# OpenBQR: a framework for the assessment of OSS

Davide Taibi[1], Luigi Lavazza[12], and Sandro Morasca[1]

1 Università dell'Insubria

luigi.lavazza@uninsubria.it, sandro.morasca@uninsubria.it,
davide.taibi@uninsubria.it, WWW home page: http://www.uninsubria.it

2 CEFRIEL

WWW home page: http://www.cefriel.it

**Abstract**. People and organizations that are considering the adoption of OSS, or that need to choose among different OS products face the problem of evaluating OSS in a systematic, sound and complete way. While several proposals concerning the evaluation of costs and benefits exist, little attention has been given to the evaluation of technical qualities and, in general, to the "usage-oriented" issues. In this paper the existing proposals are examined, the different types of qualities and issues that are relevant to potential users are described, and a coherent and innovative method for the evaluation of OSS is proposed. The proposed method is expected to support the potential user in the evaluation and choice of OSS in a flexible way, taking into account all the aspects that are relevant to the user.

## 1    Introduction

Open Source Software is a continuously growing movement. In order to give an idea of the size of the phenomenon, note that at the end of 2006 there were over 100,000 ongoing OSS project based on the best known repositories (such as SourceForge, CodeHaus, Tigris, Java.net and Open Symphony). OSS can also boast of several success stories: programs like the Apache projects, Netscape/Firefox, Eclipse, Linux, MySQL, and several others are well known and used by a huge number of people worldwide. Nevertheless, there are several areas where OSS was not adopted, at least not as widely as it could be expected. An example is given by the so called desktop environments and office applications. In fact, even in the areas where OSS has been successful, there are several potential users that did not adopt OSS.

Understanding why the adoption of OSS is limited is quite complex. A first reason is that the very concept of Open Source is hardly understood   [1] [2] . People tend to confuse OSS with free software (i.e., software that can be used without paying any fee) and open standards with proprietary disclosed software (like PDF) [1]. Another reason is that it is not obvious how to carry out the cost/benefit

analysis, given that the acquisition cost of OSS is usually null. Recently, the concept of Total Cost of Ownership (TCO) has been proposed as a mean to evaluate the cost of adapting, managing and maintaining OSS; nevertheless, the concept of TCO is not widely used, partly because it is not well understood (there are several, often not coherent, definitions) and partly because there is the suspect that most published TCO evaluations are driven by software vendors who want to convince customers that the commercial option is economically profitable. Finally, deciding the adoption of OSS requires the evaluation of the qualities of candidate OS programs, and their comparison with commercial programs. However, assessing the qualities of OSS is still a practice not well consolidated. Organizations facing the problem of deciding about the adoption of OSS have hardly any guide for carrying out a well structured comprehensive evaluation.

On the other hand, the producers of Open Source software cannot rely on clear indication concerning the factors that could determine the success of their products.

In this paper we discuss the qualities of OSS that determine its success and the features of OSS that should evaluated by potential users and adopters. Based on these considerations, a framework for the assessment of OSS is proposed. The goal is that such framework explicitly describes the qualities and properties of OSS that are considered important by both users and producers. In this way the framework can be employed by potential users for evaluating OSS. On the contrary, producers will get indications of what users value more, thus understanding what needs to be improved in their proposals.

The paper is structured as follows: Section 2 presents the current situation and the most recent proposals concerning OSS evaluation. Section 3 describes the features of OSS that –according to our analysis and understanding– are deemed important by organizations and professional users. Based on these considerations, our proposal for an OSS evaluation framework –named OpenBQR– is described in Section 4. In Section 5 we describe the validation activities that we carried out in order to confirm the capability of OpenBQR to represent the important features of OSS. Section 6 describes a web-based tool for carrying out the evaluations according to the criteria defined by the OpenBQR. Finally, Section 7 draws some conclusions.

