Universidad Rey Juan Carlos

Master in Libre Software

FℓOSS Project Evaluation

Software Quality Evaluation of FℓOSS Cloud Project: OpenStack

Author:
Daniel H.Gámez V.

Lecturer:
Daniel Izquierdo

January 20, 2014

CC BY-SA 3.0

http://creativecommons.org/licenses/by-sa/3.0/legalcode

Abstract

With the advent of Cloud Computing technology nowadays and the high demand for resources and services hosted on the Internet, arises the need to implement solutions based on this scheme, taking full advantage of available resources and getting the most profit of the services provided.

In particular this analysis will focus on the selection of a FℓOSS[1] Cloud Computing project, considering that today there are several alternatives in this area, taking into account factors such as the available professional support, functionality, vitality, longevity, quality and other important elements on the project.

The proper evaluation of existing projects in this field is a critical task for IT managers, among other things because to the costs of selecting the wrong technology could be later unapproachable. Must be considered functional, technical and social requirements to get the right decision.

To make this analysis, the Open Business Readiness Rating (OpenBRR) methodology is going to be used, a light-weight method to evaluate FℓOSS projects. Defining categories and asigning weights to associated metrics, this model offers a rating system with which the selection of a software can be made easier.

Initially the main cloud computing solution that is going to be evaluated is OpenStack, and then it will be compared with other FℓOSS competitive solutions available in the market. Results are going to be shown and discussed in order to provide a winning option.

[1] Free Libre Open Source Software

Contents

1 Introduction

In this report will be analysed a FℓOSS project that leads the Cloud Computing field in today's market, this is OpenStack[2], and will be compared with other similar powerful solutions that directly compete with this, such as CloudStack[3], Eucalyptus[4] and OpenNebula[5].

As an initial assessment, Business Readiness Rating (OpenBRR) methodology is going to be used, proposed as an open standard light-weight method developed by Carnegie Mellon West Center, CodeZoo, SpikeSource and Intel, to evaluate FℓOSS projects.

Different approaches to evaluate software exists. At least two previous initiatives "Open Source Maturity Model" (Bernard Golden) and "CapGemini Open Source Maturity Model" (CapGemini) [Duijnhouwer, 2003], which aim to provide open source adopters a methodology for assessing and evaluating the suitability of open source software. However, new models like OpenBRR use similar concepts for a thorough software evaluation, with more detailed evaluation of the data and scoring to assess the software's business readiness, especially for operational and support aspects, it provides a scientific light-weight model for a rating system that contains a clear mapping from evaluation data to scoring, and then the resulting final rating. The goal of this methodology is among others, is to produce an standard for software assessment that is widely adoptable and easily usable in as many evaluation situations as possible.

The calculation employed in OpenBRR weights the factors that have proven to be most important for successful deployment of open source software in specific environments. Among these are functionality, quality, performance, support, community size, security, and others. In [Figure 1] are shown the four phases of software assessment in OpenBRR, which are:

- Quick assessment: to rule in or rule out software packages and create a shortlist of viable candidates.
- Target usage assessment: ranking the importance of categories or metrics.
- Data collection and processing: for data manipulation.
- Data translation: translating the data into the Business Readiness Rating.

[2] http://www.openstack.org/
[3] http://cloudstack.apache.org/
[4] https://www.eucalyptus.com/
[5] http://opennebula.org/

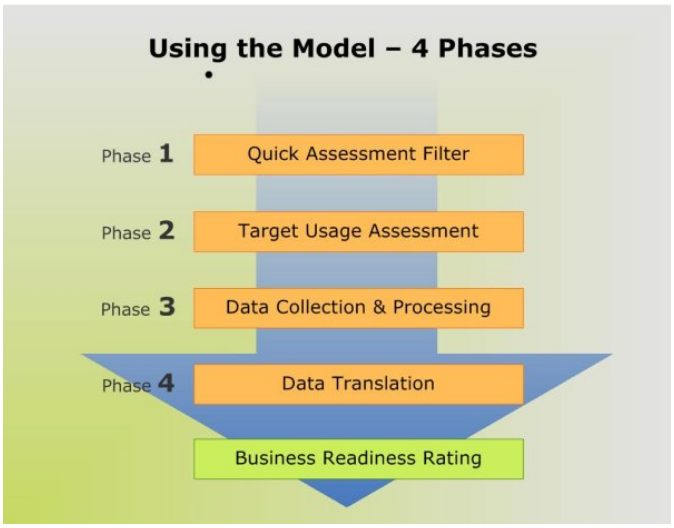


Figure 1: OpenBRR phases

In a more detailed granularity, in OpenBRR a software component is scored with a value from 1 to 5, with 1 being "Un-acceptable" and 5 being "Excellent", as shown in [Figure 2].

