System provide a System (Web preferably) to help on Administration Tasks?	Yes, a web framework	5
		Yes, not web-based	3
		No	1
i18n	Does The Cloud System provide multi-language support	Yes, more than 5 languages	5
		Yes, from 1 to 5 languages	3
		No	1
Quota Management	Does The Cloud System provide a system for quota management	Yes, for computing, storage and networking	5
		Yes, but not for all functionalities	3
		No	1

Table 1: Functionality

Metric	**Description**	**Score Specification**	**Score**
Performance or Benchmark Tests Available	This measures if there was any performance testing done and benchmarks published typically in comparison to other equivalent solutions	Yes, with good results	5
		Yes	3
		No	1
Performance Tuning and Configuration	Is there documentation or tool to help fine-tune the component for performance?	Yes, extensive	5
		Yes, some	3
		No	1

Table 2: Efficiency

Metric	**Description**	**Score Specification**	**Score**
Company Support	Which is the amount of companies providing support ?	More than one	5
		Just one	3
		No one	1

Table 3: Support

Metric	**Description**	**Score Specification**	**Score**
Documentation Update	When was the last time that documentation was updated ?	Last week	5
		Last Month	4
		Last Three Months	3
		Last Year	2
		More than one year ago	1
Number of contributors to documentation	Amount of people who contributed to documentation last year	Ten or more people	5
		Five to ten people	4
		Two to Five people	3
		One person	2
		No person	1

Table 4: Documentation

Metric	**Description**	**Score Specification**	**Score**
Mean commits	Which was the mean number of commits per developer last month?	20 or more commits / developer	5
		10 to 20 commits / developer	4
		5 to 10 commits / developer	3
		1 to 5 commits / developer	2
		<1 commit / developer	1
File Territoriality	Percentage of files of the total modified by a unique developer	<5%	5
		5%-10%	4
		10%-20%	3
		20%-50%	2
		>50%	1
Community Growth	% of committers number increase/decrease in the last year, calculated as percent difference in the amount of people who committed changes on 2012 compared to 2013	>50%	5
		25% to 50%	4
		0% to 25%	3
		0% to -25%	2
		<-25%	1

Table 5: Community

The Quality Model Spreadsheet previously described can be downloaded at next repository:

`https://github.com/MSWL-PROJ-EV-2013-2014/projev.git`

In particular, the Quality Model Spreadsheet can be downloaded on path "spreadsheet/BRR_ProjEval_2013.ods
or directly through next path:

`https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/BRR_ProjEval_2013.ods`

3.3.2 Weights

Apart from the Quality Model Question / Metrics, the spreadsheet selected, inspired by OpenBRR Quality Model, contain the concept of "weights". Weights are simply a percentage to prioritize those aspects that are important for the role on her/his concept of quality.

On previous section Role Description, the priorities for the role selected was explained from a descriptive perspective. However, on this section, a quantitative specification will be provided.

As stated before, the role is aware of the **Importance of the Community** for an Open Source Project to succeed along the time. Apart from that, this role considers that **Documentation** is the next factor in terms of importance, as it helps to strengthen and consolidate Community itself.

Role's experience also makes the **Importance of having professional support** important enough to consider it next. Last, but not least, the board member considers that **having a full featured functionality and good efficiency/performance is not as important as the previous factors**.

However, although having not high priority, considers that efficiency has more impact on quality than functionality does. **Regarding metrics, no special concern is communicated from the role, so that all the metrics for each category seems important equally**.

According to the role's priorities, her/his preferences and concerns, the resulting table for Categories will be as the following one:

Category	Weight
Community	30%
Documentation	25%
Support	20%
Efficiency	15%
Functionality	10%

Table 6: Weights

Regarding metrics, no special consideration has been taken into. Depending on the number of metrics, proportionality will be applied, so that weight is proportional to all the metrics.

- For those categories having two metrics, 50% weight will be applied to the metrics.
- For those categories having three metrics, 33,33% weight will be applied to the metrics.
- And so on, and so forth.

