https://oscm.disqu.de

Open Source Community Monitoring

Version 1.0, 30.09.2025

The DISQU Open Source Community Monitoring (OSCM) provides an overview of how active and stable the community of an Open Source project is. Open Source software is a crucial part of achieving digital sovereignty and ensuring longevity and flexibility of IT setups. But one has to keep in mind that Open Source software also requires regular maintenance and continuous development, which usually are carried out by the community. Users of Open Source software therefore have an interest in the stability of the projects. For the community it is also relevant to understand which projects have a very active and stable community and which projects are in greater need of additional support.

This Monitoring aims to provide insights into this aspect by analyzing publicly available data of Open Source projects and aggregating and comparing them with one another. As a result, it delivers an indication of how active and stable the community of a given project is.

DISQU is monitoring Open Source projects and publishing their scores on https://oscm.disqu.de.

1. General design

To get an overview of the community of a project, several metrics are collected and scored individually. This scoring is based on values ranging from 0 (very poor) to 5 (excellent). The overall rating for the project is calculated as the average of all these metrics. In this way the Monitoring provides an easy-to-interpret value to quickly assess the status of a project, while also offering detailed insights into the individual metrics.

For many of the metrics, data is collected over a defined period of time in the past (for example, the last 90 days). To perform this data filtering, not the last 90 days starting from today are considered, but the last 90 days since the last data collection was completed.

2. Commit based metrics

Commits are individual changes made to the source code of a project. Each commit records what was modified, when it was done, and who contributed the change. Because commits represent the conducted development work, they serve as a direct indicator of how actively a project is maintained. A steady flow of commits usually means that a community is engaged, continuously improving the code, fixing bugs, and adding features. On the other hand, long periods without commits may suggest reduced activity or limited community involvement.

2.1. Commit numbers

> This metric takes a look at the number of commits produced within a project. As this is influenced by the size of the project as well as the working style of the contributors, it is hard to define an absolute number of commits required to identify a community as good or stable. Thus, we analyze the historic development of the number of commits: nine time frames with a length of ten days each are created, and for each of these windows the number of commits is counted. These data points are interpolated linearly, so that a rise or decent of commit activity can be seen.

In the following example we can see that there are period of times with higher and times with lower commit activity, but the overall trend is going upwards:

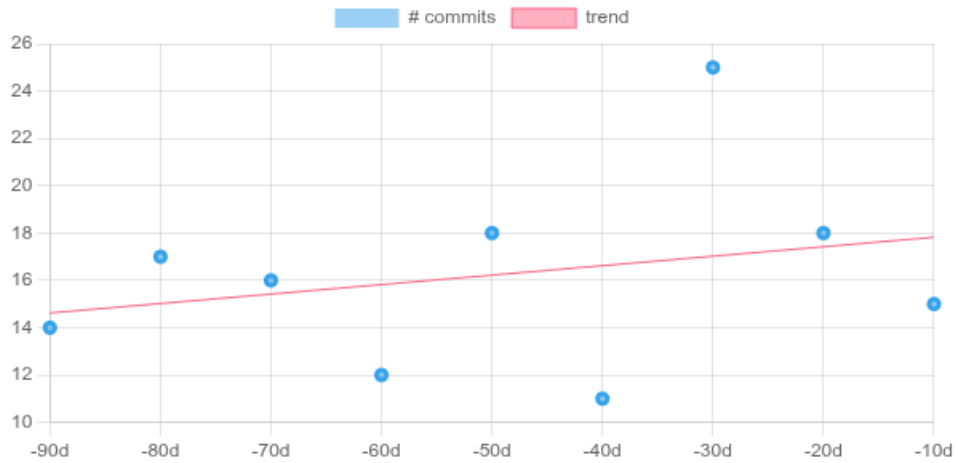

Figure 1: Commits per 10-day interval over the last 90 days with regression line.

For scoring we take into account the slope s of the linear regression:

Score	slope s
0	$s < -0.10$
1	$-0.10 \leq s < -0.075$
2	$-0.075 \leq s < -0.05$
3	$-0.05 \leq s < -0.025$
4	$-0.025 \leq s < 0.00$
5	$s \geq 0.00$

2.2. Committers

Besides the number of commits, the number of people creating them (committers) is also important, as projects developed by a single person or a very small group of people are at high risk of not being continued. Here we face the same problem: it is not easy to say how many committers a project should have, so we again take a look at the historic development (as we did it for the commits). The assumption is that the available committers were enough to maintain the project in the past, so they will probably be enough people to continue its maintenance.

