Software Sustainability Plan

Today, software plays a crucial role in advancing and accelerating state-of-the-art academic research. It is therefore important to adhere to proven best practices when developing research software, as it will help avoid errors, improve maintenance and sustainability, while accelerating the overall development process. Also, it will help other researchers to better understand the intricacies of the software, and the analysis performed with it. This is an important prequisite for reproducibility of the scientific results, and will allow them to adopt the software into their own workflows, possibly even contributing to the software or expanding it.

To promote the practice of Open Science, and the principles of FAIR data and FAIR software, the Netherlands eScience Center actively participates in the National Platform Open Science.
In line with this initiative, the eScience Center strives to make the software developed in the projects it participates in to become freely and sustainably available, as much as possible, for reuse by other researchers.

The following questions are meant to make transparent the expected sustainability and impact of software as developed within Netherlands eScience Center projects. The questions are classified into three groups: “minimum effort”, “recommended practices”, and “long term aspects”.
Naturally, not all of the software is equally viable when it comes to potential reuse; research software typically consists of some generic components which could in principle be reused in other projects, which are then complemented by some other, non-reusable components with which the software is tailored to the problem at hand.
Generally speaking, at least the ‘minimum effort’ should be made for the complete body of code, i.e. both the reusable and non-reusable parts. For the potentially reusable parts, higher standards should be set.

Finally, note that our eScience Research Engineers can help with implementation during the course of the project.

Minimum effort

From the start of the project,

1. will the software be available under a permissive license such as
   Apache-2.0 (preferred),
   MIT,
   BSD or
   GPL?
   - [ ] Yes
   - [ ] No (warrants explanation)

2. will the software be developed in a publicly accessible repository such as GitHub (preferred),
   GitLab, or
   BitBucket?
   - [ ] Yes
   - [ ] No (warrants explanation)
3. will you adhere to the FORCE11 recommendations for citing software (https://doi.org/10.7717/peerj-cs.86; section *What software to cite*)?
   - [ ] Yes
   - [ ] No (warrants explanation)

As soon as possible after the project starts,

1. will a persistent identifier such as a DOI issued by Zenodo be added to the software?
   - [ ] Yes
   - [ ] No (warrants explanation)

Recommended practices

As soon as possible after the project starts,

1. will the software have its own entry in one or more software repositories such as the Research
   Software Directory (https://software.esciencecenter.nl), KBLab (http://lab.kb.nl/), Biotools (https://bio.tools/), or an other relevant directory?
   - [ ] Yes
   - [ ] No (warrants explanation)
2. will the software have documentation targeting new users that illustrates the
   software's intended usage?
   - [ ] Yes
   - [ ] No (warrants explanation)
3. For your software, will you fill in this checklist or an equivalent one?

   While more compliance is better, a 100% score is not usually expected. The checklist mainly serves as guidance on what aspects of your code could be better organized. Also, some sections of the checklist may be more relevant than others: use your best judgement in choosing which rules to comply with.
   - [ ] Yes
   - [ ] No (warrants explanation)

4. Will you prominently post the results from the abovementioned checklist, for example as a 'badge' in the README?
    - [ ] Yes
    - [ ] No (warrants explanation)

Long term aspects

During the course of the project, engineers from the Netherlands eScience Center will help lay the groundwork for making the software sustainable, for example through implementing software engineering best practices. However, the sustainability of software past the project's end date is ultimately the reponsibility of the project's Principal Investigator, who may delegate it to, for example, the research team, or the research community at large. Due to its business model, the Netherlands eScience Center cannot typically sustain all software past the project's end date.

This section tries to answer the generic question: "What will you do *during the project* to ensure that the software lives on past the project's end date?"

Below are a few suggestions to get you started. Feel free to interpret the above question in the broadest sense.

1. What efforts are being undertaken to create the critical mass necessary to start building a community of users, promotors, or otherwise interested parties around the software? For example,
    - through publications in mainstream media such as blogs, newpaper articles,
      YouTube videos, tweets, etc.
    - through organizing workshops, hands-on user trainings
2. Is there additional funding (in-kind or cash) that will be used to support the project's software outputs?
3. If the software is made available as a service, for how long will the service be offered? Which party will host the service? How will this be made possible
   financially?

e.g. some services may be transferred to external parties such as SURFsara, DANS, or an other suitable party, either during the project or shortly after the project ends.

1. Describe other aspects that promote the software's longevity.