## 2    State of the art and related work

**The economic perspective**

The first and most obvious problem with OSS is to assess its cost. Often OSS is free, i.e., there is no fee to pay in order to use the software; however, even in these cases it is clear that using OSS requires some investment. TCO (Total Cost of Ownership) addresses the evaluation of the cost of adopting and using a software program, including all the expenses, and spanning the whole lifecycle of the system   [8]. Therefore, TCO involves the evaluation costs due to acquisition, adaptation, deployment, training, operation, maintenance, etc.

TCO applies to both OS and commercial software, thus allowing the comparison of costs. In fact, TCO became popular also because it was used to support both the thesis that OSS is more convenient than commercial software, and the vice versa.

Although TCO had the merit of providing a sound and comprehensive basis for the evaluation of SW costs, it is limited with respect to two important issues:

- TCO does not address the costs that are connected with the evolution of the user's business process, which could require updating the software or even changing it, thus calling for additional investments.

- TCO does not include the evaluation of benefits, thus providing an incomplete view of the financial consequences of adopting the considered software.

Other proposals have addressed these limitations of TCO. In order to take into consideration the future evolution of the users' needs, Cosenza proposed the Total Account Ownership (TAO) index, which aims at representing the degree of freedom of the user with respect to the technology provider   [7]. The TAO considers issues like contracts and licenses, software adaptability, openness of formats and interfaces, documentation, training and assistance providers, etc., and indicates to what extent adopting a given piece of software is a commitment for the future.

The Full Business Value (FBV) aims at representing the whole value of the investment and includes the assessment of: system efficiency; system effectiveness; business efficiency; business effectiveness. The TCO can therefore be seen as a means to prove part of the information required by the FBV.

However, none of the TCO, TAO and FBV indexes address the issue of software quality. Since the adequacy of the software –from both the functional and quality point of view– is of fundamental importance, it is clearly necessary to assess them.

Next section discusses the evaluation of technical qualities as well as the assessment of the software adequacy with respect to the business process it is supposed to support.

## The quality perspective

Recently, the problem of evaluating OSS became evident, so that a few organizations invested some effort in the creation of models for the quality and evaluation of OSS. The variety of models proposed witnesses the attention for the problem, but also demonstrates the difficulty of defining a fully satisfactory model.

The Open Source Maturity Model (OSMM)   [3] is an open standard that aims at facilitating the evaluation and adoption of OSS. The evaluation is based on the assumption that the overall quality of the software is proportional to its maturity.

The evaluation is performed in three steps:

1. Evaluation of the maturity of each aspect. The considered aspects are: the software product, the documentation, the support and training provided, the integration, the availability of professional services.

2. Every aspect is weighted for importance. The default is: 4 for software, 2 for the documentation, 1 for the other factors.

3.    The overall maturity index is computed as the weighted sum of the aspects' maturity.

The OSMM has the advantage of being quite simple. It allows fast (subjective) evaluations. However, the simplicity of the approach is also a limit: several potentially interesting characteristics of the products are not considered. For instance, one could be interested in the availability of professional services and training, in details of the license, etc. All these factors have to be 'squeezed' into the five aspects defined in the model.

In general we doubt that using 'maturity' as a proxy of the overall OSS quality is a good idea. Since we are interested in the evaluation of the OSS quality, it is much more effective to go straight for the definition of metrics that represent directly the aspects of the SW product that determine the quality for the user, i.e., what the users consider important in order to make OSS suitable for usage.

The Open Business Readiness Rating (OpenBRR) [5] is an OSS evaluation method aiming at providing software professionals with an index applicable to all the current OSS development initiatives, reflecting the points of view of large organizations, SMEs, universities, private users, etc. On the official Open BRR site several evaluations are available. They can be examined and easily adapted: you just need to input the parameters that suit best your needs in the spreadsheet containing the evaluation. The proponents of the method plan to apply it to all SourceForge and Java.net projects, so that potential users can find a ready to use evaluation of the software they are interested into.