3.3.3 Tools for Metrics Retrieval

Once Quality Model has been explained, it is time for an explanation on how to carry out the different methods to obtain the metrics associated to the quality attributes defined. Tools used to obtain the metrics have been basically three:

1. **OpenNebula web page**. To obtain information related to **Functionality** and **Professional Support**.
2. **Google**. To obtain information related to benchmarks comparing Cloud Computing Systems and, in this manner, obtain metrics related to **Efficiency**.
3. **cvsanaly** [8]. This tool, written in Python, allows obtaining and analyzing a particular software among different kind of them (CVS, as well as Subversion and GIT) and stores analyzed information on a SQL database. This information can be later analyzed in order to measure, for example:
   - Number of commits in a period of time, or a a particular time or date.
   - Most active committers.
   - And, in general, all the information that can be retrieved from that particular Software Configuration Manager.
   - Territoriality of the project, by means of calculation of the different people modifying a particular file.

- Type of files committed.

This kind of analysis fits perfectly to obtain metrics of the Quality Model associated to **Community**and **Documentation**, with metrics such as last update of documentation, mean number of commits per developer or files territoriality as percentage of files being touched by a unique developer.

3.3.4 Data Sources

Regarding sources of data used to obtain the metrics, next datasets have been used:

- **Documentation associated to Key Features on OpenNebula project** [9]. Through this documentation, all the characteristics, functionalities and different features provided by the main component have been found out. It has been used to find out attributes having to do with **Functionality** category, such as if Billing System exist, if there is a configuration tool for administration, etc.
- **Documentation associated to Commercial Support** [10]. This documentation has helped on finding out attributes having to do with **Support** category, clarifying the amount of companies providing professional support.
- **Cloud Stack Benchmark comparison** [11]. This documentation has helped on finding out attributes having to do with **Efficiency** category, clarifying if Benchmark studies exist.

4 Analysis

This section summarizes the score obtained by OpenNebula Open Source project on the previously proposed Quality Model. Each of the next subsections cover the application of the Quality Model, and how by using the tools described on the data sources, an score is obtained for each category. Later, a final score will be obtained by setting weights to each metric and each category, to obtain the final score for OpenNebula.

Spreadsheet covering all this information, weights and scores, can be checked on next URL:

`https://github.com/MSWL-PROJ-EV-2013-2014/projev.git`

In particular, the Quality Model Spreadsheet can be downloaded on path "spreadsheet/OpenNebula/OpenNebu
or directly through next path:

`https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/OpenNebula/`
`OpenNebula_BRR_ProjEval_2013.ods`

4.1 Functionality

This chapter focuses on Functionality category metrics. It must be remarked that, for this category, following parameters have been considered:

- **Tools:** This category has been filled out basically through information existing on OpenNebula Open Source Project Web. Just a web browser (Firefox in this case) has been used.
- **Data Sources:** Key Feature chapter on OpenNebula Open Source Project [9].
- **Category Weight:** As previously stated, for this category it has been considered a weight of 10%.

Taking into account previous data, next table shows the final score of this category.

Metric	**Score Obtained**	**Score**	**Metric Weight**	**Metric Score**
Billing System	Yes, fully supported	5	20%	1
Multi-Platform Support	No	1	20%	0.2
Administration Configuration System	Yes, a web framework	5	20%	1
i18n	Yes, more than 5 languages	5	20%	1
Quota Management	Yes, for computing, storage and networking	5	20%	1
Unweighted rating	**4.2**			
Weighted rating (10%)	**0.42**			

Table 7: Functionality Score

4.2 Efficiency

Efficiency category metrics are handled in this section. For this category, it must be remarked that following parameters have been considered:

- **Tools:** This category has been filled out basically through information existing on Benchmark studies about OpenNebula. Google, DuckDuckGo and a web browser (Firefox in this case) have been used.

- **Data Sources:** Performance comparison between OpenStack and OpenNebula [11].
- **Category Weight:** As previously stated, for this category it has been considered a weight of 15%.

Taking into account previous data, next table shows the final score of this category.

Metric	**Score Obtained**	**Score**	**Metric Weight**	**Metric Score**
Performance Testing and Benchmark Reports available	Yes	3	50%	1.5
Performance Tuning and Configuration	No	1	50%	0.5
Unweighted rating	**2**			
Weighted rating (15%)	**0.3**			

Table 8: Efficiency Score

4.3 Support

Support category metrics is studied in this chapter. For this category, it must be remarked that following parameters have been considered:

- **Tools:** This category has been filled out basically through information existing on OpenNebula Open Source Project Web. Just a web browser (Firefox in this case) has been used.
- **Data Sources:** OpenNebula information associated to Commercial Support [10].
- **Category Weight:** As previously stated, for this category it has been considered a weight of 20%.

