

Using FLOSSmole Data in Determining Business Readiness Ratings

Ashutosh Das
Carnegie Mellon West
Moffett Field, CA 94035 USA

msg4ashutosh@gmail.com

Anthony I Wasserman
Carnegie Mellon West
Moffett Field, CA 94035 USA

1.650.335.2807

tonyw@west.cmu.edu

ABSTRACT

This paper is a preliminary report on using FLOSSmole data retrieved from open source repositories (forges) to calculate Business Readiness Rating scores.

Categories and Subject Descriptors

D.2.8 [Software Engineering]: Metrics –Product metrics

General Terms

Measurement.

Keywords

Open source, Business Readiness Rating, FLOSSmole.

1. INTRODUCTION

Open source projects, like any other software, vary in the quality of their software, documentation, and support, and hence in their suitability for widespread use. There are many thousands of open source projects, with more than 145,000 registered projects on SourceForge alone. Some of these are well-suited for widespread adoption, while others may never achieve the level of quality and functionality needed for such use, especially in a business setting where the expectation is that the software will install easily, run dependably, and are regularly maintained and updated.

The process of evaluating open source software for potential use can be tricky. Many experienced IT managers and system administrators have an instinctive sense about open source projects, allowing their organizations to create a culture and expertise around open source. For those without such experience, 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 IT managers from trying to do so.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WoPDS 2007, June, 2007, Limerick, Ireland.

Copyright 2007 ACM 1-58113-000-0/00/0004...\$5.00.

2. THE BUSINESS READINESS RATING

Despite the available functionality and high quality of many open source projects, many companies have been slow to adopt open source software, preferring to rely upon commercial products from their current preferred vendors. The Business Readiness Rating™(BRR) [1] framework was created in 2005, with the goals of simplifying the evaluation and adoption process for open source software and helping IT decision-makers to become more comfortable in selecting and using open source software. A user of the BRR framework can evaluate an open source project by assigning weights to various factors of the project and calculating a score that gives a measure of its suitability for use. The BRR project is itself a community-based project, with the intent of obtaining both quantitative and qualitative reviews for many different projects.

Through a lengthy public review process, we identified categories that are important for open source evaluation. We used those categories, along with those found in standard evaluation techniques (such as ISO/IEC 9126 and the newly released ISO/IEC 25000), and condensed them down to seven areas for evaluation:

Functionality
Operational Software Characteristics
Support and Service
Documentation
Software Technology Attributes
Community and Adoption
Professionalism

The first four categories are quite similar to those that one would use to evaluate closed source software. For the latter three, 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.

The term “Operational Software Characteristics” refers to those qualities of a software system that can be evaluated without access to the source code. It includes such areas as reliability, performance, scalability, usability, installability, security, and standards compliance. “Software Technology Attributes”, by contrast, involves access to the source code to review software architecture, code quality, and internal documentation. “Professionalism” refers to the management processes of the project, and covers such topics as project governance, maintenance of a project roadmap, and rapid response to reported issues.

As additional examples, vendor support, standards compliance, test coverage, and the presence of published books are characteristics that indicate a high degree of business 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. Thus, there is some natural variation among the rating scores for a specific open source component. In addition, the initial Quick Assessment Phase of the evaluation process allows an evaluator to search projects that meet certain criteria, such as the type of license (GPL, BSD, etc.) or the language(s) in which the project is written.

3. THE NEED FOR DATA

An important objective for the OpenBRR project is to use actual project data as part of the evaluation process, rather than relying solely upon subjective evaluation criteria. One difficulty with this goal is that the data is at a different level of detail from the evaluation criteria. For example, someone performing an evaluation may want to be sure that there is a sizeable user base and an active discussion forum. In the OpenBRR model, these qualities are grouped into the “Community and Adoption” area, but these items are not directly found in the data elements associated with a project. Instead, it is necessary to find elements that can serve as a proxy for the various evaluation criteria. For example, the number of messages posted per week on a project forum could be a useful metric that serves as a measure of the size and vitality of the project community. Similarly, the number of downloads can be a way to judge the size of the user base.

With this idea in mind, we turned to the FLOSSmole project (<http://ossmole.sourceforge.net>) [1.2] as a source of data that could be used in computing some of the OpenBRR scores. This was the first step in a larger project aimed at analyzing and creating a framework for data collection from multiple repositories. Some preliminary mappings of the FLOSS project data (including number of downloads, page views, bugs opened vs. closed, forum postings, etc.) to BRR metrics (such as functionality, usability, support, etc.) have been completed, and we report on that work in the remainder of this paper.

