Open-Source Project Comparison

# Introduction

The goal of this activity is to explore the process of comparing several open source projects and selecting the best alternative for a particular situation. The evaluation approach is built on the Project Evaluation Rubric used in prior activities.

The overall process for this comparison has the following steps:

1. Define the reason why you are performing the comparison.
2. Adjust the rules for setting the Level for each evaluation criterion based on your reason for comparing the projects.
3. Create a weighting scheme for the criteria in the Project Evaluation Rubric.
4. The resulting rubric, customized for the purpose of your evaluation, will be applied in the next assignment.

This activity covers preparing to do the comparison, steps 1-3 above.

The evaluation is not about product functionality. While comparing product function is a normal consideration in product evaluation, that part is outside the scope of this assignment. For this assignment, assume that the two products provide equivalent functionality.

### Date: 2022-02-02T10:46:00 Group ID: Team-Blue

Complete the table below to assign a role to each team member. If you have only 3 people, combine Speaker & Reflector.

|  |  |
| --- | --- |
| **Team Role** | **Team Member Name** |
| **Recorder**: records all answers & questions, and provides copies to team & facilitator | **Peter James Mangelsdorf** **pjm349** |
| **Speaker**: presents answers to the class and talks to facilitator for the team | **Charles Porter cap399** |
| **Manager**: keeps track of time and makes sure everyone contributes appropriately | **Eric Savoy ets43** |
| **Reflector**: considers how the team could work and learn more effectively | **Jay Patel jp3592** |
| **Participant** | **Hoang Nguyen hn374** |

Reminders:

1. Record the time whenever your team starts a new section or question.

2. Write clearly and with enough detail that your responses can be easily understood.

# A. Defining the Reason for the Evaluation

## Start time: 2022-02-0210:48:00

You may have different reasons for evaluating FOSS projects. Possible reasons include:

a) You are looking for a product in a particular application area. If that is so, you should also consider why you are looking at that application. You might just want a product that you can use, or you might expect to modify a product and use the modified result. You might also be looking for a FOSS community that you can join to work in an application area.

b) You might not care about the product application area, but rather be interested in products that use a particular technology (programing language, framework, etc.) that you want to learn.

There may be other reasons you are looking at FOSS projects in addition to these. But whatever the reason, it is likely to change how you evaluate a FOSS project.

## Use Scenario - Describe the reason you are looking at open source projects:

* We want an open source course evaluation system. We are looking at other projects to help aid in our own development. We want to see what projects already exist, what they do well, what they could improve upon, and whether any already meet the criteria of our own project.
* The final product is intended to be shared across many institutions, hence a closed-source model may make contributing from multiple institutions difficult.
* It is also easier to evaluate the quality of open source project code without requesting source code privileges from closed-source maintainers.
* Level of Activity/Community is important towards maintaining a project, especially an obscure product like this. Unfortunately, there does not appear to be a dedicated “Course Evaluation” community, hence a “static” codebase may be acceptable.

# B. Adjust the Rules for Criteria Level Scores

## Start time: 2022-02-02T11:00:00

Keeping in mind the evaluation reason you defined above, you have decided to start evaluation by scoring each candidate project using the Project Evaluation Rubric. Review the general rules for assigning a Level (0-2) for each criterion and refine those rules for your use scenario.

Discuss and answer the following questions:

## 1. Are any of the evaluation criteria “showstoppers” for your scenario? That is, is there a particular score for any criterion which, if not met, will mean that the project should be dropped from consideration? If yes, identify the criterion and your reasoning.

* Invalid Licensing: If the license is either **missing**, **unusual**, **explicitly all-rights-reserved**, or otherwise, then evaluation should stop
* Bad Technology: If the code follows **obscure**, **old**, **explicitly proprietary**, **poor standards**, or otherwise developer-pain-inducing technology, evaluation should stop. Also, if the product uses technology that no member on our team is familiar with, it is most likely a show stopper.
* Product Size: **Small “student” projects** are especially common. **Comprehensive but hard to edit school-management-systems** that happen to include a course evaluation module are also common. Neither size is especially desirable.
* Other: The other criteria have less impact because the product-space is so small that any work done will largely need to be new (Issues and Level of Activity don’t matter as most can be created by new development). ***Note*** that this assumes we are looking to more-or-less build a new project. There are some course-evaluation projects, to which these criteria would matter more.

