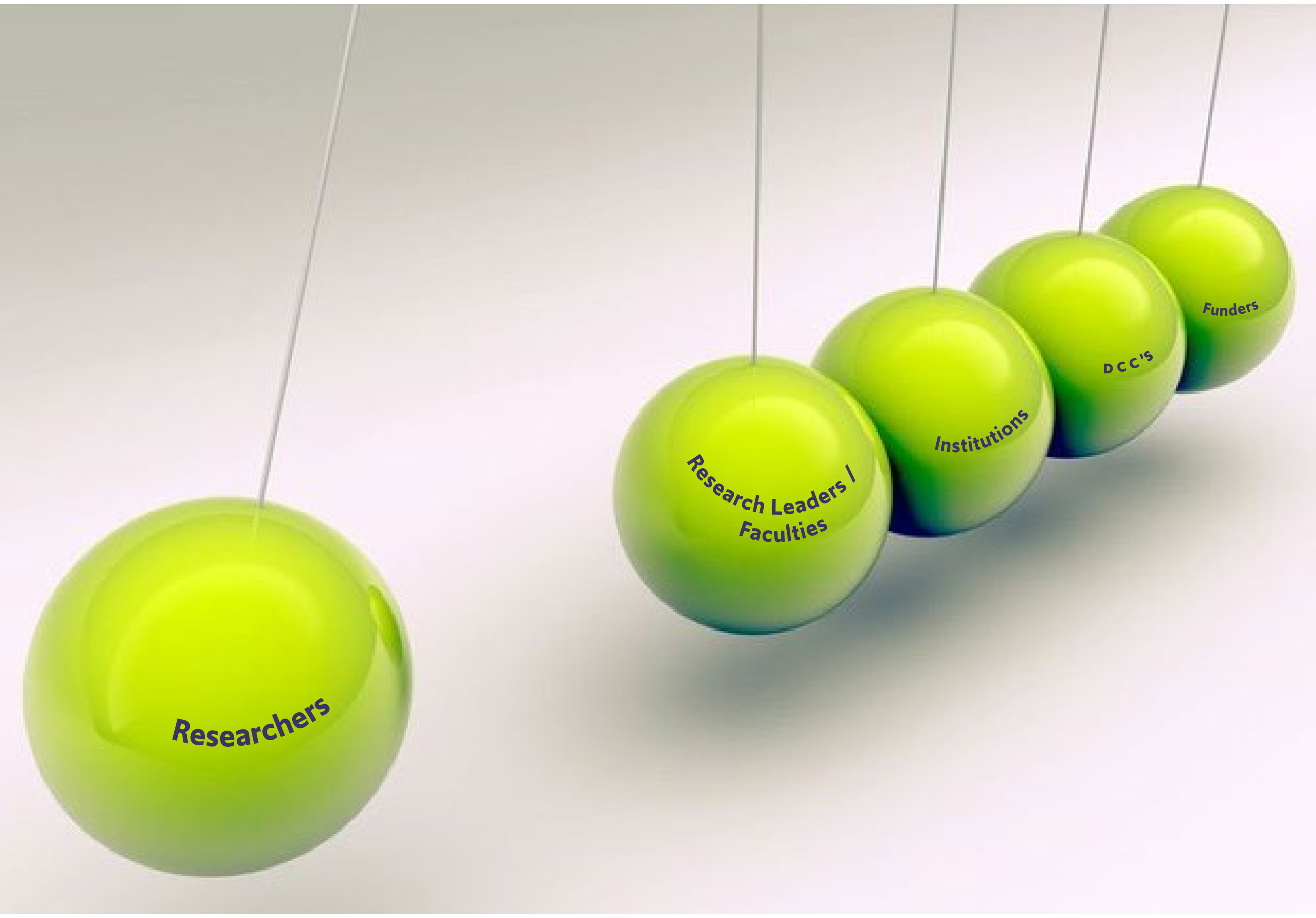
Research software sustainability in the Netherlands: Current practices and recommendations

National Coordination Point Research Data Management



LCRDM

The National Coordination Point Research Data Management (LCRDM) is a national network of experts on research data management (RDM) in the Netherlands. The LCRDM connects policy and daily practice. Within the LCRDM experts work together to put RDM topics on the agenda that ask for mutual national cooperation.

more information: www.lcrdm.nl

JANUARY 2021 | VERSION 1.0
LCRDM TASK GROUP RESEARCH SOFTWARE SUSTAINABILITY

Content

1] Introduction & summary

This LCRDM report presents a survey of current research software practices. We hope that the results will contribute to a profound discussion among all the stakeholders: what is needed in terms of policies and conditions to implement sustainable research software management in the Netherlands?

Research software has a pivotal role in enabling reproducible research and open science. The findings and recommendations in this report are based on interviews with research staff experienced in software development as well as an exploratory review of existing national and international initiatives. Key topics in this report are academic recognition, necessary skills and relevant support.

This report was created by the LCRDM task group Research Software Sustainability; please refer to Appendix 1 Background of this report for more information.

Research software in the context of open science

In its most basic form, open science calls for making the primary outputs of research publicly accessible with no or minimal restrictions. Research software is a key output of contemporary research and it is increasingly considered – alongside publications and research data – as one of the key pillars of open science (see Figure 1). While open access and FAIR data have received considerable attention in the context of open science policy in the Netherlands (see, for example, National Plan Open Science), this is not the case for research software and software sustainability. This report provides an exploratory assessment of topics and challenges in this area, resulting in a number of policy and other recommendations.

