OSSpal: Finding and Evaluating Open Source Software

Anthony I. Wasserman(✉), Xianzheng Guo, Blake McMillian, Kai Qian, Ming-Yu Wei, and Qian Xu

Carnegie Mellon University, Silicon Valley, Moffett Field, CA 94035, USA
{tonyw,blake.mcmillian,ming.yu.wei,qian.xu}@sv.cmu.edu,
{xianzheg,kaiq}@andrew.cmu.edu

Abstract. This paper describes the OSSpal project, which is aimed at helping companies, government agencies, and other organizations find high quality free and open source software (FOSS) that meets their needs. OSSpal is a successor to the Business Readiness Rating (BRR), combining quantitative and qualitative evaluation measures for software in various categories. Instead of a purely numeric calculated score OSSpal adds curation of high-quality FOSS projects and individual user reviews of these criteria. Unlike the BRR project, for which there was no automated support, OSSpal has an operational, publicly available website where users may search by project name or category, and enter ratings and reviews for projects.

Keywords: Open source software · Software evaluation · Open source forges · Software metrics · FOSS · FLOSS · Software taxonomy

1 Introduction

Free and open source software (FOSS) has flourished in the past decade. GitHub, the leading repository for FOSS projects, now hosts more than 50 million projects (not all FOSS). These projects vary in software quality, project maturity, documentation, and support, and hence in their suitability for widespread use. Projects in their early stages aren't mature enough for use outside the development team and the small band of brave souls who are willing to try almost any piece of software and accept the results.

Over time, some of these projects will thrive, becoming stable and useful software, while other projects will be abandoned, with little or nothing to show for the effort. High quality open source software (FOSS) allows developers to incorporate reliable code in their applications and to focus their efforts on other product functions. FOSS software can also allow users to avoid the costs of proprietary software for use both within their organizations and in their products.

While business decision makers, such as corporate IT managers, have a good understanding of the business models of traditional proprietary software vendors and how to work with them, open source software presents them with a new set of challenges. While some open source software is provided through vendors who offer regular releases, technical support, and professional services, other software may not have an established

F. Balaguer et al. (Eds.): OSS 2017, IFIP AICT 496, pp. 193–203, 2017.
DOI: 10.1007/978-3-319-57735-7_18

source of commercial support, even though the software itself is of good quality from a technical perspective.

Evaluating open source software is quite different from evaluating traditional packaged applications and tools. Open source software can be freely used according to the terms of its license (see the Open Source Definition and open source licenses at http://www.opensource.org). Traditional enterprise software vendors often provide pre-purchase support for a trial of the software. While some FOSS projects have commercial sources of training and support, most users of FOSS must seek support from online forums and documentation or, in some cases, from commercially published books. It's still uncommon for industry analysts to include open source software in their product evaluation frameworks.

For organizations without prior experience with open source software, it has, until now, been necessary to rely solely on internal evaluations and "word of mouth" recommendations. Even finding suitable candidate software can be daunting, and the process can prevent managers from trying to do so.

In the remainder of this paper, we review the initial goals and the experience with the Business Readiness Rating (Sect. 2), describe the approach of the OSSpal project (Sect. 3), describe the project status (Sect. 4), and conclude with goals for future work.

2 The Business Readiness Rating

For open source software to be widely accepted and used, it's essential to simplify the evaluation and adoption process and to increase the comfort level of decision makers in choosing and using open source software. Achieving this goal led to the concept of the Business Readiness Rating (BRR) in 2005 [1]. People often rely on reviews from trusted sources, e.g., Consumer Reports in the US, or collected opinions with summary scores, for technical and consumer products, restaurants, films, and automobiles. Some of these ratings are multidimensional, while others are a single combined score reflecting all facets of the reviewed product.

The Business Readiness Rating followed this well-established tradition, giving reviewers a framework to evaluate various characteristics of an open source project as they apply to that reviewer's specific requirements. Golden's Open Source Maturity Model (OSMM) [2] used a matrix of weights and values to calculate a maturity level for open source software. The OSMM, along with QSOS [3], and unpublished work by the consulting firm CapGemini, were all influences on the BRR approach. Subsequently, the OpenBQR project [4], within the Qualipso Project, drew upon all of these to create a new quality evaluation framework, but that effort does not appear to have progressed beyond the initial work reported in 2007.