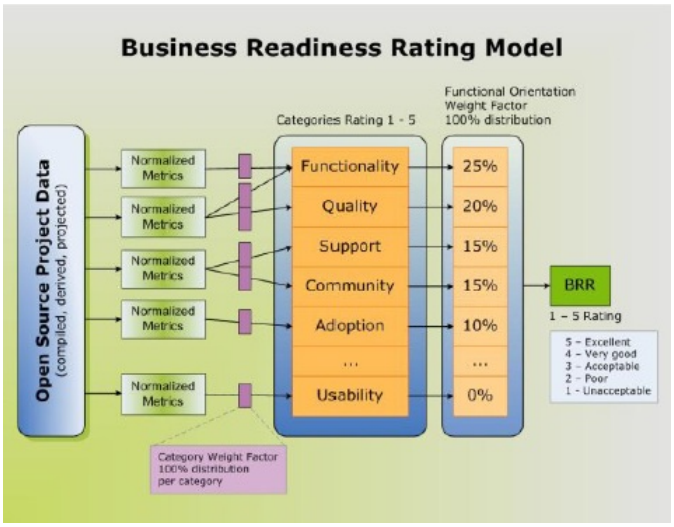


Figure 2: OpenBRR model

Finally, the categories (in alphabetic order) and corresponding metrics selected in common were:

Category	Metrics
Community	- Mean commits per developer last month - Percent of files touched by only one developer - Community growth (commit number variation last year)
Documentation	- Documentation update frecuency - Number of contributors to documentation
Efficiency	- Performance testing and benchmark reports available - Performance tuning and configuration available
Functionality	- Billing System - Multi platform suport - Administrators configuration system - i18n - Quota facilities
Support	- Number of companies providing professional support

2 Methodology

The aim of this section is to provide a way to replicate the current study, to show used tools and to formally reference the data repositories consulted.

2.1 Quick Assessment

In this phase was used a Goal-Question-Metrics aproach [Basili et. al., 1994], in which (1) goals must focus on an object, aimed to certain purposes, examining the object from one or different points of view and within a certain context, (2) questions break down one or several roles into concrete issues to be inspected, and (3) metrics define measurable quantities or logical aspects that provide information to offer precise answers and assessment in response to the set of questions.

Here basically was necessary to:

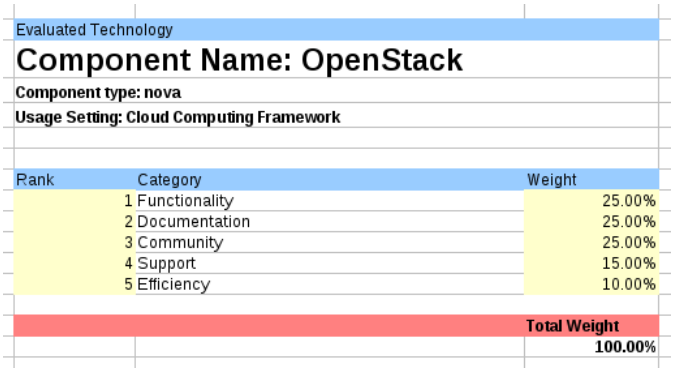
- Identify a list of components to be evaluated.
- Measure each component against the quick assessment criteria.
- Remove any components that do not satisfy user requirements from the list.

The role assumed in this analysis was from the System Administrator standpoint, being this a well-defined role at present, with specific needs and concrete tasks within an organization.

2.2 Target Usage Assessment

The presented OpenBRR Quality Model depends on a spreadsheet template which facilitates the calculation of cuantitative rankings. For this purpose, was necessary to define the category and metric weights in the following way.

- **Category weights:**
  - Rank the 5 categories according to importance (1 for highest, 5 for lowest).
  - Assign a percentage of importance for each, totaling 100% over the chosen categories (as shown in [Figure 3]).



Evaluated Technology		
Component Name: OpenStack		
Component type: nova		
Usage Setting: Cloud Computing Framework		
Rank	Category	Weight
1	Functionality	25.00%
2	Documentation	25.00%
3	Community	25.00%
4	Support	15.00%
5	Efficiency	10.00%
	Total Weight	
		100.00%

Figure 3: OpenStack category weights

- **Metric weights:**
  - For each metric within a category, assign a percentage of importance, totaling 100% over all the metrics within one category.
  - For each metric within a category, rank the metric according to importance to business readiness.

