In this post I will try to share some points that I take away from the process of participating as a reviewer

and appeals reviewer

in RPGF4.

In a second post (below) I will share my reflections on the voting process

.

Preconditions

Round scope

The aim of RPGF4 is to reward onchain builders who have deployed contracts to the Superchain and contributed to the success of Optimism by increasing demand for blockspace and driving value to the Collective.

Metrics-based evaluation

Along with a more narrow focus, this round also differed fromthe previous round in that the voting design prescribed a new form of automated, metrics-based evaluation.

Details on the thinking behind the metrics-based design can be foundhere, and I would like torepeat what is pointed out in that document:

A metrics based voting experience is only viable for a subset of impact today and can't be applied to evaluate all contributions to the Optimism Collective.

At the same time, an objective and metrics-based approach - for better and for worse - rules out subjective human judgement at times when it might be tempting to use it. There have definitely been times during this round when I felt frustrated about this.

[

1128×566 88.7 KB

](https://europe1.discourse-cdn.com/bc41dd/original/2X/1/1bc528d84302e615697feb33320d30db213c3aa5.png)

With that said, I appreciate the idea of smaller, more targeted rounds and the possibility to experiment with different ways of evaluating impact.

Impact definition

I'm glad that the decision was made to use the Foundation's fallback definition of "profit = impact", implying that no previous grants would be subtracted from this round's OP allocation.

While I do think that previous funding should ideally be taken into account, I believe that the road towards formally defining 'impact' and 'profit' is still long, and we should not rush it or pretend that we are already there.

Experimental attitude

As mentioned, I value the experimental attitude behind Optimism's RPGF.

Part of this, it seems to me, is to accept the possibility of failure.

While this round does have some features that I find problematic, it is essential in any experiment to allow things to run their course and not accidentally hide possible design flaws by trying to apply common sense on an individual basis.

Review process

I spent quite a lot of hours on the review process, as a reviewer and appeals reviewer.

Two main things, I think, could relatively easily be improved in future rounds: More clarity around what reviewers need or need not check, and the software tools available to reviewers.

What must reviewers check?

As reviewers, we were offered a handy reviewer guide. But while this document contained just a short list of fairly specific things to look for ("What to review projects on"), reviewers are also expected to ensure that applications comply with the general application rules, and it is my impression that this left quite a bit of room for subjective interpretation - and

uncertainty.

Some specific points around which there seemed to be ambiguity:

- Github repo age.

The guidelines say: “Does the repo hold no code or is it newly created, raising a red flag?” What is a red flag? A warning signal? A disqualification?

- Unverified contracts.

Reviewers are being asked to verify that the applicant has “made their contract code available in a public Github repo”. What happens when a contract shows as unverified on e.g. optimistic etherscan? How would a reviewer know if the deployed contract corresponds to the code in the Github repo?

- OSO slug.

This was arguably the most mysterious topic in the review process.

- Downstream contracts.

During the review process, it surfaced that the automatic verification performed by OSO considered anything downstream of a project’s stated deployer as in scope. While it makes sense that it could be impractical for some projects to explicitly list all of their contracts in their application, this does raise a number of questions with regard to the review process and the task of the reviewers. In particular, exactly what code do reviewers need to attest to having seen?

[

1620×1164 204 KB

](https://europe1.discourse-cdn.com/bc41dd/original/2X/d/d4a011768b7e4c79def4d05606b211dc50c28667.png)

Ideally, reviewers should be given instructions that clearly specify what action they must take

when they see this or that. And they need to know exactly what they need to look for

.

This requires the reviewers to understand the automatic verification process clearly

, such that they know what they can safely ignore.

Especially in a round like this with so much emphasis on objectivity and quantitative metrics, it seems important that eg. overlapping impact, transaction counts and transaction timing should be automatically checked …and possibly subjected to random sample tests by appointed badgeholders with the necessary skillset.

Other badgeholders could be asked to notify these specialists in case of doubt, but abstain from rejecting applications based on their own tests.

I think more clarity will lead to more consistency and delegation of responsibility. Feedback and questions from the human reviewers can be used to inform the automatic process - and the automatic process can apply the learnings consistently.

Reviewer tooling

Suggestions for improvement of the Charmverse UI that was used in the review process:

- Filtering.

Reviewers absolutely need to be able to easily see which projects are assigned to them and the current status of these projects. The same is true for appeals reviewers.

- Notifications.

It would be awesome if there was some kind of automatic notification in case a reviewer has missed a review, and also whenever new projects are assigned to a person.

- Rejection reasons.

We need the option to reject an application due to missing or incomplete information. It does not feel good to choose ‘spam’ or ‘deceiving badgeholders’ in cases where an application is simply incomplete for reasons that the reviewer can’t know.

- Reviewer notes.

The current interface offers 3 kinds of free text comments: a) The almost invisible ‘reviewer notes’ in the top right corner; b) the public comments on the bottom of the screen; and c) the explanation attatched to rejections. I think the reviewer notes are of little use at this point, and it might be better to remove them completely to avoid confusion. It would be nice to have the option of also offering an explanation when you accept an application. The public notes on the bottom of the page actually proved quite useful in one or two cases where an applicant pitched in and explained something.

- Application links.

Links to project website, Github, contracts etc. really should be shown as links and not just as text. Having to copy-paste these things into new browser tabs every time very, very quickly grows tiresome!

- Changing votes.

The flow around a reviewer changing his opinion could be better. In some cases, trying to change a vote also resulted in the cancellation of the four other reviewers’ votes - which was all the more problematic as due to anonymized user names it wasn’t always possible to see who you needed to contact if you had accidentally done this…

- Data integrity.

It is really important that reviewers can see exactly what applicants put in their applications. In a few cases, bugs in the UI prevented reviewers from accurately seeing parts of the application data. Applications were mistakenly being rejected because of this until the bugs were discovered.

In a future round, maybe a few reviewers could be employed to test out the software before the start of the actual review process?

It would also be good to allow/request applicants to review their own applications in the same UI(s) that reviewers and voters will be using - that way, any data integrity issues are much more likely to be discovered early on.

Also, maybe some sort of collaboration could happen between Charmverse and Retrolist.app. In this round, I would say that Retrolist did a better job at reliably mirroring the application data and providing helpful links etc. But reviewers still had to use Charmverse because that’s where the voting functionality was.