2.1 Evaluation Categories

From discussions with evaluators, we identified categories that are important for the open source evaluation process. We used those categories, along with those found in

standard evaluation process documents (such as ISO/IEC 9126 [5] and ISO/IEC 25010:2011 [6]), and condensed them to seven (initially twelve) areas for evaluation:

- Functionality
  How well will the software meet the average user's requirements?
- Operational Software Characteristics
  How secure is the software? How well does the software perform? How well does the software scale to a large environment? How good is the UI? How easy to use is the software for end-users? How easy is the software to install, configure, deploy, and maintain?
- Support and Service
  How well is the software component supported? Is there commercial and/or community support? Are there people and organizations that can provide training and consulting services?
- Documentation
  Is there adequate tutorial and reference documentation for the software?
- Software Technology Attributes
  How well is the software architected? How modular, portable, flexible, extensible, open, and easy to integrate is it? Are the design, the code, and the tests of high quality? How complete and error-free are they?
- Community and Adoption
  How well is the component adopted by community, market, and industry? How active and lively is the community for the software?
- Development Process
  What is the level of the professionalism of the development process and of the project organization as a whole?

The first four categories are quite similar to those used to evaluate proprietary, i.e., closed source, software. For the latter three topics, it is usually easier to obtain the data for an open source project. In an open source project, the size of the user community is important for assessing the availability of informal support available, and the speed with which a posted question might be answered or a problem in the code might be fixed. Open source projects contain extensive data on the size of the development team and the list of outstanding issues, as well as the number and frequency of releases, data that is difficult, if not impossible, to obtain from closed source products. In this way, the project data can be used in the evaluation process, thus adding a quantitative aspect to what has traditionally been an informal process.

The term "Operational Software Characteristics" refers to those aspects of a software system that can be evaluated without access to the source code. It includes such quality-related areas as reliability, performance, scalability, usability, installability, security, and standards compliance. Evaluating "Software Technology Attributes" involves access to the source code to review software architecture, code quality, and internal documentation.

As additional examples, vendor support, standards compliance, test coverage, and the presence of published books are characteristics that indicate a high degree of readiness for an open source component. Different organizations can and should apply

different priorities and weightings to these categories (and perhaps to subcategories), based on the intended use of the software and their risk acceptance profile.

In summary, the BRR was conceived as an open and standard model to assess software to increase the ease and correctness of evaluation, and accelerate the adoption of open source software. Such a model should include the crucial requirements of a good software rating model — that it be complete, simple, adaptable, and consistent.

2.2 Experience and Shortcomings of the BRR

The BRR was used informally for many software evaluations since its initial release in 2005. As a manual process, it failed to gain much traction in the community. We attribute that situation primarily to the absence of automated tools to assist in the calculation, but also because of personal situations affecting team members. In addition, we found that business users of FOSS were rarely willing to take the time and effort to contribute their assessments back to the community, sometimes because they viewed their studies as having proprietary value to their companies. But the problems went well beyond those, and it took us some time to recognize them fully.

First, in most cases, it was easy to estimate which FOSS projects would receive a high BRR score, based on high awareness of the project, along with existing documentation and commercial support. FOSS projects such as MySQL and OpenOffice were mature, well-supported projects, well-suited for organizational adoption.

Next, a numeric score alone fails to reveal necessary details about a FOSS component, particularly how well it works in practice, as well as individual issues that can't be captured in the evaluation subcategories. That's particularly true of the functionality category, where a complex piece of software may be created to perform multiple functions, but may not do all of them satisfactorily for a specific requirement. The BRR didn't have a way to evaluate in detail the functionality of a FOSS product in a specific software category, nor could one easily do an evaluation based on a single key characteristic such as "security".