In the first step of the evaluation, the list of programs to be evaluated is compiled. Then every component is evaluated with respect to a set of indicator selected according to the target usage and including: the type of license, the compliance with standards, the existence of a user base, the availability of reliable support, the implementation language, internationalization, etc. Then the functionality of products is evaluated. The features of a "reference application" are identified and their importance is graded with respect to "standard usage". Then every product is evaluated with respect to how well it implements every feature. Finally, the grades are normalized and the final evaluation (a grade in the 1..5 range) is computed.

The Open BRR is a relevant step forward with respect to the OSMM, since it includes more indicators, the idea of the target usage, and the possibility to customize evaluations performed by other, just by providing personalized weights. With respect to the latter characteristics, the Open BRR as however some limits: one is that for many products it is difficult to choose a "reference application" that reflects the needs of all the users; another is that there are lots of possible target usages, each with its own requirements; finally, every subjective evaluation performed by a user could be not applicable to other users. In any case, the final score is probably a too synthetic indicator to represent the complex set of qualities of a software product.

Qualification and Selection of Open Source Software (QSOS) is a model for the selection and comparison of OS and free software   [4]. The evaluation process is carried out in four independent iterative phases. The definition phase aims at identifying the factors to be considered in the following phases. Phase 2 aims at collecting from the OS community the relevant information concerning the products. The goal is to create for every product an identity card (IC) reporting general information (name of the product, release date, type of application, description, type of license, project URL, compatible OS, …), available services, functional and technical specifications, … The quality aspects of the selected products are evaluated, and a grade (in the 0..2 range) is assigned according to the evaluation guidelines provided by QSOS. Phase 3 is dedicated to the definition of the selection criteria. The user's needs and constraints are described. Phase 4 consists in the comparison of the products' evaluation forms with the selection criteria, and in the identification of the product that matches betters with the user's needs and constraints.

Although in principle the method is effectively applicable to most OSS, the QSOS approach does not represent a relevant step forward with respect to other evaluation methods. Its main contribution is probably the explicitation of the set of characteristics that compose the IC, and the provision of a guideline for the consistent evaluation of these characteristics. Nevertheless, the evaluation procedure is too rigid and a bit cumbersome. For instance, it is required to define the IC of products that could be filtered away in phase 4 because they do not match the requirements. Such a procedure is justified when the ICs of products are available from the OS community before a user begins the evaluation. However even in this case it may happen that the user needs to consider aspects not included in the IC: this greatly decreases the utility of ready-to-use ICs. The strict guidelines for the evaluation of the IC, necessary to make other users' scoring reusable, can be ill suited for a specific product or user. Finally, even though in the selection criteria it is possible to classify requirements as needed or optional, there is no proper weighting of features with respect to the intended usage of the software.

# 3    Features of OSS that determine its acceptance by professional users

Assessing Open Source Software can require a complex process. In this Section, we describe the characteristics that are taken into account by people in order to assess the overall quality of OSS when choosing and adopting an OSS.

After a complete analysis of requirements, a set of parameters should be assessed, which favour a complete comprehension of the OSS being evaluated. We have identified several straightforward indicators that clearly show the quality of a software package to be adopted, divided into five different areas: functional requirement analysis, target usage assessment, internal quality, external quality, and likelihood of support in the future.

**Target usage assessment**

*License*: Not all open licenses are equal. Some licenses are more restrictive than others. If you need to extend the software, copy left properties are important because they allow modification of the code base and the redistribution of the modified version as long as the new product stays open.

*Compliance with standards*: for several application domains compliance with standards is important. For instance, in a website implementation, valid W3C-HTML code is a first step toward more compatibility with browsers and better rendering of pages. Using only strict HTML (that is, the Strict HTML DTD) makes the site easier to maintain and evolve.

*Implementation language*: if customization work and internal support are required, it is important to choose a product written in a programming language that is sufficiently mastered by the organization's programmers.

*Internationalization Support*: useful for applications that need to be translated into different languages.

*Books*: the availability of books about the software is a strong indicator of the software's level of maturity and popularity.