    Functionality (see [Figure 4]): a set of attributes which provide functions and properties that meet the operational requirements of the tool.

Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Billing System	Yes	20.00%	5	1
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Multi Platform Suport	No	15.00%	1	0.15
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Administrators Configuration System	Yes	30.00%	5	1.5
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
i18n	No	10.00%	1	0.1
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Quota Facilities	Yes	25.00%	5	1.25

Figure 4: Functionality metrics

Documentation (see [Figure 5]): existance of varios kinds of documentation, for different user groups, available in multiple formats. A users contribution framework is also necessary.

Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Documentation Update	Yes	55.00%	4	2.2
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Number of contributors to documentation	10 or more people	45.00%	5	2.25

Figure 5: Documentation metrics

Community (see [Figure 6]): number of commits and developers, and usual patterns in the community.

Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Mean commits / developer last month	5 to 10 commits / developer	30.00%	3	0.9
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Percent of files touched by only one developer	> 50%	30.00%	1	0.3
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Community growth (commit number variation)	0 to 25 %	40.00%	3	1.2

Figure 6: Community metrics

Support (see [Figure 7]): professional support that helps fine-tune for the local deployment and troubleshooting is always desirable.

Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Company Support	More than one company providing support	100.00%	5	5

Figure 7: Support metrics

Efficiency (see [Figure 8]): Measures if there is any performance testing done and benchmarks published, typically in comparison to other equivalent so-

lutions. Also if there is any documentation or tool to help fine-tune the component for performance.

Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Performance Testing and Benchmark Reports available	Yes	30.00%	3	0.9
Metric Name	Raw Score	Weight	Unweighted Score	Weighted Score
Performance Tuning & Configuration	Yes, some	70.00%	3	2.1

Figure 8: Efficiency metrics

2.3 Data Collection and Processing

Here is necessary to gather data for each metric used in each category rating, and calculate the applied weighting for each metric. Data sources used for this purpose were:

Functionality	
Billing system	- https://wiki.openstack.org/wiki/SystemUsageData - https://wiki.openstack.org/wiki/QuantumUsage
Multi platform suport	- http://www.openstack.org/software/start/
Admins configuration system	- http://www.openstack.org/software/openstack-dashboard/
i18n	- https://wiki.openstack.org/wiki/Translations - https://www.transifex.com/projects/p/openstack/
Quota facilities	- http://docs.openstack.org/trunk/openstack-ops/content/quotas.html - http://docs.openstack.org/userguide-admin/content/cli_set_quotas.html

Documentation	
Documentation update frecuency	OpenStack Git 'nova' repository: - https://github.com/openstack/nova.git

Community	
Commits per developer	- https://github.com/openstack/nova.git
Percent of files touched by only one developer	- https://github.com/openstack/nova.git
Community growth (commit number variation in the last year)	- https://github.com/openstack/nova.git

Support	
Companies providing professional support	- http://www.mirantis.com/openstack-services/support/ - http://www.rackspace.com/cloud/openstack/ - http://support.rackspace.com/

Efficiency	
Performance testing and benchmark reports	- http://hal.inria.fr/docs/00/91/69/08/PDF/RR-8421.pdf - http://www.openstack.org/summit/openstack-summit-hong-kong-2013/session-videos/presentation/benchmarking-openstack-performance-at-scale - http://thoughtsoncloud.com/index.php/2013/12/benchmarking-and-scaling-openstack-in-the-enterprise/ - http://www.mirantis.com/blog/openstack-swift-configurations-performance-benchmarking/
Performance tuning and configuration	- http://www.openstack.org/assets/presentation-media/openstackperformance-v4.pdf - http://www.slideshare.net/openstackindia/openstack-nova-and-kvm-optimisation

2.4 Data Translation

In this phase are used the category ratings and the functional orientation weighting factors to calculate the OpenBRR score, basically through a summarization of the results for each category.

Each metric has an associated Weighted Score which is calculated by multiplying the assigned Weigth (a percentage) by the Unweighted Score obtained after the evaluation. Then all the Weighted Scores are summarised to calculate the Unweighted Rating. Subsequently, the Weighted Rating corresponds to a multiplication of each Unweighted

Rating by the corresponding category Weight (a percentage). And finally the total OpenBRR score is a summarization of Weighted Ratings from all categories.