Third, using the BRR requires finding candidate FOSS software to evaluate. Many organizations lack internal expertise with FOSS software, and thus don't know where to begin, especially when they don't fully understand the concepts of FOSS. Doing a search for "open source content management systems", for example, yields a vast number of results (148 M when one of the authors tried it), that provide very little help to the user. Even if the search led the organization to cmsmatrix.org, which provides comparative information on content management systems (CMS), there are more than 1200 listed CMS's. In short, without previous FOSS experience or the use of external consultants, an organization isn't likely to be successful finding high-quality FOSS candidates this way. It was initially difficult for us to appreciate how challenging it is for someone without FOSS experience and technical knowledge to find FOSS software. For those users, it's easier to rely on opinions provided by IT industry analysts, which are almost exclusively limited to commercial software products and services.

Finally, and most significantly, we found that people rely heavily on the opinions of others, including both peers and "experts". The numeric score from the BRR would help

them to form a short list of candidates for their use, but they then wanted to see reviews or experience reports for those candidates.

All of these issues convinced us to take the project in a different direction while remaining focused on the goal of helping people find high-quality FOSS software that they could successfully adopt and use.

3 From BRR to OSSpal

3.1 Overview of Changes to the Model

We changed the project name to OSSpal because we thought that the concept of finding FOSS was broader than just "business" and we wanted a clean break with the BRR approach. But the new name had another nice property, namely the double meaning of "pal". In American and UK English, "pal" is an informal term for a friend, hence OSSpal. But the name was also chosen as a tribute to the late Murugan Pal, a key cofounder of the BRR project.

There are several major differences between OSSpal and the BRR, along with numerous less significant changes. First, we removed the calculated score, based on our previously-noted observation that the single digit result was not valuable to users, particularly because it hid the details of the seven evaluation criteria, which in turn hid the lower level details of such key areas as operational characteristics and functionality. Second, given the difficulty that people have in finding candidate FOSS software, we decided to create short lists for them. In other words, we curated FOSS projects, using some quantitative measures, including the number of commits, the number of forks, and the number of subscribers. Note that these metrics are not specifically correlated to FOSS quality, but rather to the level of project activity. We leave the assessment of quality to the individual reviewers of a project.

Furthermore, we grouped those projects into categories, based on the software taxonomy produced annually by the International Data Corporation (IDC) [7]. This grouping allowed us to define both generic evaluation criteria and category-dependent evaluation criteria. Finally, we built a website, using the open source Drupal content management system (CMS) [8], organizing the projects along the lines of the IDC taxonomy so that users could search by name and category, and allow registered users to enter their own reviews of specific projects.

We explored other sites that provide evaluation of FOSS projects. Of these, the most significant one is OpenHub (formerly Ohloh) [9], managed by BlackDuck Software. OpenHub collects data from open source repositories (forges) such as GitHub. Available information includes the project status, the number of contributors, the programming languages used, the number of contributions made by individual contributors, and metrics of project activity. Site visitors can leave an overall rating and a written review of individual projects, but the rating is simply based on a 1-to-5 star scale, intended to cover all aspects of the project, without the granularity used in OSSpal. Nonetheless, the detailed information provided on OpenHub is extremely valuable, especially for developers looking to join and contribute to a project, and we decided to provide a link between projects on OSSpal and the detailed developer-related data on OpenHub.

The OSSpal approach differs from other evaluation approaches, in that it uses metrics to find qualifying FOSS projects in the various categories, but leaves the assessment of quality and functionality of individual projects to external reviewers, who may also add informal comments to their scores.

3.2 Implementation and the Quick Assessment Tool

Implementation of a site for searching and adding projects, as well as adding user reviews, was a principal goal for the OSSpal project, especially since the absence of automated tools was a major shortcoming of the earlier BRR effort. We chose to build OSSpal on Drupal because using a CMS allowed us to greatly reduce the amount of coding needed and thus devote more effort to creating the content. In retrospect, we are very pleased with this decision. The Drupal core Taxonomy module was particularly helpful, as we were easily able to map software categories from the IDC taxonomy into a Drupal taxonomy, and thus associate FOSS projects with a category in the taxonomy. Furthermore, it's easy to modify the taxonomy as IDC modifies their taxonomy. We also gained the flexibility to modify the implementation taxonomy as needed. While our goal was to stay as close to the IDC taxonomy as possible, we found a few areas, particularly in the area of application development software, where we wanted a finer granularity. Making that change allowed us to refine the functionality for different types of application development languages, environments, and tools.