*Interest by well known industry and market analysts and consultants*: the availability of research reports on the software by analysts from leading market research firms (like Gartner or IDC) usually witnesses the relevance and diffusion of products.

**Internal quality**

With OSS it is possible to examine the internal quality of software, which is generally not disclosed for commercial software. For the purpose of evaluating the internal qualities you can choose among the many metrics proposed in literature and effectively supported by tools, like McCabe Cyclomatic Complexity, Chidamber and Kemerer's object-oriented metrics suite, Halstead complexity metrics, etc.

**External quality**

The main indication for the external quality of a software product is the defect density. It is therefore interesting to evaluate the number and severity of bugs over time, as well as defect removal speed. The latter is also a good indicator of the quality of support for the product.

   In some cases it is also relevant to evaluate the defect removal process. In some cases, the removal of specific defect can be sped up by "donations" (in practice, you pay the organization maintaining the software for solving the problem that is relevant for you). If you are considering a product with a high donation/bugs ratio, you must consider this cost of maintenance in your cost/benefit analysis.

**Probability of support in the feature**

General it is needed that a software product is supported as long as it is in use. We can estimate if the OSS being evaluated will be supported in the future through an in-depth analysis of the community of OSS developers, assessing:

- The "vitality" of the product, indicated by its age and the number and frequency of releases.

- The number of companies involved in the development. A large number of companies is a good index of probability for a continuative support.
- The number of developers per company is useful to understand how important – or even "strategic"– each company considers the OSS product.
- The number of independent developers is also relevant, since a large community of developers guarantees a continuous development and maintenance effort.

## 4    Open BQR: a framework for the evaluation of OSS

We defined Open BQR as an extension and integration of Open BRR and QSOS to address some of the problems of the current OSS evaluation methods, which are still immature, due to the relative novelty of the field. Here, we list some problems that Open BQR helps addressing.

- Existing methods usually focus on specific aspects of OSS.
- Some methods proceed to evaluating indicators before they are weighted, so some factors may be measured or assessed even if they are later given a very low weight or even a null one. This results in unnecessary waste of time and effort.
- No OSS evaluation method adequately deals with internal and external product qualities, even though the source code is available.
- The dependence of the users of OSS is not adequately assessed, especially the availability of support over time and the cost of proprietary modules developed by third parties.

During the definition of Open BQR, we tried to build a complete, simple, repeatable, adaptable and open OSS evaluation method with the following characteristics. As such, Open BQR can be used by several types of users, including ICT experts who need to evaluate and select OSS products, OSS developers, software quality assurance and measurement professionals. The main features of Open BQR concern the investigation of a number of relevant aspects of an OSS product, including:

- Functional adequacy to requirements;
- Quality, in terms of absence of defects or time-to-fix;
- Availability of maintenance support;
- Cost of non OSS modules or necessary development tools;
- Other issues like license type, programming language ...

The evaluation process is composed of three phases, in the same line of thought as Open BRR, as we detail in the following subsections. These phases and their sub-phases consider the OSS product features outlined in Section 3.

**Quick Assessment Filter**

Like in the Open BRR a list of topics is identified, along with their characteristics. Unlike in the Open BRR, the characteristics are measured only after a weight has been assigned to them. The idea is to avoid data collection for characteristics that may be deemed of no or little importance, and reduce the effort and time for defining

a measurement plan and collecting the data. This phase is divided into five steps, each of which addresses a different *area*, as follows.

1. **Selection of indicators based on scope and target use**: First, the application target is selected (Mission-critical, Regular, Development, Experimentation, …). Second, the license type is assessed, to check if the license type allows the development of the product as required by the specifications. Third, standard compliance, implementation language, internationalization support, books, and interests by major analysts are taken into account.

2. **Analysis of external qualities**: mainly, this step addresses the defects uncovered, the percentage of those that were fixed, and the distribution and average of the time it took to fix them.

3. **Analysis of internal qualities**: internal sub-characteristics qualities from the ISO9126 standard have been selected, along with other common indicators, such as McCabe's cyclomatic complexity.