Then the software's OpenBRR score should be published, this way users are able to feed their evaluations back into the open source community. Thereby may be taken decisions regarding the choice of specific software based on specific quantitative results. Although either way, there should be a rational interpretation of the results.

2.5 Tools used in the process

On the one hand, in the early stages of the method, are used internet searches directly on the official websites of each technology, as well as a formal documentation available for them. However, going forward in the process, it is necessary to use specialized tools that allow more detailed analysis of data sources. Among the specialized tools used are the following:

- Repository Handler: a FℓOSS (under GNU General Public License version 2) Python library for handling code repositories through a common interface. It allows the use of CVSanalY tool.
  Available in github.com as: https://github.com/MetricsGrimoire/RepositoryHandler.

- CVSanalY: a FℓOSS (under GNU General Public License version 2) tool to analyze software repositories [Garcia, 2009]. It extracts information out of source code repository logs and stores it into a database. It is written in Python and for the purposes of this analysis uses the standard Python Distutils, Python MySQLDB, Repository Handler and Git.

  Regarding the database, CVSanalY provides a set of tables that represents the history of the project based on the information from the repository log, these tables are filled during the parsing process. Main tables with relevant information for the analysis are:

  - actions: describes the different actions performed in every commit.
  - files: contains an identifier for every file or directory found in the repository.
  - file_links: contains relationships between files, always in a parent-child relationship.
  - files_types: an extension of the schema, containing the file type associated of every file found in the repository.

  - people: contains the name and email (when available) of the people involved in the repository.
  - scmlog: contains every commit in the repository, represented by a record with univocal identificator.

  Available in github.com as: https://github.com/MetricsGrimoire/CVSAnalY.

- MySQL: a widely used FℓOSS (under GNU General Public License) relational database management system, which allows the analysis of the database filled by CVSAnalY.

  Available in mysql.com as: http://dev.mysql.com/downloads/mysql.

- NumPy: a FℓOSS (under BSD license) Python library. A fundamental package for scientific computing in Python. It allows the use of Scipy.

  Available in github.com as: https://github.com/numpy/numpy.

- Scipy: a FℓOSS (under BSD license) scientific Python tool, for mathematics, science, and engineering. Here it is used to generate graphycs related to the analysed repositories and metrics.

  Available in github.com as: https://github.com/scipy/scipy.

- Matplotlib: a FℓOSS (under PSF license) python 2D plotting library which produces publication quality figures. It allows to visualice metrics used in this analysis.

  Available in github.com as: https://github.com/matplotlib/matplotlib.

3 Analisys

Here will be disscused the motivation for the selected weigths in the categories and metrics, more details of how to retrieve each of the metrics, and the the application of the weights associated to OpenStack, to the other projects.

3.1 Categories weigths

Weigths selected for each category were:

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

As mentioned before, the role assumed in this analysis is from the System Administrator standpoint, thus the functionality attribute has a significant weight, since it is important to have as much possible features available. However, the evaluation of a software project, should equally consider the elements that ensure the quality of it, and this is why the most important weights are equitably distributed between this and two other attributes that are considered critical.

It is essential to the existence of proper documentation, since the software solution is going to be deployed and maintained by SysAdmins who will need all posible details about it. It is desirable to have installation manuals, FAQs, study cases, as well as end-user manuals.

The presence of a well formed community is also crucial, in the sence that it is important to count with active contributors to the project. It will guarantee in a certain way the sustainability of the project over time.

From this point, start to drop the weights associated with other attributes, that are less valuated from the point of view of the role assumed. For instance, the support would probably be needed if there is a disaster incident, or if there is required to implement a mandatory feature that SysAdmins are not able to handle, but anyway with proper documentation it is feasible.

When discussing efficiency, and further related to the performance, it will be always a very relative atribute. All depends on the data sources consulted and their reliability, also their interpretation. On the other hand, it will depend on the available resources for the implementation and the production environment estimated for the project deployment. Therefore, this attribute has the lowest weight.

3.2 Metrics weigths

Following are showed the weights asigned to each metric, totaling a 100% for each category. As earlier, the evaluation of this project, should equally consider the various metrics in order to ensure the quality, but anyway the role assumed is going to introduce certain preference over weights.

3.2.1 Category: Functionality

Metric	Weight
Administrators configuration system	30%
Quota facilities	25%
Billing system	20%
Multi platform suport	15%
i18n	10%

- Administrators configuration system: since SysAdmins are going to administrate the cloud computing system, this feature is the most heavy.
- Quota facilities and Billing system: functionalities like this, are critical for daily operations considering hosting customers.
- Multi platform suport: since SysAdmins install the platform and usually do not modify it, it suffices to support Linux platforms, asumming that this is the main common platform for every server in the infrastructure.
- i18n: SysAdmins nowadays are used to a common language (English) by default when administrating systems, so this feature provides no further benefit.