The quality attributes of FOSS projects can be classified into two general categories, hard metrics and soft metrics. Hard metrics are objective quantifiable measurements, covering most attributes in areas of software technology and development process. They can be collected efficiently through sending API calls to GitHub and Open Hub based on automated scripts. Soft metrics are subjective qualitative measurements, covering most attributes in areas of operational software characteristics as well as service and support.

To make the collection process of the hard metrics more efficient, we developed a web service to gather quantifiable FOSS project information from Open Hub and GitHub to determine if a FOSS project is high quality. Instead of searching for project attributes manually, the relevant attributes are returned to the user after querying the project's name in our web service. A user simply enters the name of a FOSS project into the search bar of the web service, and the script will display the hard metrics for the project after querying Open Hub and GitHub. The project information that is returned to the user includes attributes such as, number of project contributors, number of commits, lines of code, and project activity.

The implementation of the quick assessment tool uses the Flask web microframework, which takes the project name and uses the Open Hub and GitHub APIs to retrieve data which is returned in JSON format and then rendered as a web page. Figure 1 shows the data about project activity returned from a query on Electron. For now, we have used the quick assessment tool to screen 59 additional projects out of 101 candidate projects for inclusion on the site.

Fig. 1. Raw results of quick assessment tool for the electron project

From our work, we were able to extrapolate two key findings pertaining to hard metrics:

(1) Effectiveness of metrics. The most effective hard metrics to find high-quality FOSS projects are the number of contributors, the number of commits, the number of subscribers, the number of forks, and the number of open issues.
(2) Thresholds of metrics. With trial and error, the optimal threshold values for each effective hard metric is identified to meet the quality baseline for including new projects. Such metrics, for example, are the number of commits >1000, the number of forks >100, and the number of subscribers >50.

3.3 A Note About FOSS Quality

We have used, but not defined, the term "high-quality FOSS software". That's intentional, since there are a large number of definitions for "software quality". Miguel et al. [10] reviewed numerous software quality models for evaluating software products (without regard to source code availability). Their comparison of six different basic quality models yielded 29 different characteristics considered by one or more of the models. Ruiz and Robinson [11] published an extensive literature review to identify many different measures of open source quality, grouped by product quality (16 measures), process quality (12 measures), and community quality (11 measures), drawn from more than 20 different relevant articles. Hauge et al. [12] performed an extensive study on adoption of FOSS software in software-intensive organizations. However, their focus was primarily on organizational issues in adoption, rather than on the quality of the software being adopted. The SQO-OSS [13] quality model is interesting because it is specific to FOSS software and focuses on measurable criteria, but our experience from the earlier BRR effort is that some important quality aspects, such as usability, cannot be scored numerically.

Our approach was to include quality characteristics in the list of items that a reviewer could address. Every review of a FOSS component leaves a place for the reviewer to evaluate both generic and specific features of the software. The initial set of generic quality-related features are: installability, usability, robustness, security, and scalability, as well as an overall assessment. Note that it is difficult to find metrics for these five aspects. The set of specific features depends on the category of the software, where the list of features draws from the IDC taxonomy for that category. This aspect of the review allows reviewers to address the important quality issue of how well the software does what it is supposed to do. In summary, a thorough review of a FOSS component could combine quantitative items, as found in SQO-OSS, with an averaged score of community ratings for the generic and category-specific properties.

3.4 Using the OSSpal Site

The user of the OSSpal site can browse by category or can search by category or project name. In that way, the user might find the Electron project, and would then see the information shown in Fig. 2.