Taking into account previous data, next table shows the final score of this category.

Metric	**Score Obtained**	**Score**	**Metric Weight**	**Metric Score**
Company Support	Just one	3	100%	3
Unweighted rating	**3**			
Weighted rating (20%)	**0.6**			

Table 9: Support Score

4.4 Documentation

Documentation category metrics are analyzed in this chapter, to measure the quality of the documentation, considered important from the invented role perspective. For this category, it must be remarked that following parameters have been considered:

- **Data Sources:** OpenNebula main source code repository, **One**, has been used to retrieve and analyze metrics.
- **Tools:** Tools used for this category have been basically **Git**, **cvsanaly2** and **MySQL Server**, to retrieve source code repository, classify it and analyze it respectively.
  - Git. Git has been used to retrieve OpenNebula source code from previously described repository. The command issue for retrieval has been:
    ```
    # git clone https://github.com/OpenNebula/one.git
    ```
    Previous command will create a directory, "one", where OpenNebula source code can be classified and analyzed.
  - cvsanaly2. cvsanaly2 tool has been used to classify source code existing on previous directory. Next command assumes that an empty database, called "cvsanaly_one", already exists. Creation of a database with MySQL will be described later, on MySQL tool usage. cvsanaly2 command issued to perform source code classification and storage has been:
    ```
    # cvsanaly2 --db-user=root --db-password=whatever --db-database=cvsanaly_one
    --extensions=FileTypes
    ```
    It is remarkable that cvsanaly2 provides extensions, which are in the end plugins that allow to expand the possibilities of analysis. Extension "FileTypes" have been included, as this extension allow to classify the type of files by its extension.
  - MySQL. MySQL server is an important tool to calculate this metrics as well. On the one hand, "cvsanaly_one" table has been created, through this commands:
    ```
    # mysql -uroot -pwhatever
    ```

Previous command gives access to MySQL server prompt. Next command allows creation of the database needed by cvsanaly2 to store the information appropriately sorted in different tables.

```
mysql> create database cvsanaly_one;
```

Once the tables have been created and filled by "cvsanaly2" tool, queries must be performed in order to analyze the metrics needed for this category. It is not the objective of this document to include a deep study of the tables that cvsanaly.

Queries used and information dumped by MySQL prompt for each of the metrics is shown below:

```
# mysql -uroot -pwhatever
```

Previous command gives access to MySQL prompt. In order to use the database, next command has to be issued:

```
# use cvsanaly_one
```

To obtain when was documentation updated last time:

```
mysql> select MAX(s.date) from files ff, scmlog s, file_types f, actions
a where s.id=a.commit_id and f.file_id=a.file_id AND f.type='documentation'
AND ff.file_name LIKE '%.txt' ORDER BY s.date;
```

This query gives a resulting date equal to:

```
+---------------------+
| MAX(s.date)         |
+---------------------+
| 2013-11-28 15:11:52 |
+---------------------+
1 row in set (1.74 sec)
```

Taking into account that the query was performed on 27th of December, 2013, resulting score is "Last Month". Meanwhile, to obtain how many people contributed to documentation on last year, next query has been used:

```
mysql> select DISTINCT(p.email) from files ff, scmlog s, file_types
f, people p, actions a where s.id=a.commit_id and f.file_id=a.file_id
AND f.type='documentation' AND ff.file_name LIKE '%.txt' AND YEAR(s.date)='2013'
AND s.committer_id=p.id;
```

This query gives a resulting number equal to:

```
+--------------------------+
| email                    |
+--------------------------+
| jmelis@opennebula.org    |
| dmolina@opennebula.org   |
| tinova@opennebula.org    |
| rsmontero@opennebula.org |
| cmartin@opennebula.org   |
| jfontan@opennebula.org   |
+--------------------------+
6 rows in set (3.32 sec)
```

Resulting score is, as demonstrated above, "5 to 10 people", and, in particular, 6.

- **Category Weight:** As previously stated, for this category it has been considered a weight of 25%.

