I thank the reviewers for their kind and constructive comments. I hope they agree that incorporating their suggestions has significantly improved the manuscript. I've listed the comments requiring responses below, with my response colored **in mahogony**.

1 Reviewer A

1) My biggest concern is simply that the people who ought to read it, won't do so (even if directed) because it is longer than what users typical want to read when they are in search of a fix to their problem. In this the article shares the same shortcoming with the well-known online essays [1] and [2].

I'd like a condensed summary of about one paragraph (maybe as a box) that I can just paste in an email or put on a website, together with a link to the full version.

I have added references to the Raymond and Tatham essays – I'm somewhat embarrassed I didn't think to add them myself, since they clearly influenced my thinking. I would argue that the checklists make a good short punchy summary, as the reviewer suggested. However, I agree having a plain-text version wouldn't hurt, so I've added a "Summary" paragraph before the checklists.

2) Does the author make a distinction between software that is advertised as useable (e.g. through project websites, presence of forums, issue tracker, ...) and software that has been made available (archive, repository, supplementary material)?

Can and should a user assume that a developer of the latter will provide any support?

Perhaps this could be made clearer.

My personal view is that FAIR principles of research make it essential that code is published together with the research (code dump with a license). However, for code dumps I wouldn't require authors to respond to more than the most fundamental inquiries (e.g., to respond when bugs are found that question the integrity of the published research). If more is demanded of authors then that disincentivizes publishing the code with the research.

On the other hand, if code is advertised (presumably with the expectation of citations or some other benefit to the developer) then I agree with the author that developers owe their users some of their time.

My focus was mostly on code that is intended to be used. I've added a paragraph at the end of the section for developers laying out my thoughts

on this distinction.

3) ”Please open a PR” (or merge request) is a phrase that can strike fear in the heart of a novice. It might be helpful to demystify this important part of collaborative software development. Maybe the a PLoS Comp Biol article [5] or something similar that touches upon basic software engineering that could provide more guidance.

I agree, and I’d argue that the relevance here is that it falls to the developers to provide a less intimidating way to contribute. I’d rather not talk about specific software engineering technologies here, though, just because I feel like it’s a topic I couldn’t do justice to without doubling the length of the paper.

4) It might be useful to prepare the new community member that conversations — although respectfule — are often to the point and that most developers won’t sugar-coat their words: if something is wrong, it will be said just like that because ultimately developers are concerned with maintaining the health of the code base and the project and have to be clear about what is acceptable.

New users should not feel offended by the phrase ”Feel free to work on it, submit a PR, and we can discuss there.” or similar. Unless you pay the developers to implement what you want, they will likely encourage you to do most of the work yourself and then spend some of their scarce time guiding you to a solution.

This is a good point (although I’d argue that developers ought to be a bit more courteous than that). I added a couple of sentences at the end of the bug report section, appended to the paragraph about feature requests.

5) Pointing out how much one can learn by interacting with experienced developers is important. This is a chance to learn from the best people in the field.

Furthermore, the networks that you build as part of a software project might last you a life time.

I added a sentence mentioning this to the section on contributing to the community.

6) I appreciate that the article primarily originates in the experience of the author. I would nevertheless find it useful if, where possible, other sources could also be cited as validation and to show that indeed these are broadly shared values.

I found the following references useful but I am emphatically not implying that they must be included in the article.

Ref [3] has a good section on the importance of communities.

I'm embarrassed to admit I hadn't read either of the suggested papers. I've now read them and added references to both.

2 Reviewer B

1) This point is already implicitly transparent throughout the paper, but it might be useful to state this more explicitly right in Section 3.1 as it (partially) motivates everything that follows: The easier you make it for the developer to narrow down/reproduce the issue the higher the chance they will help you. I think it's worth pointing out that developers are usually handling multiple issues at the same time and typically tend to assign lower priority to issues that are harder and time-consuming.

This is a very good point. I added a new paragraph under the description of a bug report.

2) In section 3.5 it is stated: "Once you've become reasonably competent with a piece of software, it's worth looking for ways to contribute back. [...] You can work on the documentation". This is a wonderful suggestion. However, I'm afraid that stated this way, users that don't feel competent enough will be hesitant/justified in contributing. I believe proposing small changes to the documentation often does not require competency. On the contrary, devs and experienced users typically take for granted a lot of things that might be missing from the docs. My point is that it might be a good idea to rephrase a bit to encourage even first users to propose changes to the docs as the best way to give back some of the time the devs/community has dedicated to them.

This is a very good point. The new version begins "Once you've begun using a piece of software, it's worth looking for ways to contribute back. This can take many forms, not all of which involve writing code (though of course contributing bug fixes or new features is extremely valuable). You don't have to be particularly expert – sometimes the best documentation comes from the inexperienced or newly experienced users, who vividly remember not understanding something in the manual and can suggest revisions that would make it clearer. "

3 Reviewer C

Thank you for your kind words!