How metrics were obtained: all calculations were made based on official documentation from the project's website available on internet, referenced in section 2.3.

3.2.2 Category: Documentation

Metric	Weight
Documentation update frecuency	55%
Number of contributors to documentation	45%

- It is required that a considerable group of contributors collaborate with the documentation in a further way, but also that it would be updated frequently.

  How metrics were obtained: using the CVSAnalY tool (discribed in section 2.5), proper tables from the database were consulted through MySQL in order to obtain the required information.

  - When was the last time that documentation was updated? (considering documentation as files with .txt extension):

```
mysql>
SELECT MAX(s.date) FROM scmlog s, actions a, file_types ft, files f
WHERE s.id=a.commit_id AND a.file_id=ft.file_id
AND ft.type='documentation' AND f.file_name LIKE '%.txt'
ORDER BY s.date;
```

– Amount of people who contributed to documentation in the current year (2013)? (considering documentation as files with .txt extension):

```
mysql>
SELECT COUNT(DISTINCT(p.id)) FROM scmlog s, actions a, file_types ft,
files f, people p WHERE s.committer_id=p.id AND s.id=a.commit_id
AND  ft.file_id=a.file_id AND ft.type='documentation' AND
f.file_name LIKE '%.txt' AND YEAR(s.date)='2013';
```

3.2.3 Category: Community

Metric	Weight
Community growth (commit number variation)	40%
Mean commits per developer last month	30%
Percent of files touched by only one developer	30%

- It is desirable in a fervent manner that the number of developers grow as the project evolves.

- It is required that developers show interest in the project in the same way that territoriality should not be present or at least should be minimized.

  How metrics were obtained: using the CVSAnalY tool (discribed in section 2.5), proper tables from the database were consulted through MySQL in order to obtain the required information.

  – Which was the mean number of commits per developer last month (November 2013)?

    * Number of commiters in the last month (November 2013):

    ```
    mysql>
    SELECT COUNT(DISTINCT(p.id)) FROM scmlog s, people p
    WHERE s.committer_id=p.id AND YEAR(s.date)='2013'
    AND MONTH(s.date)='11';
    ```

* Number of commits in the last month (November 2013):

```
mysql>
SELECT COUNT(s.id) FROM scmlog s, people p WHERE s.committer_id=p.id
AND YEAR(s.date)='2013' AND MONTH(s.date)='11';
```

Then the average would be calculated as [Number of commits / Number of developers].

– Percentage of source code files modified only by one developer in the history of the project?

* Total number of files in the project:

```
mysql> SELECT COUNT(DISTINCT(file_path)) FROM file_links;
```

* Number of files modified only by one developer:

```
mysql> SELECT COUNT(p.email) FROM scmlog s, people p, file_links fl
WHERE fl.commit_id=s.id AND s.committer_id=p.id GROUP BY fl.file_path
HAVING COUNT(p.email)=1;
```

Then the percentage would be calculated as [(Files touched by one developer / Total files) * 100].

– Committers number increase/decrease, calculated as percent difference between years 2012 and 2013?

* Number of commiters in year 2012:

```
mysql> SELECT COUNT(s.id) FROM scmlog s, people p
WHERE YEAR(s.date)='2012' AND s.committer_id=p.id;
```

* Number of commiters in year 2013:

```
mysql> SELECT COUNT(s.id) FROM scmlog s, people p
WHERE YEAR(s.date)='2013' AND s.committer_id=p.id;
```

Then the Percentage Growth Rate would be estimed as {[(CurrentYear - PastYear) / PastYear] * 100}.

3.2.4 Category: Support

Metric	Weight
Companies providing professional support	100%

- Since this is the only metric for the category, there is no further analysis for the selected weight.

How metrics were obtained: based on official online documentation from the project's website, and internet research, referenced in section 2.3.

3.2.5 Category: Efficiency

Metric	Weight
Performance tuning and configuration	70%
Performance testing and benchmark reports available	30%

- For a SysAdmin it is highly productive to count with guidelines that improve the cloud computing solution.
- Eventually the performance testing and benchmark reports provide an important starting point for a new implementation.

How metrics were obtained: based on official online documentation from the project's website, and internet research, referenced in section 2.3.