Taking into account previous data, next table shows the final score of this category.

Metric	**Score Obtained**	**Score**	**Metric Weight**	**Metric Score**
Documentation Update	Documentation updated last month	4	50%	2
People Contributing Documentation on last year	5 to 10 people	4	50%	2
Unweighted rating	**4**			
Weighted rating (25%)	**1**			

Table 10: Documentation Score

4.5 Community

Last, but not least, Community category metrics are analyzed to measure the stability and community health. Nevertheless, this category has been considered the most important one. For this category, it must be remarked that following parameters have been considered:

- **Data Sources:** OpenNebula main source code repository, **One**, has been used to retrieve and analyze metrics.
- **Tools:** Tools used for this category have been basically **Git**, **cvsanaly2** and **MySQL Server**, to retrieve source code repository, classify it and analyze it respectively.
  - Git. Git has been used to retrieve OpenNebula source code from previously described repository. Usage is exactly the same as described in Documentation.

– cvsanaly2. cvsanaly2 tool has been used to classify source code existing on previous directory. Usage is exactly the same as described in Documentation.

– MySQL. MySQL server is an important tool to calculate this metrics as well. Apart from database creation, described on previous section Documentation, MySQL has been used in order to obtain the metrics need to score the project on this category. Queries used and information dumped by MySQL prompt for each of the metrics is shown below:

  ```
  # mysql -uroot -pwhatever
  ```

  Previous command gives access to MySQL prompt. In order to use the database, next command has to be issued:

  ```
  # use cvsanaly_one
  ```

  To obtain the number of committers per developer on last month (last complete month was November), first the total number of committers must be issued:

  ```
  mysql> select COUNT(DISTINCT(p.email)) as ''Number of Commiters'' from
  scmlog s, people p WHERE YEAR(s.date)='2013' AND MONTH(s.date)='11'
  AND s.committer_id=p.id;
  ```

  This query gives a resulting number equal to:

  ```
  +---------------------+
  | Number of Commiters |
  +---------------------+
  |                   6 |
  +---------------------+
  1 row in set (0.11 sec)
  ```

  On the other hand, the total number of commits must be calculated. In particular, it can be done with next query:

  ```
  mysql> select COUNT(s.id) as ''Number of Commits'' from scmlog s, people
  p WHERE YEAR(s.date)='2013' AND MONTH(s.date)='11' AND s.committer_id=p.id;
  ```

  This query gives a resulting number equal to:

  ```
  +-------------------+
  | Number of Commits |
  +-------------------+
  |               315 |
  +-------------------+
  1 row in set (0.04 sec)
  ```

  Resulting score is 315/6 = 52.5, which is in the end "20 or more commits / developer". Meanwhile, to obtain the territoriality of files, first the total number of files must be calculated:

```
mysql> SELECT COUNT(DISTINCT(file_path)) AS ‘‘File Number’’ FROM file_links;
```
This query gives a resulting number equal to:

```
+-------------+
| File Number |
+-------------+
|        4215 |
+-------------+
1 row in set (3.05 sec)
```

On the other hand, the number of people who modified just one file can be calculated with next command:

```
mysql> SELECT f.file_path, COUNT(p.email) AS ‘‘People touching file’’
FROM scmlog as s, people as p, file_links as f WHERE f.commit_id=s.id
AND s.committer_id=p.id GROUP BY f.file_path HAVING COUNT(p.email)=1;
```
This query gives huge result with all the files being touched by just one person, giving a resulting number equal to:

```
+--------------------------------+----------------------+
| file_path                      | People touching file |
+--------------------------------+----------------------+
| apptools                       |                    1 |
| apptools/install.sh            |                    1 |
| ...                            |                  ... |
| sunstone/routes/appenv.rb      |                    1 |
| sunstone/routes                |                    1 |
+--------------------------------+----------------------+
2230 rows in set (0.00 sec)
```