3.1 Using the FLOSSmole Data

The FLOSSmole dataset collects a variety of data on the FLOSS projects, including project functionality (also called ‘topic’), project OSS license(s) type, operating system, programming

language, natural language support, tracker information, developer information, etc [2]. For our data collection, we find the basic project information (name, identifier, project URL) and the project classification data very helpful.

FLOSSmole aggregates data from the following OSS repositories:

- sourceforge.net
- rubyforge.org
- freshmeat.org
- objectweb.org
- Free Software Foundation (fsf.org)

The FLOSSmole collector used an SQL interface (over HTTP) to query the flossmole.org database. Collecting data from flossmole.org using the SQL interface (over HTTP) is not very robust. The FLOSSmole database may contain as many as tens of millions of records in many tables and the response time is subject to how many other queries are running at that instant. Many queries for the latest snapshot of data timeout and have to be processed in chunks of 500 or 1000 records only. We eventually decided to download the database dump and use those instead of directly querying their database over HTTP.

We built an OpenBRR database schema to hold the various data items needed for calculating information about projects. This database is shown in the Appendix.

3.2 Data Collector Prototype

Multiple data collectors are used to gather the project related information required to calculate the Business Readiness Rating. Of these, the primary data collector is the one for FLOSSmole and the data collectors for project repositories (currently only Sourceforge.net).

The initial implementation of FLOSSmole data collector used FLOSSmole’s SQL interface over HTTP. After discussion with the FLOSSmole team, it was modified to use the periodic data export from FLOSSmole. Using a custom data parser and db loader, the upload process is very simple. However, the following aspects have been taken into consideration.

- Audit trail – of data uploaded is maintained and a copy of the uploaded data files is archived.
- Security – currently this operation can only be performed by the database administrator. Subsequently a set of authorized persons will have the capability to handle this. Having the audit trail will help in managing the process.
- Automation – though currently this is a manual effort, a more automated approach will be implemented. The first phase of automation would include downloading the new data files from FLOSSmole repository.

The data collectors for the project repositories use an HTML scraping technique. Based on the data collected from FLOSSmole, a unique URL is created using project short name (a.k.a. “proj_unixname” in FLOSSmole) and project integer id. This URL is then used to gather additional metrics on project access, project download, and project activity.

As other data collectors are implemented, the following concerns have to be addressed;

- Data Synchronization – Currently, it is not very difficult to synchronize data between FLOSSmole and project repositories. As more varied collectors are implemented (e.g. collecting blog information on any project), the data synchronization issue needs a special check.
- Data insufficiency – Not all data collectors will return the same quality of data. Provision has to be made to handle data holes and data insufficiency.
- Resistance to web crawling – Project repositories are sensitive to Internet bots due to obvious reasons. Currently, in the prototype phase, the crawling is kept to a minimum to avoid denial of access from the repositories.

4. WEB APPLICATION PROTOTYPE

We created a prototype Web-based application to allow users to search the resulting dataset to find open source projects that meet criteria specified by the users. This prototype is intended as a way for us to continually improve the user’s experience in identifying open source projects that meet his requirements.

In the implementation phase, the UI was slightly changed to follow the BRR framework more closely.

1. Users will select a project topic (e.g. CRM, database, editor, algorithms, hardware, networking, etc.) in which they are interested. Optionally, they may select one or more filters for their initial “Quick Assessment”. The filters currently available are;
 - a. By “License Type” – License under which the FLOSS product is released.
 - b. By “Operating System” – Platforms on which the FLOSS product runs.
 - c. By “Programming Language” – In which the FLOSS product is implemented

A screen shot of this process is shown in Figure 1. Users can make selections in any of the six boxes for any selected software category. This step has the effect of creating a “short list” of candidate projects in that category.

2. The user then selects the “Usage Setting” for the FLOSS product. The available choices are “Mission Critical”, “Routine Use”, “Internal Development/ISV”, and “Experimentation”. The users will also see a list of projects that match the above filtering criteria. For example, if a user selects “Mission Critical”, then only projects with a development status of “Production/Stable” will be shown.
3. Then the user sees the final report for the top projects. The user also gets a summary of the BRR rating for all the projects that were selected by the filter. A screen shot of this report (using an incomplete sample data set) is shown in Figure 2. This matrix shows the selected projects on one dimension and the BRR categories on the other, including the weightings for each, either the default values or specific weights chosen by the user.

The scores are computed based on weights assigned to the various evaluation categories. Standard weightings for the four predefined usage settings are shown in Figure 3. Users can revise the weightings for these categories and can also define new usage settings with their own preferred weightings.

The values retrieved from the FLOSSmole dataset for a project is mapped to one or more of the evaluation criteria., FLOSSmole data about licenses, operating systems, implementation languages, and project status are used in the initial screening. In the subsequent stage, some of the developer and project data is used. For example, information on the number of developers is a factor in weighing project activity level. The number of open and closed bugs is used in analyzing the project community and the software technology attributes.