4. **Product support over time**: this can be quantified based on the number of programmers that provide solutions to the incoming requests.

5. **Existence of required functionalities**: based on the user requirements, the functionalities of the OSS product are weighted on a 0 - 9 scale, to assess their relative importance. The required functionalities are then assessed on a 0 – 100 scale (the value '0' meaning "not implemented" and the value '100' meaning "fully implemented").

**Data Collection & Processing**

The outcome of the previous phase is a list of all the necessary indicators, along with their assigned weights. This phase is organized as follows:

1. **Pruning**: All of the indicators with a zero weight or a weight below a user-specified threshold are eliminated.

2. **Measurement**: The remaining characteristics are measured.

3. **Normalization**: The weights for each of the five areas described in the steps of Phase 1 are normalized to a total of 100. This allows for a fair comparison across different areas, i.e., each area will receive a score between 0 and 100.

4. **Assessment**: The final score for each area is obtained, and a single score is computed for the entire product as a weighted sum of the results obtained for the single areas, if needed. This can be useful for a first assessment, for instance for filtering out products whose total overall score is too low. An in-depth comparison among OSS products requires the knowledge of the values obtained in the single areas, along with the evaluation of costs.

**Data Translation**

This last phase consists in visualizing the results of the evaluation of the various products for comparison purposes. An example of a polar plot for immediate visualization is given in **Fig. 1**.

## 5    Validation

In order to validate the approach three well known Content Management Systems have been evaluated by means of Open BQR. The results have been compared with the informal evaluations of the same tools expressed (via forums etc.) by the community of users. For space reasons we cannot provide the entire evaluation, so we report here a few details to show how one should proceed with OpenBQR. The three products we analyzed are Mambo (http://www.mamboserver.com), Drupal (http://www.drupal.org) and WebGUI (http://www.webgui.it). We started by setting a number of requirements for the application, and we ranked them in order of importance. We envisioned a personal web application with a user-defined layout (weight 10), with user-operated functionalities for creating, reading, updating, deleting web pages (weight 10) and files (weight 5), image gallery (weight 7), and support for the Italian language (weight 4). We then carried out the other activities of the Quick Assessment Filter.
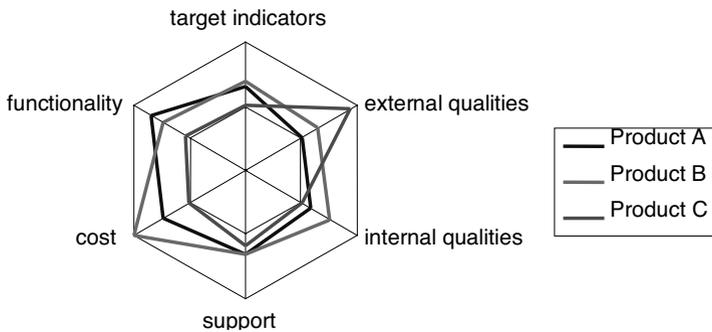


**Fig. 1.** Open BQR evaluation: visualization of the comparison of three products.

Some details of the evaluation are reported in Appendix A. The following comparison table reports the synthesis of the results for the three products.

| Product | Mambo | Drupal | Web GUI |
|---|---|---|---|
| Target usage assessment | 30,69 | 20,31 | 28,45 |
| Analysis of external qualities | 11,72 | 9,66 | 14,14 |
| Availability of support in the future | 35,68 | 29,14 | 13,10 |
| Satisfaction of functional requirements | 100 | 100 | 100 |
| **Overall evaluation** | **78,10%** | **68,10** | **55,69** |
| Rating | ☆☆☆☆ | ☆☆☆ | ☆☆ |

According to our analysis, the best product is Mambo, although from the point of view of the external quality WebGUI is better.

## 6    A web-based tool implementing OpenBQR

A web based tool is being developed to help users apply the method in a coherent way. The tool is designed to be able to easily manage a complete assessment, from

the requirements analysis, to the final visualization of the results. The main goal of the tool is to provide a framework for the comprehension and application of the Open BQR model, through all its steps, starting from the requirements analysis, to the assessment of all the indicators and at the end showing a complete report with the total score and a radar graph.