Resulting score is, as demonstrated above 2230/4215 = 0.529, which gives an score of “more than 50% of the files”. Last, but not least, it must be calculated how community has grown or abated on last year. To do so, the number of committers on 2012 has been calculated with next query:

```
mysql> select DISTINCT(p.email) as ‘‘Number of Commits’’ from scmlog
s, people p WHERE YEAR(s.date)=’2012’ AND s.committer_id=p.id;
```
This query gives a resulting number equal to:

```
+--------------------------+
| Committers               |
+--------------------------+
| rubensm@dacya.ucm.es     |
| j.melis@fdi.ucm.es        |
| jfontan@gmail.com         |
| cmartins@fdi.ucm.es       |
| tinova79@gmail.com        |
| danmolin@fdi.ucm.es       |
| support@c12g.com          |
| hsanjuan@opennebula.org   |
| rsmontero@opennebula.org |
| ruben@pc-ruben.(none)     |
| jmelis@opennebula.org     |
| jfontan@opennebula.org    |
| tinova@opennebula.org     |
| dmolina@opennebula.org    |
| cmartin@opennebula.org    |
| hector@convivencial.org   |
+--------------------------+
16 rows in set (0.03 sec)
```

Later, the number of commiters in 2013 has been calculated by issuing next query:

```
mysql> select DISTINCT(p.email) as ''Number of Commits'' from scmlog
s, people p WHERE YEAR(s.date)='2013' AND s.committer_id=p.id;
```

This query gives a resulting number equal to:

```
+--------------------------+
| Committers               |
+--------------------------+
| jfontan@gmail.com         |
| tinova79@gmail.com        |
| rsmontero@opennebula.org |
| jmelis@opennebula.org     |
| jfontan@opennebula.org    |
| tinova@opennebula.org     |
| dmolina@opennebula.org    |
| cmartin@opennebula.org    |
| dmolina@c12g.com          |
+--------------------------+
9 rows in set (0.04 sec)
```

This information is not enough, as, on this small number of results, it can be observed that some committers are repeated. Indeed, for each table, it can be asserted than on 2012 there were 9 unique committers. Meanwhile, on 2013, there were 6 unique committers. After fixing the results, it can be ensured that resulting score is, as demonstrated above, is a decrease of 3/9 =0.33 (33%), what means a "decrease of more than 25%".

- **Category Weight:** As previously stated, for this category it has been considered a weight of 30%.

Taking into account previous data, next table shows the final score of this category.

Metric	**Score Obtained**	**Score**	**Metric Weight**	**Metric Score**
Mean commits / developer last month	20 or more / developer	5	33%	1.67
Percent of files only modified by one developer	>50%	1	33%	0.33
Community growth	<-25%	1	33%	0.33
Unweighted rating	**2.33**			
Weighted rating (30%)	**0.7**			

Table 11: Community Score

4.6 Final Score

With data existing on the previous tables, where partial scores for each category and weighted rating have been calculated, give the possibility to calculate the final total score for OpenNebula.

Final score is just calculated by adding the weighted rating of each of the category, what gives a final result of 3.02, as shown summarized in next table:

Category	**Unweighted rating**	**Weight**	**Weighted Score**
Functionality	4.2	10%	0.42
Efficiency	2	15%	0.3
Support	3	20%	0.6
Documentation	4	25%	1
Community	2.33	30%	0.7
Final Score	**3.02**		

Table 12: OpenNebula Score

Once the Final Score for OpenNebula has been obtained, one question appears: **Is OpenNebula a good quality project? There is no answer to this question**. OpenBRR does not define to which level a project is a quality project, as it just allows comparing between projects of the same nature for a evaluator to take a final decision.

On next section, taking advantage of the scores obtained to the same Quality Model from other Cloud Computing Open Source Projects, in particular CloudStack, Eucalyptus and OpenStack, and adjusting the weights of them to be the the same as the ones used for calculating OpenNebula score, a calculation of the best project fitting the necessities of the role will be performed.

5 Results

On previous section Analysis, the calculation of the quality score of OpenNebula Open Source Project has been carried out. However, this score is only valid to compare the project against other similar nature projects in order to take a decision on which project fits better for the necessities identified by the invented role described in subsection Role Description.