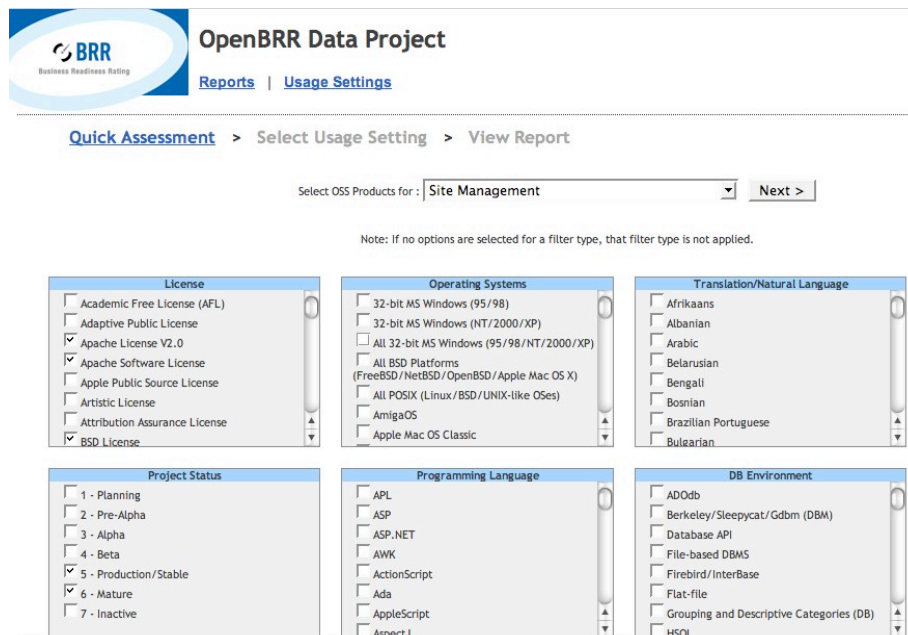


Figure 1 – Quick Assessment Filtering

	Routine Use	ISPCongfig Hosting Control Panel	Uniform Server	Ganglia	The JGenerator	Seagull PHP Application Framework
Functionality	25.0 %	0.0	0.0	0.0	0.0	0.0
Usability	15.0 %	0.0	0.0	0.0	0.0	0.0
Quality	15.0 %	0.0	0.0	0.0	0.0	0.0
Security	5.0 %	0.0	0.0	0.0	0.0	0.0
Performance	10.0 %	0.0	0.0	0.0	0.0	0.0
Scalability	5.0 %	0.0	0.0	0.0	0.0	0.0
Architecture	5.0 %	0.0	0.0	0.0	0.0	0.0
Support	5.0 %	0.0	0.0	0.0	0.0	0.0
Documentation	5.0 %	0.0	0.0	0.0	0.0	0.0
Adoption	0.0 %	5.0	5.0	5.0	4.0	4.0
Community	10.0 %	5.0	5.0	5.0	4.0	4.0
Professionalism	0.0 %	0.0	0.0	0.0	0.0	0.0
BRR Rating	100.0 %	0.5	0.5	0.5	0.4	0.4

Project	BRR Rating	Functionality	Usability	Quality	Security	Performance	Scalability	Architecture	Support	Documentation	Adoption	Community	Professionalism
aeonserv	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	0.0
App_Site_Content_Management_System	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	0.0
CCC	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	0.0
ClanSphere (formerly BXCP)	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	3.0	0.0
Destiney Scripts	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	0.0
Dream Tags	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
FMPP - FreeMarker-based PreProcessor	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	3.0	0.0
Frikki - a Java Wiki	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	3.0	0.0
Ganglia	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	5.0	0.0
GeoScope	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	0.0
Hoople	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	0.0
Imbrium	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0

Figure 2 – Mockup of results from BRR evaluation

Usage Setting	Functionality	Usability	Quality	Security	Performance	Scalability	Architecture	Support	Documentation	Adoption	Community	Professionalism	Actions
Mission Critical Use	25.0 %	10.0 %	15.0 %	15.0 %	15.0 %	5.0 %	5.0 %	5.0 %	0.0 %	0.0 %	0.0 %	5.0 %	Edit Delete
Routine Use	25.0 %	15.0 %	15.0 %	5.0 %	10.0 %	5.0 %	5.0 %	5.0 %	5.0 %	0.0 %	10.0 %	0.0 %	Edit Delete
Internal Development / ISV	15.0 %	10.0 %	15.0 %	10.0 %	0.0 %	0.0 %	10.0 %	10.0 %	15.0 %	0.0 %	15.0 %	0.0 %	Edit Delete
Experimentation	20.0 %	10.0 %	10.0 %	5.0 %	5.0 %	5.0 %	10.0 %	10.0 %	15.0 %	5.0 %	5.0 %	0.0 %	Edit Delete