Through the web-based tool, we also collect data about the usage of Open BQR (what features the users consider more important, what kind of software they are interested into, etc.). These data are used to improve the method and the tool. The users' privacy is protected by a nondisclosure agreement. Data are published only in aggregated form and we take care that no information about specific users is ever made public.
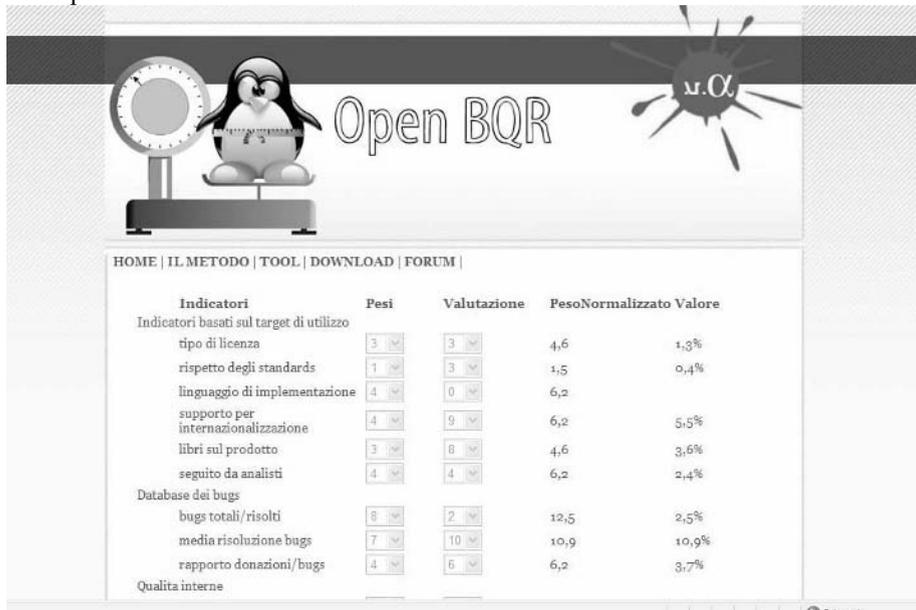


**Fig. 2.** A screenshot of the Open BQR web-based tool.

## 7   Conclusions

In this paper, we have introduced a new OSS quality evaluation framework, which we built by integrating and extending existing approaches, so as to take advantage of their strengths, alleviate some of their drawbacks, and include some additional characteristics of interest. OpenBQR is fairly complete, simple, repeatable, adaptable, and open, so it can be used by different software organizations.

Future work will include using OpenBQR for the evaluation and comparison of OSS products in several different areas. This may lead to tailored, more specific versions of OpenBQR for the different application areas. Also, this will allow us to further validate both the way the approach is used and its usefulness in reflecting what the software industry expects from an OSS quality evaluation framework. This

will entail interviews and studies that will involve all the major stakeholders, i.e., the OSS producers and the OSS users.

## 8     References

[1]   D. Cerri and A. Fuggetta, "Open Standards, Open Formats, and Open Source", July 2006, Submitted for publication

[2]   A. Fuggetta. "Open source software: an evaluation". Journal of Systems and Software, April 2003.

[3]   "Making Open Source Ready for the Enterprise: The Open Source Maturity Model", from "Succeeding with Open Source" by Bernard Golden, Addison-Wesley, 2005, available form http://www.navicasoft.com

[4]   Atos Origin, "Method for Qualification and Selection of Open Source software (QSOS), version 1.6", http://www.qsos.org/download/qsos-1.6-en.pdf

[5]   "Business Readiness Rating for Open Source - A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software", BRR 2005 - RFC 1, http://www.openbrr.org.

[6]   S. H. Kan, "Metrics and Models in Software Quality Engineering, 2nd Edition", Addison Wesley Professional, 2003.