In parallel to the redaction of this final assignment, other lectures have performed the evaluation of CloudStack, Eucalyptus and OpenStack Open Source Projects. Spreadsheets developed by them can be obtained on next repository:
`https://github.com/MSWL-PROJ-EV-2013-2014/projev.git`

In particular, the Quality Model Spreadsheets can be downloaded, respectively, on next paths:

- "spreadsheet/CloudStack/CloudStack_BRR_ProjEval_2013.ods"
- "spreadsheet/eucalyptus/BRR_ProjEval_eucalyptus.ods"
- "spreadsheet/OpenStack/OpenStack_BRR_ProjEval_2013.ods"

or directly through next urls:

- `https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/CloudStack/CloudStack_BRR_ProjEval_2013.ods`
- `https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/eucalyptus/BRR_ProjEval_eucalyptus.ods`
- `https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/OpenStack/OpenStack_BRR_ProjEval_2013.ods`

Previous models contain an analysis of the rest of the Cloud Computing Open Source Projects, similar to the one performed . However, they contain weights adapted to what other students' invented roles have considered appropriate.

For this reason, the models have been reused in order to calculate the scores of each of these other projects, but taking into account the invented role described previously on section Role Description, which, in turn, means considering the weights described in Weights.

The score of each of the rest of the projects is shown in next subsections, on a summarized way similar to the Final Score described in subsection. Besides this, a comparison table showing a top down projects classification, always considering the Quality Model proposed in subsection Details of the Selected Quality Model.

It must be remarked that information existing on next subsections can also be obtained through next repository:
`https://github.com/MSWL-PROJ-EV-2013-2014/projev.git`

In particular, the Quality Model Spreadsheets can be downloaded, respectively, on next paths:

- "spreadsheet/OpenNebula/CloudStack_BRR_ProjEval_2013.ods"
- "spreadsheet/OpenNebula/BRR_ProjEval_eucalyptus.ods"
- "spreadsheet/OpenNebula/OpenStack_BRR_ProjEval_2013.ods"

or directly through next urls:

- https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/OpenNebula/CloudStack_BRR_ProjEval_2013.ods
- https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/OpenNebula/BRR_ProjEval_eucalyptus.ods
- https://github.com/MSWL-PROJ-EV-2013-2014/projev/blob/master/spreadsheet/OpenNebula/OpenStack_BRR_ProjEval_2013.ods

  The reason for each of the projects to be contained in "OpenNebula" folder is simply that they follow the weights proposed for OpenNebula project.

5.1 CloudStack

This section contemplates the final score of CloudStack Open Source Project. Final score is just calculated by adding the weighted rating of each of the categories, what gives a final result of 3.571, as shown summarized in next table:

Category	**Unweighted rating**	**Weight**	**Weighted Score**
Functionality	3.8	10%	0.38
Efficiency	3	15%	0.45
Support	3	20%	0.6
Documentation	5	25%	1.25
Community	2.97	30%	0.891
Final Score	**3.571**		

Table 13: CloudStack Score

5.2 Eucalyptus

This section contemplates the final score of Eucalyptus Open Source Project. Final score is just calculated by adding the weighted rating of each of the categories, what gives a final result of 3.43, as shown summarized in next table:

Category	**Unweighted rating**	**Weight**	**Weighted Score**
Functionality	4.6	10%	0.46
Efficiency	3	15%	0.45
Support	3	20%	0.6
Documentation	4.5	25%	1.13
Community	2.64	30%	0.79
Final Score	**3.43**		

Table 14: Eucalyptus Score

5.3 OpenStack

Finally, the final score of OpenStack Open Source Project. Final score is just calculated by adding the weighted rating of each of the categories, what gives a final result of 3.31, as shown summarized in next table:

Category	**Unweighted rating**	**Weight**	**Weighted Score**
Functionality	3.4	10%	0.34
Efficiency	3	15%	0.45
Support	5	20%	1
Documentation	4.5	25%	1.13
Community	1.32	30%	0.4
Final Score	**3.31**		

Table 15: OpenStack Score

5.4 Cloud Computing Open Source Projects Comparison

Last, but not least, it is important to set a top-down table, ordered by weighted final score, with the Quality Model decided. Next table shows project classification depending on the Final Score, as well as the weighted score obtained for each category:

Project	**Final Score**	Functionality	Efficiency	Support	Documentation	Community
CloudStack	**3.57**	0.38	0.45	0.6	1.25	0.891
Eucalyptus	**3.43**	0.46	0.45	0.6	1.13	0.79
OpenStack	**3.31**	0.34	0.45	1	1.13	0.4
OpenNebula	**3.02**	0.42	0.3	0.6	1	0.7
Winner Project	**CloudStack**					

Table 16: Comparison Table

For this reason, it can be ensured that, according to the Quality Model and the weights specified, **CloudStack is the most appropriate Cloud Computing Open Source Project**, at least among the set of projects which have been analyzed.