Figure 3 – Weightings for predefined BRR Usage Settings

5. CURRENT STATUS

Both the BRR project and its use of FLOSSmole data are still in a very early stage. Among the next steps are:

1. *Normalizing the data.* There is often a lot of fluctuation from month to month in data such as the number of downloads, number of bugs filed and resolved, etc. The BRR rating should depend on more stable metric values (e.g. monthly average for the last 6 months) for better stability and maturity in reporting. The number of months could be debated (between 2 to 6 months) to balance between being overly affected by data fluctuations (for very small duration) and not being responsive to changes (for larger durations)
2. *Means for collecting subjective metrics.* OpenBRR.org should be able to collect more subjective metrics, ones that are hard to determine from the available project data and require better

algorithms to calculate or human input. This could be either by partnering with other FLOSS data sites like Ohloh.com and Krugle.com or by gathering this data at the OpenBRR.org site itself by allowing individuals to submit subjective reviews in addition to obtaining the quantitative data. There are some other collected reviews and metrics on open source, including SourceKibitzer (<http://www.sourcekibitzer.org>), and we plan to review these for additional ideas on how to collect and present both qualitative and quantitative data.

3. *Support for more FLOSS projects.* There is a need to expand data collection from a variety of other project repositories. The challenges involved are:
 - Developing support for incomplete data. Not all project repositories have the support to provide the same set of metrics.
 - Developing support for dedicated sites (forges). FLOSS projects like Eclipse, NetBeans, MySQL,

and others have dedicated hosting sites. Support for product with dual-license strategies.

- Access to additional sources of project data, such as Ohloh (<http://www.ohloh.net>)
4. *Improving the web application.* A complete BRR evaluation uses not only the quantitative information, but also assessment of its functionality and informal reviews. The underlying database schema can be extended to include this additional information.
 5. *Extending the model.* Many users are not interested in just a single open source project, but in an integrated set of them. Many open source applications use open source infrastructure as the underpinnings of the application, and want to evaluate the entire set of components as one.

In summary, OpenBRR must be viewed as a federation of repositories, collecting data from numerous sources and presenting a synthesis of results. The effort must be ongoing, reflecting the continuing appearance of new repositories and changes to existing ones.

Through seamless data collection from varied sources (project repositories, data analysis sites, and specialized search engines) and by incorporating intelligent analysis, OpenBRR.org can be developed as a authoritative site for recommendations on FLOSS projects, using the FLOSSmole data and more.

6. REFERENCES

- [1] Wasserman, A.I., Pal, M., and Chan, C., "The Business Readiness Rating Model: an Evaluation Framework for Open Source", Proc. Of Workshop on Evaluation Techniques for Open Source Software, 2nd Int'l. Conf. on Open Source Systems, Como, Italy, June, 2006.
- [2] Conklin, M.S., Howison, J, and Crowston, K. Collaboration using OSSmole: a repository of FLOSS data and analyses. Proc. Of Workshop on Mining Software Repositories, Int'l. Conf. on Software Engineering, St. Louis, MO, May, 2005.

APPENDIX: Summary of OpenBRR Database Schema

Data Tables

Table name	Description
forges	Contains data about the various forges, e.g. sourceforge.net, objectweb.net, fsf.org, etc. This data is used internally by the data collectors and web application.
projects	Contains project data. It has columns to hold various project metric data.
troves ¹	Contains meta-data about the various project attributes like licenses, programming languages, operating systems, status, etc. that are stored in different tables.
licenses	Contains the open source license data
programming_languages	Contains the list of programming languages
operating_systems	Contains the list of operating systems
status	Contains the list of project status
topics	Contains the list of (functional) topics for the projects, e.g. "Networking", "Editors", "Databases", "CRM", etc.
usage_settings	Stores the "User Setting" information. Examples are: "Mission Critical Use", "Routine Use", "Experimentation", along with user-definable settings.

Relation Tables

Table name	Description
project_topics	Contains the m x n relation between projects and the topics that they are classified under.
project_licenses	Contains the m x n relation between projects and OSS licenses under which they are released
project_prog_langs	Contains the m x n relation between projects and programming languages in which they are implemented or provide an API for.
project_os	Contains the m x n relation between projects and OS platforms they support

¹ Trove is a term borrowed from sourceforge.net. It refers to a project attribute like license information, programming language used, support for various (natural) languages, support for platform OS, etc. or the topic(s) under which the project is classified.