The regression is built for the number of committers during:

- the last 0 to 30 days
- the last 31 to 60 days
- the last 61 to 90 days
- the last 91 to 120 days

Score	slope s
0	$s < -0.10$
1	$-0.10 \leq s < -0.075$
2	$-0.075 \leq s < -0.05$
3	$-0.05 \leq s < -0.025$
4	$-0.025 \leq s < -0.00$
5	$s \geq 0.00$

2.3. Elephant in the room

For all projects it might be a significant risk if most of the commits are created by a single person: this person probably has accumulated most of the knowledge and is not easy to replace. On the other hand this person might lose interest in the project or just not have the time to maintain it anymore. When an individual performs the maintenance of a Open Source project for a company, it is possible that the company will discontinue its support after that individual leaves the company or takes on new responsibilities.

If an individual is doing most of the work, this person is sometimes referred to as 'the elephant in the room'.

For this metric we take a look at all commits of the last 120 days. The committers of these commits are sorted in descending order based on their share on the total number of commits. Based on this, the number of committers required to produce more than 50% of the commits is counted.

Score	committers needed to reach $\geq 50\%$ of commits
0	1 committer
1	-
2	2 committers
3	-
4	3 committers
5	≥ 4 committers

3. Issue based metrics

Issues typically represent bug reports, feature requests, or tasks related to the project. They allow a structured way for users and contributors to communicate problems, suggest improvements, and coordinate development work. Using is-

sues in open source projects not only helps organize and track development tasks, but also increases transparency and makes it easier for newcomers to contribute to the project.

The volume and handling of issues provide valuable insights into how active, responsive, and organized a project's community is.

3.1. Created

The number of newly created issues is measured across four intervals (0–30, 31–60, 61–90, 91–120 days) and analyzed through linear regression.

Issue creation serves as an indicator of community interaction and responsiveness. A flat slope reflects stable activity and balanced reporting. A positive slope indicates growing engagement, often linked to an expanding user base or active development. Here should be noted that a positive increase in the number of issues can also indicate emerging problems or conflicts within the community, and therefore the score for an increased amount of issues is the same score a community can get for a stable amount of issues. A negative slope signals declining activity, which may suggest reduced user involvement or weakening project support.

Score	Slope of the linear regression line
0	< -0.1
1	< -0.075
2	< -0.05
3	< -0.025
4	< 0.0
5	≥ 0.0

3.2. Closed

Analyzing the ticket closing activity within a given time frame provides valuable insight into community stability. We consider the percentage of closed issues relative to all issues created in that period. A consistently high closure rate indicates that the community is actively maintaining the project, addressing problems, and preventing backlog. Low or declining closure rates, by contrast, may signal reduced activity or limited resources, which can weaken long-term trust and sustainability of the open-source project.

Score	Percentage of closed issues relative to all issues created in that period
0	$\leq 2\%$
1	$> 2\%$
2	$> 5\%$
3	$> 10\%$
4	$> 25\%$
5	$> 30\%$

3.3. Comments

This metric evaluates how evenly issue discussion is distributed among contributors in an open source project. The score is derived from the highest individual share of comments, expressed as a percentage of all comments in a 90-day window. A lower score indicates that one contributor is responsible for a disproportionately high percentage of comments, suggesting weaker community quality, whereas a higher score reflects more balanced participation, typically associated with a stronger and healthier open source community.

Score	Percentage of highest individual share of comments
0	$> 50\%$
1	$> 40\%$
2	$> 30\%$
3	$> 20\%$
4	$> 15\%$
5	$\leq 15\%$

3.4. Response Time

This metric quantifies issue responsiveness by measuring, in hours, the time from issue creation to the first non-reporter, non-bot comment across issues opened in the preceding 90 days. Automated bot comments, such as those generated for issue triaging, template enforcement, or dependency updates, are excluded because they do not reflect substantive human engagement and would artificially lower measured response times. It derives a score from both the mean and median response times.