[7]   G. Cosenza, Liberi di Cambiare, Computer Business Review On-line Italy, http://www.cbritaly.it/Aree-tematiche/OSS/Liberi-di-cambiare. (In Italian).

[8]   J. Smith David, D. Schuff, R. St. Louis, "Managing your total IT cost of ownership", Communications of the ACM, Volume 45, n. 1 (January 2002).

[9]   http://www.qualipso.eu

# 9    Appendix A. Details of the evaluation of CMS

| Product | Drupal 4.7.4 | Mambo 4.5.3 | WebGUI 7.0 |
|---|---|---|---|
| **System requirements** | **Drupal** | **Mambo** | **WebGUI** |
| Application Server | PHP 4.3.3+ | PHP 4.1.2+ | mod_perl |
| Cost | Free | Free | Free |
| Database | MySQL, Postgres | MySQL | MySQL |
| License | GNU GPL | GNU GPL | GNU GPL |
| Operating System | Any | Any | Any |
| Programming Language | PHP | PHP | Perl |
| Web Server | Apache, IIS | Apache, IIS, any PHP enabled web server | Apache |
| **Support** | **Drupal** | **Mambo** | **WebGUI** |
| Commercial Manuals | Yes | Yes | Yes |
| Commercial Support | Yes | Yes | Yes |
| Commercial Training | Yes | Yes | Yes |
| Developer Community | Yes | Yes | Yes |
| Online Help | Yes | Yes | Yes |
| Public Forum | Yes | Yes | Yes |
| Third-Party Developers | Yes | Yes | Yes |
| **Ease of Use** | **Drupal** | **Mambo** | **WebGUI** |
| Mass Upload | Free Add On | No | Yes |
| Prototyping | No | No | Yes |
| Server Page Language | Yes | Yes | Yes |
| Spell Checker | Free Add On | No | Limited |
| Style Wizard | No | No | Yes |
| Template Language | Limited | Yes | Yes |
| WYSIWYG Editor | Free Add On | Yes | Yes |
| **Built-in Applications** | **Drupal** | **Mambo** | **WebGUI** |
| Blog | Yes | Yes | Yes |
| Document Management | Limited | Free Add On | Limited |
| File Distribution | Free Add On | Free Add On | Yes |
| Link Management | Free Add On | Yes | Yes |
| Mail Form | Free Add On | Yes | Yes |
| Photo Gallery | Free Add On | Free Add On | Yes |

| Indicators | Assigned weight | Evaluation | Normalized weight | Weighted evaluation |
|---|---|---|---|---|
| *Target usage* | | | | |
| type of license | 9 | 10 | 15.52 | **15.52** |
| standard compliance | 5 | 8 | 8.62 | **6.90** |
| implementation language | 0 | 0 | 0.00 | **0.00** |
| internationalization support | 4 | 10 | 6.90 | **6.90** |
| books | 2 | 4 | 3.45 | **1.38** |
| analysts and consultants | 0 | 0 | 0.00 | **0.00** |
| *External qualities* | | | | |
| bug number | 6 | 8 | 10.34 | **8.28** |
| average time for defect removal | 4 | 5 | 6.90 | **3.45** |
| effect of donations of defect removal speed | 6 | 0 | 10.34 | **0.00** |
| *Internal qualities* | | | | |
| complexity | 0 | 0 | 0.00 | **0.00** |
| reuse | 0 | 0 | 0.00 | **0.00** |
| dependencies | 0 | 0 | 0.00 | **0.00** |
| others | 0 | 0 | 0.00 | **0.00** |
| *Future support* | | | | |
| number of releases | 9 | 10 | 15.52 | **15.52** |
| number of organizations supporting the software | 5 | 9 | 8.62 | **7.76** |
| number of programmers per organization | 4 | 9 | 6.90 | **6.21** |
| number of independent programmers | 4 | 9 | 6.90 | **6.21** |
| **Total** | | | 100.00 | **78.10** |