## 2. Do any of the general guidelines for assigning Levels to the criteria need adjusting for your use scenario? If yes, define a new rule for assigning a Level for that criterion. Note your rules in Table 1.

* Ease of Modification: How easily can new components be added?
* Data Privacy/Safety: How seriously did previous contributors take security? Is it openly acknowledged? Are there tests present for common attacks? (Fuzzing, Invalid Access).

**Table 1. Rules for Assigning Levels to Criteria Based on the Use Scenario**

|  |  |
| --- | --- |
| **Evaluation Criteria** | **Notes on Assigning Levels to Criteria** |
| Licensing | Score a 2 if the license indicates the project is free or open source. Score a 1 if the license is free or open source, but overly restrictive. Otherwise score a 0. |
| Technology | Score a 2 if project uses desired technology. Score a 1 if project uses technology that is acceptable, but not ideal. Score a 0 if project uses obsolete technology. |
| Level of Activity | Score a 2 if there has been activity on the project for each quarter of the last year. Score a 1 if some of the previous year’s quarter were active. Score a 0 if there has been no activity in the past year. |
| Number of Contributors | Score a 2 if the project consists of 10 or more contributors. Score a 1 if the project consists of between 3 and 10 contributors. Scare a 0 if the project has less than three contributors. |
| Product Size | Score a 2 if the product is robust enough to support multiple institutions. |
| Issue Tracker | Score a 2 if the project has issue tracking features available for existing and new contributors. |
| New Contributor | Score a 2 if the project has resources on how to contribute. |
| Community Norms | Score a 2 if there is a clear code of conduct. Score a 0 if there is a considerable amount of rude and undesirable behavior in the community. Score a 1 if there is little or no undesirable behavior, but no code of conduct. |
| User Base | Score a 2 if product has more than 10 users (institutions). Score a 1 if the product has 2 users (institutions). Score a 0 if 1 or less users. |

**Report out** - The speaker will report out for the team.

# C. Define Weights for the Criteria

## Start time:

Based on your use

scenario, some criteria may be more important than others. One way to express that is to assign a Weight to each criterion and multiple the Level by that Weight.

Discuss and answer the following questions:

## 1. Are any of the evaluation criteria unimportant for your scenario? That is, the weight should be low for this criterion for your scenario. If yes, identify the criterion and your reasoning.

* All criteria have been determined to be important, however, given how few projects there are, we assume we will need to fork and start our own project. With that in mind...
* Community Norms: Most projects are static, there is no community.

## 2. Identify and rank the three or four criteria that you think are most important for your evaluation. You should also note your reasons for your choices.

* Technology: Must be sustainable technology, not old or obscure or hard to use
* Product Size: Whole-System projects should be avoided, as well as “student-demo” projects
* Licensing: Needs to be open-source to contribute to it

## 3. Assign a Weight (1-3) for each criterion. Assign 1 for if the criterion is unimportant, 3 if the criterion is among the most important, and 2 for anything in-between. Record your weights and reasons in Table 2 below.

**Table 2. Weights Based on the Current Evaluation Use Case**

|  |  |  |
| --- | --- | --- |
| **Evaluation Criteria** | **Weight** **(1-3)** | **Notes** |
| Licensing | 3 | It must be easy to contribute to. If the license makes that difficult, getting future contributors will be difficult. |
| Technology | 3 | No outdated technology. Also no obscure technology. A focus on persistent storage for many years should be included. |
| Level of Activity | 2 | Less important but desirable. A few reviewed projects have a community around them which we should bias towards. |
| Number of Contributors | 2 | Less important but again desirable. Prefer projects that have a significant number of contributors if they exist. |
| Product Size | 3 | Must not be too large or too small. |
| Issue Tracker | 2 | Less important, but desirable. Good for determining what state the application is in. |
| New Contributor | 2 | Less likely to exist. Will prefer, but assuming that this niche product category is unlikely to attract less experienced developers. |
| Community Norms | 2 | Less important, as fewer participants in this niche category, can spend more time on negotiating differences. |
| User Base | 2 | Less likely to exist. Again, if it exists (which it appears so on some projects), then pursue it. |

**Report out** - The speaker will report out for the team.

Copyright 2022 Gregory W. Hislop. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/