Score	Mean time between time of issue creation and first comment
0	$\geq 360h$
1	$< 360h (15 \text{ days})$
2	$< 168h (\text{one week})$
3	$< 96h$
4	$< 48h$
5	$< 24h$

Score	Median time between time of issue creation and first comment
0	$\geq 168h$
1	$< 168h (15 \text{ days})$
2	$< 72h (\text{one week})$
3	$< 20h$
4	$< 7h$
5	$< 3h$

The response time score is calculated as the average of the median-based score and the mean-based score, since both indicators capture different but equally important aspects of responsiveness: the mean reflects overall response behaviour, while the median highlights the typical experience for most contributors. Higher scores indicate faster and more reliable responsiveness, which is typically associated with a healthier and more sustainable open-source community.

4. Merge Request based metrics

A merge request (also called a pull request) is a developers proposal to integrate changes from one branch into another, typically reviewed and discussed by peers before being merged into the main codebase. Merge Requests directly indicate that people are working on the project by providing bug fixes or new features.

4.1. Created

A larger number of created Merge Requests directly shows a high level of contributions. This can be counted by retrieving the data from code collaboration platforms. However, for larger projects a higher number of new merge requests can be expected. Thus, the amount of merge requests created in the last 90 days is compared to the number of lines of code.

Score	average amount of Merge Requests per 50 000 lines of code
0	≤ 1
1	> 1
2	> 2.5
3	> 5
4	> 10
5	> 50

4.2. Accepted

A high number of merge requests that are accepted is an indicator that contributions to the project are usually of high quality. Furthermore, it shows that the community is inclusive and welcoming rather than not rejecting contributions.

Like the number of created merge requests, this metric is calculated in relation to the number of lines of code. The basis is all merge requests that where open or created during the last 90 days.

Score	share on relevant merge requests that got accepted
0	$\leq 5\%$
1	$> 5\%$
2	$> 15\%$
3	$> 45\%$
4	$> 65\%$
5	$> 75\%$

4.3. Rejected

A high number of rejected merge requests indicates either contributions of lower quality or maintainers unwilling to accept contributions from others.

Like the number of created merge requests, this metric is calculated in relation to the number of lines of code. The basis are all merge request that were open or created during the last 90 days.

Score	share on relevant merge requests that got rejected
0	$> 25\%$

Score	share on relevant merge requests that got rejected
1	$\leq 25\%$
2	$\leq 20\%$
3	$\leq 15\%$
4	$\leq 7\%$
5	$\leq 1\%$

4.4. Reviews

In projects with a stable and active community, merge requests receive one or more reviews. Ideally, those reviews are created by different people.

The basis is all merge requests that are still open or were closed during the last 90 days.

This data is interpreted in two metrics:

4.4.1. Number of created Reviews (Reviews)

This is the number of reviews created for the relevant merge requests, divided by the number of merge requests. In this way, it shows the average number of reviews an open merge request usually receives.

Score	reviews/merge request
0	≤ 0.25
1	> 0.25
2	> 0.5
3	> 1
4	> 1.4
5	> 1.7

4.4.2. Number of persons reviewing (Reviewers)

In many projects code reviews are done by a small group of people. But ideally, there are multiple reviewers performing the reviews, as this shows that deeper knowledge of the codebase is shared among several individuals.

For this metric, the total number of reviewing individuals is divided by the number of reviews.

Score	Reviewers per review
0	< 0.06
1	≥ 0.06
2	≥ 0.1
3	≥ 0.25
4	≥ 0.5
5	≥ 0.6

4.5. Duration

If merge requests get approved or closed in a short time span, this indicates an active community.

We analyze the time between creation and closure of merge requests that were accepted or declined during the last 90 days. The median of these values is used for the score.

Score	Median time to close
0	$>$ 10 days
1	$\leq$ 10 days
2	$\leq$ 7 days
3	$\leq$ 4 days
4	$\leq$ 2 days
5	$\leq$ 1 day

5. Conclusion

OSCM is a monitoring approach evaluating community activity of Open Source projects using three core indicators: issues, commits, and merge requests. These dimensions capture user interaction, development progress, and collaborative contributions. The individual metrics are aggregated, and their average is calculated to produce an overall score for each project.

The advantage of this approach is that it provides a clear, comparable measure of a communities stability and engagement, while also allowing for deeper insights into specific aspects. Future versions of the framework will expand the assessment to include additional factors, enabling an even more comprehensive evaluation of open source communities. For example, there are plans to incorpo-

rate the size of the OS project and the programming language more strongly into the individual metrics, thereby refining the assessment.