3.3 Application of OpenStack weights on other projects

This process consisted of apply unweighted scores from metrics of every other projects (CloudStack, Eucalyptus and OpenNebula), to the spreadsheet associated to OpenStack, without varying the percentages asigned to OpenStack project (as explained in section 2.4), assuming the System Administrator rol.

4 Results

Here are some results regarding the Unweighted Ratings obtained by each project according to the corresponding category:

Category: Functionality	
Project	Unweighted Rating
Eucalyptus	4.8
OpenNebula	4.4
CloudStack	4.2
OpenStack	4

Category: Documentation	
Project	Unweighted Rating
CloudStack	5
OpenStack	4.45
Eucalyptus	4.45
OpenNebula	4

Category: Community	
Project	Unweighted Rating
CloudStack	4.3
Eucalyptus	2.7
OpenStack	2.4
OpenNebula	2.2

Category: Support	
Project	Unweighted Rating
OpenStack	5
CloudStack	3
Eucalyptus	3
OpenNebula	3

Category: Efficiency	
Project	Unweighted Rating
OpenStack	3
CloudStack	3
Eucalyptus	3
OpenNebula	1.6

In the following table is showed the final rating for each of the projects evaluated with OpenBRR methodology:

Project	Score
CloudStack	4.13
OpenStack	3.76
Eucalyptus	3.74
OpenNebula	3.26

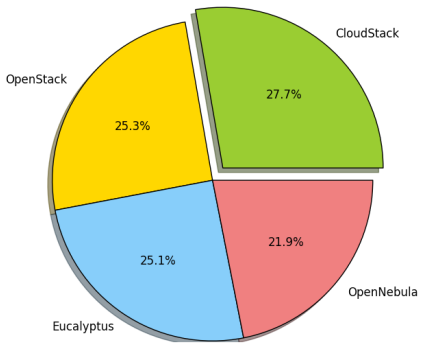


Figure 9: OpenBRR final score

5 Conclusions

In general, the four projects obtain competitive results regarding the evaluated metrics. Although any way Eucalyptus seems to have the better functionality features, CloudStack appears to have the better documentation and community, OpenNebula is the only option with lower score in relation to efficiency, and OpenStack shows a winning

score regarding the support. After applying OpenBRR evaluation method, there is a notable difference between CloudStack and the other projects in the final score obtained, indicating that this is the cloud computing project that best fits the proposed requirements.

It is important to note that the applied model corresponds to a light-weight evaluation method, and as such, allows a superficial analysis. In this sense, there are some elements that could be improved in the application of procedures, including for example:

- Establish a better definition frame for possible categories, reducing a bit the tailoring for the evaluator.
- Put away the imprecise terminology in the top nodes of the hierarchy.
- Exploit a bit more the 5-level scales for the criteria ranges.

Some problems encountered during the application of the methodology had relation with the vague interpretation of categories or methods proposed by the model, that eventually are very relative or are more focused on qualitative factors. However it is clear that no decision should be based solely on quantitative elements.

Three particular issues were on the one hand, the choice of a single repository for the analysis of some metrics, as the choice of 'nova' core repository for OpenStack project. It was not taken into account many other repositories that complement modularly the project.

On the other hand inadequate queries in repositories databases, can lead to unreliable results. For instance, considering only text files (with txt extension) as documentation for the project. Sometimes it is difficult to perform the appropriate questions to obtain the required information.

Finally, another problem turned out to be the evaluation of very short periods of time. It is convenient to make queries covering a wider range of time and take greater advantage of the available records in the respective repositories.

References

[Atos, 2006] Origin, Atos. (2006). Method for Qualification and Selection of Open Source Software (QSoS) version 1.6. qsos.org.

[Basili et. al., 1994] Basili V, Caldiera G and Dieter-Rombach H. (1994). The Goal Question Metric Approach. University Of Maryland and Universitat Kaiserslautern.

[Deprez et. al., 2008] Deprez, J C, Haaland K, and Kamseu F. (2008). QualOSS Methodology & QUALOSS assessment methods CETIC. June 2008.

[Duijnhouwer, 2003] Duijnhouwer, F-W. (2003). Open Source Maturity Model. Cap Gemini S.A. Nederland.

[Garcia, 2009] Garcia C. (2009). CVSAnaly: A tool to analyse software repositories. LibreSoft, Spain.

[OpenBRR.org, 2005] OpenBRR.org. (2005). Business Readiness Rating for Open Source: A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software. BRR 2005 - RFC 1.