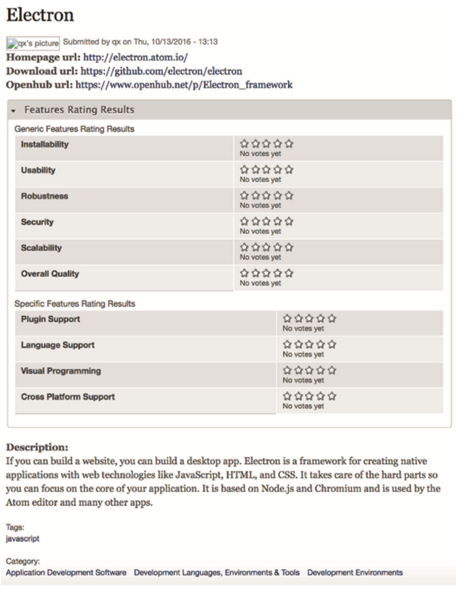

Fig. 2. OSSpal display for electron project

The information presented about Electron shows the user the location for the project home page, the software download page, and the detailed project information on OpenHub. The main section presents an average of the ratings, with the top section addressing generic software product issues and the lower section focused on features specific to its category. In this case, the project is newly listed, so there are no reviews

to summarize. Once there are reviews for the project, a user can scroll through them to see the individual ratings and reviewer comments, ten reviews to a web page.

Since the site currently contains a relatively small number of FOSS projects, we wanted to simplify the process for users to propose a new project for inclusion, and included a form on the OSSpal site for that purpose (see Fig. 3). As with traditional open source projects, the average user cannot commit a new project to the directory.

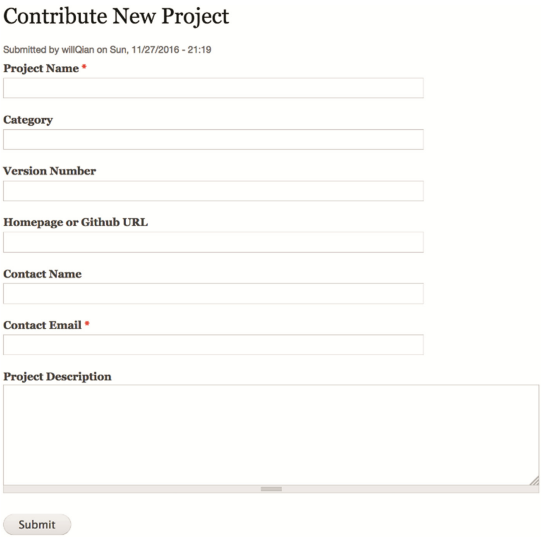

Fig. 3. Form for proposing a new FOSS project for the OSSpal site

Instead, the proposal is scored against the Quick Assessment metrics for possible site inclusion.

4 Status and Future Directions

At the outset, the OSSpal site has more than 300 FOSS projects identified by the OSSpal team, allocated among the 86 IDC categories. Initially, there are no registered users, and hence no reviews of those projects. However, we expect this situation to change quickly once the publicity spreads. We also expect the number of projects to grow significantly, with some projects added by the OSSpal core team but most submitted by the public. As noted, we have developed a tool for quickly assessing candidate projects for inclusion on the OSSpal site, but we believe that further refinement is needed, especially as the number of projects grows and as some FOSS projects become outdated or supplanted by newer projects.

As these numbers increase, the OSSpal site can provide data for analyzing and understanding the process by which people find and evaluate software. We also hope that the IT industry analysts will also draw upon the results of submitted reviews and

include FOSS projects, particularly those with commercial support, to their lists of recommended software for their clients.

Another important issue involves handling successive versions of FOSS projects. Of course, the information associated with the FOSS project should be kept up-to-date. Beyond that, though, is the question of how to associate user reviews with software versions, particularly when minor updates are released frequently. Major releases of a software project may have a significant impact on overall quality. It's not unusual for such releases to address a major concern, resulting in greatly improved performance or user interfaces. Thus, it can be valuable to tie evaluations to specific versions of a FOSS project. For example, Drupal 7 was released in early 2011 (after a code freeze in early 2009) and is still being maintained, with Drupal 7.54 being the current version at the time of this writing, even though Drupal 8 was released in November, 2015. On the surface, it seems straightforward to separate reviews of Drupal 7 from Drupal 8, but it's not clear that reviews of Drupal 7.43 should be separated from reviews of Drupal 7.44 (or perhaps anything newer), since the release frequency is approximately once per month. However, the Google Play Store for Android software handles reviews for each release of each app separately. The eventual solution may be to offer multiple options to the user, but that remains a research question.

