As part of the Retro Funding: Dev Tooling Mission, the Collective will experiment with collecting data on the utility of Dev Tools to improve the evaluation algorithm. This process is heavily inspired by Deep Funding's approach to rating dependencies.

Motivation

While we have robust data to quantify the usage of Dev Tools by Onchain Builders, we lack sufficient insights into the utility these tools provide. By asking Onchain Builders to rate their dependencies, we aim to identify metrics that best measure utility. This initiative serves as a pilot to explore data collection methods that can enhance the evaluation algorithm's accuracy.

Rating Process

Participants will compare the importance of dependencies for specific Onchain Builder projects usingPairwise. For example, they will answer questions like: "Which dependency is more critical to [Onchain Builder's] success?"

- Comparison Mechanism

: A slider allows participants to weigh the relative importance of two dependencies.

- Justification

: Participants must provide a brief explanation for their ratings.

- Timeline

: Ratings will be collected from February 3rd to February 17th.

[

image

2048×1161 186 KB

](https://europe1.discourse-cdn.com/bc41dd/original/2X/b/bb1bec3438bdb12deb30eb4c5740ec2e6da74918.jpeg)

Below you can find the list of Onchain Builders, whose dependencies will be rated initially. This initial set was select based on a diversity of use-cases and significant onchain activity.

List of Onchain Builder Projects

1. Velodrome
2. Synthetix
3. Sushi
4. Smol Dapp
5. Boost
6. Sound.xyz
7. Across
8. Basepaint
9. Agora
10. Superfluid
11. Ethereum Attestation Service
12. Giveth

List of dependencies

Up to 30 dependencies of each Onchain Builder will be surfaced in the rating process. The complete methodology developed by OSO for this experiment is described below.

All other things being equal, the list of dependencies attempts to prioritize web3 dependencies, as opposed to more general web2 dependencies, in order to give a more realistic approximation of projects likely to appear in the Dev Tooling round.

1. Identify relevant repos from Onchain Builders

. OSO looked up any public repo in the project's GitHub namespace that (i) has been updated since 2024-07-01, (iii) is based in Rust, Solidity, or TypeScript, and (iii) has more than five stars. This generated a set of 90 repos for the experiment and at least 2-3 per project.

1. Fetch a complete list of package dependencies for each repo.

OSO uses GitHub's public SBOM (software bill of materials) data to identify all direct (and indirect) repo dependencies. Here is an example of the SBOM for the eas-contracts repo, which includes more than 800 distinct packages. OSO's most recent SBOM snapshot was taken on 2025-01-25.

1. Lookup the owner of each package dependency

. OSO uses a public dataset called deps.dev to map package versions to maintainer repos. As packages can change owners over time, we've extracted the name of the current owner of the package. The same data can be validated directly from the package registries, i.e., npm for JavaScript/TypeScript and crates for Rust. Here is an example for solady.

1. Derive metrics for each package maintainer repo

. As the same repo may own more than one package, OSO aggregates all relevant packages into an array and pulls metrics and metadata for each of the maintainer repos included in the dependency graph. Metrics include number of stars, number of package dependents in the full OSO dataset, and number of package dependents in the RF4 cohort. Metadata includes language, date of last update, license, and whether the repo is associated with a project that has applied for Retro Funding in the past.

1. Filter out maintainers with no recent activity.

In order to do some initial pruning of the dependency graph, we only include projects that have had at least three months of developer activity since 2024-07-01. Although this may exclude some critical dependencies, it is intended to result in a more realistic set of dev tools that might apply for Retro Funding.

1. Prioritize the top 30 repos for each onchain builder

. In order to keep the pairwise matching work reasonable for participants, we limit the number of dependencies to just 30 per project. (Most projects had more than this, even after applying the filters described above.) First, we prioritize any repo associated with a project that has applied for Retro Funding in the past. Next, we rank repos based on the degree to which they appear more commonly in RF4 builder dependencies than all projects in the OSO dataset. For Onchain Builders with lots of dependencies, this is intended to filter out some of their more generic web2 dependencies. Finally, we combine these two lists in order and truncate at 30 items per project.

1. Serialize a subset of all data for Pairwise.

OSO exports the data in the standard expected by the Pairwise API.

You can find the full methodology implemented by OSO here including CSV and JSON exports of the data.

Voter Groups

1. Onchain Builders

: 13 maintainers of the listed projects will be invited to rate their own dependencies.

1. Experts

: 30 community members will be randomly selected based on their GitHub trust scores. These ratings could help counterbalance potential biases from maintainers.

1. Open Participation

: The rating process will be open to anyone. Cohorts will be analyzed based on GitHub activity and reputation within the Collective.

How this data will be used

The collected ratings will inform the identification of metrics best suited to evaluate the utility of developer tools. This process mirrors Deep Funding's data challenges, where participants develop predictive models based on the ratings.

As a pilot, this initiative explores how human input can refine the evaluation algorithm. While the findings may not directly inform algorithm updates, they will provide valuable insights into future data collection methods.