6 Conclusion

Contrary to what may appear initially, where OpenStack seems to be the most important Cloud Stack Computing Open Source Project, the final conclusion of this document is clear: **CloudStack is the most appropriate Cloud Computing Open Source Project**, at least, for a self-defined Quality Model and taking into account those factors considered more important for an invented role acting as evaluator. However, special attention has to be paid before taking the final decision. At least, some inspection on the results of the weighted scores for each category against this project competitors must be analyzed.

At a first glance, it could happen that the project was much better in less important categories, such as Functionality, Efficiency or even Support, compared to CloudStack competitors, but nothing could be further from the truth. CloudStack is the **best project** in what has been determined the most important categories for the invented role, namely, **Documentation and Community**.

Besides this, on the rest of categories, CloudStack is never the worst case, at least on a unique way. For Functionality category, CloudStack is, at least, better than OpenStack, although worst than OpenNebula and Eucalyptus. Meanwhile, for Efficiency category, CloudStack equals in first place with Eucalyptus and OpenStack, improving OpenNebula. Support category is not the strength of CloudStack, but at least it seems that its score equals Eucalyptus and OpenNebula.

Considering previous information, there is no doubt that **CloudStack Open Source Project is the one fitting best the requirements of the evaluator**.

This document is an entry point to the different steps that must be followed in order to determine the best option when selecting a particular Open Source project, and in particular:

1. **Identifying the set of candidate projects.** Select a group of Open Source Projects among the different possibilities by considering the nature of the product that must be acquired, together with the information existing on specialized magazines, scientific papers, blogs and other Internet information sources in order to build a set of candidate projects to be evaluated.

2. **Determining a Quality Model.** It can be based on existing ones, as described in section Open Source Software Project Quality Models, or a self-defined/adapted version to fit better into the requirements of the company. It is recommended to set a Quality Model based on GQM (Goal/Question/Metric) model, by setting Goals, raising Question and obtaining Metrics that allow answering them.

3. **Weighting the different categories.** Depending on the type of Quality Model selected, it could be possible that a classification in importance of the different categories existing on the model must be carried out, as described in subsection Weights.

4. **Analyzing and Classifying candidate projects.** Using a particular set of tools, applied to different sets of data sources, an analysis of the candidate projects should be performed. This analysis must determine as result a final top-down classification of the different projects, for the evaluator to select the most appropriate project for the necessities proposed.

5. **Final decision.** Before taking the final decision, some other questions must be asked: Is the project a good option for my necessities? Does the project have special weakness compared to other projects? And if so, can they suppose a problem for the company/organization?. Are there projects that, although being slightly worst in the Quality Model score result, are, somehow, more balanced, having fewer weaknesses?

Facing the decision of which Open Source Project must be acquired for a particular company, organization, government institution or for any other enterprise is normally a hard thing. This document aims to be an entry point to follow some very basic steps for facilitating this task, and minimize the risk associated to making a bad choice.

References

[1] ISO/IEC. ISO/IEC 9126. Software engineering - Product quality, 2001.

[2] X. Franch and J. P. Carvallo. Using quality models in software package selection. *IEEE*, 2003.

[3] CMMI Product Team. CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033), 2010.

[4] SpikeSource Carnegie Mellon West Center for Open Source Investigation, CodeZoo and Intel. Open Business Readiness Rating, 2005.

[5] Raphal Semeteys at Atos Origin. Qualification and Selection of Opensource Software, 2004.

[6] Research Project Promoted by the European Union. QUALOSS: Quality of Open Source Software, 2006-2009.

[7] FLOSS Metrics Consortium. Free/Libre and OpenSource Software Metrics, 2010.

[8] Metrics-Grimoire. Cvsanaly, 2009.

[9] OpenNebula. Opennebula 4.4 version key features, 2013.

[10] OpenNebula. Opennebula support and professional services, 2013.

[11] J. Rouzaud-Cornabas E. Caron, L. Toch. Comparaison de performance entre Open-Stack et OpenNebula et les architectures multi-Cloud: Application la cosmologie, 2013.