Finally, there is work to be done with the site functionality and infrastructure. The current OSSpal.org site is hosted on a shared Linux server, which is adequate for light site traffic. As popularity increases, it will become necessary to migrate the site, initially to a dedicated server, and eventually to a dynamically scalable, cloud-based environment. A second issue involves analysis of the site visitors, particularly the source(s) of reviews. The success of OSSpal depends on reliable reviews and the recognition of efforts by people to unfairly influence the overall ratings; this is a problem already faced on a large scale by numerous well-known sites that accept reviews from the public, and we will have to address it in the context of OSSpal so that the results are not skewed.

These projects are just a representative sample of important issues for the OSSpal project. Addressing these issues, among others, will make it possible for the project to help organizations find and adopt FOSS projects.

Acknowledgments. We are grateful to the early sponsors of the Business Readiness Rating project and the lead author's related research: Google, HP, IBM, and Intel. We are pleased to acknowledge the other co-founders of the BRR project: Pete Kronowitt, Nat Torkington, and the late Murugan Pal. Finally, we appreciate the contributions of former Carnegie Mellon University Silicon Valley students (Anirudh Bhargava, Sneha Kedlaya, Poorva Jain, and Pramothini Dhandapany) for their help with the initial version of the OSSpal site and the set of FOSS projects.

References

1. Wasserman, A.I., Pal, M., Chan, C.: Business readiness rating for open source. In: Proceedings of the EFOSS Workshop, Como, Italy (2006)
2. Golden, B.: Succeeding with Open Source. Addison Wesley, Boston (2004)
3. Semeteys, R. et al.: Method for Qualification and Selection of Open Source software (QSOS), version 1.6, Atos Origin (2006). http://www.qsos.org

4. Taibi, D., Lavazza, L., Morasca, S.: OpenBQR: a framework for the assessment of OSS. In: Feller, J., Fitzgerald, B., Scacchi, W., Sillitti, A. (eds.) OSS 2007. ITIFIP, vol. 234, pp. 173–186. Springer, Boston, MA (2007). doi:10.1007/978-0-387-72486-7_14
5. ISO: Software Engineering – Product Quality – Part 1: Quality Model. ISO/IEC 9126-1:2001. International Standards Organization, Geneva (2001)
6. ISO: Systems and Software Engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. ISO/IEC 25010: 2011. International Standards Organization, Geneva (2011)
7. International Data Corporation Software Taxonomy 2016. International Data Corporation, Framingham, MA (2016). https://www.idc.com/getdoc.jsp?containerId=US41572216
8. Drupal, 27 February 2017. drupal.org
9. OpenHub, 27 February 2017. openhub.net
10. Miguel, J.P., Mauricio, D., Rodríguez, G.: A review of software quality models for the evaluation of software products. Int. J. Softw. Eng. Appl. **5**(6), 31–53 (2014)
11. Samoladas, I., Gousios, G., Spinellis, D., Stamelos, I.: The SQO-OSS quality model: measurement-based open source software evaluation. In: Russo, B., et al. (eds.) Open Source Development, Communities and Quality. IFIP International Federation for Information Processing, vol. 275, pp. 237–248. Springer, Boston (2008)
12. Hauge, Ø., Ayala, C., Conradi, R.: Adoption of open source software in software-intensive organizations–A systematic literature review. Inf. Softw. Technol. **52**(11), 1133–1154 (2010)
13. Ruiz, C., Robinson, W.: Measuring open source quality: a literature review. Int. J. Open Sour. Softw. Process. **3**(3), 189–206 (2013)