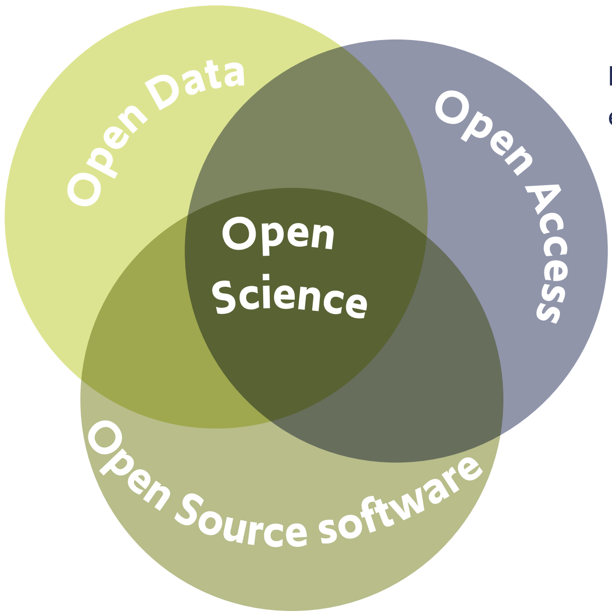


Figure 1. Open source software as key element of open science (Jomier, 2017)[1]

1] Open Science -- towards reproducible research. J. Jomier, Information Services & Use 37 (2017) 361–367, doi: 10.3233/ISU-170846

The complexity of software reuse

When considering research software as research output, it is important to understand that there are fundamental differences in relation to research data when it comes to preservation and reuse. As put recently by Daniel Katz[2], Chief Scientist at the US National Center for Supercomputing Applications:

"Well-written source code can be preserved and read, which allows the reader to see its intent and algorithms. However, using such source code can be more challenging, as doing so often depends on the hardware and software environment, as well as software dependencies. (...) In other words, we can preserve source code for reading, and we can preserve executable software for re-execution, but we cannot simply preserve software for reuse. For reuse, we need to sustain the software: to supply human effort that continually works on the software to adapt it as needed for changes in the environment and dependencies."

2] Open Science -- towards reproducible research. J. Jomier, Information Services & Use 37 (2017) 361–367, doi: 10.3233/ISU-170846

As Katz points out, the issues that arise in making software suitable for reuse are far more complex, substantial and long-term than is the case for FAIR data. In other words, software can only be made reusable when it is sustained, and developed with sustainability in mind.

Software is also fundamentally different from data when it comes to replication and reuse of research results and valorisation in industry and education. Because research software output is executable, it offers unique opportunities for researchers to build upon each other's work. As such, open research software allows research data and results to be challenged, to become dynamic and interactive and to be reused and built upon faster than ever before. We conclude that the lessons and best practices from the field of research data management are not easily transferred to research software.

Summary of the recommendations

This report offers recommendations (see Chapter 3) for research performing organisations, research funders and individual researchers. These recommendations should contribute to appropriate recognition of research software as academic output as well as to an increase in software reuse and – overall – improved software sustainability in the Netherlands. At a general level, the recommendations can be summarised as follows:

1. Create awareness
2. Develop policies
3. Stimulate considered use and reuse of software
4. Provide training & support
5. Adjust academic evaluation
6. Allocate funding

2] Findings from the interviews

Members of the task group interviewed a total of 37 researchers and research software engineers, with the goal of assessing their current practices for preserving and reusing research software. In order to ensure coherence in the interview results, we developed an interview guideline (see Appendix 2). Participants were selected based on their relevant experience in research software development and experience in sharing their software. The group of participants included researchers at all career stages, from full professors to PhD candidates, working in a broad range of research areas, including the humanities, life sciences, physics, engineering and computer sciences. See Appendix 3 for a summary of the interview results. The findings are categorised according to seven topics, presented below.

In interpreting the results, it should be noted that research software is multi-faceted and is very diverse in terms of size, complexity and format. This implies that not all views, opinions and practices pertain to the same idea or definition of 'research software'. In its most minimal form, software is a short script to validate certain data or to produce figures for a paper. This type of software is often intended to be used only by the researchers themselves or members of their research group. At the other end of the spectrum, entire software packages are written to be used by a broader group, including international research consortia, commercial parties and non-academic stakeholders. In those cases, software development practices are typically closer to professional standards.

Motivation for sharing

Most of the interviewees reported that they felt motivated to share their software, for various reasons. Most commonly, personal benefits were mentioned: sharing code contributes to establishing collaborations, increases the impact of the research, and brings recognition and exposure to the researcher. Sharing software was also considered to increase its quality, because other researchers may identify or even fix mistakes and because publishing code makes the publishing researcher more careful and diligent knowing the software will be publicly available. Due to code being freely accessible and including timestamps with its origin, sharing can also be a safeguard against plagiarism.

Academic principles were also given as a motivation to share software: making one's work open and reproducible is important for scientific and scholarly research to progress. Many of the interviewees had a strong conviction that, because research is mostly publicly funded, there is an obligation for it to be open and for researchers not to waste time by reinventing the wheel. Reciprocity was also mentioned: if you make use of open source software, you should contribute to the community. Some interviewees mentioned that it is problematic for academia to become dependent on commercial parties for – possibly expensive – software.

Besides intrinsic motivation, various external factors play a role, such as requirements from the institution or funding agency, or because sharing is the de facto standard in certain academic fields. Notably, some interviewees did not feel they benefited much from sharing their code: in fields where sharing is not the norm, the time spent on making it reusable was not deemed worth it. Others mentioned that sharing software can be scary "since everyone can see what you've done".

Archiving

When asked if they archive their software, the majority of interviewees reported that they do so or intend to do so. However, interviewees interpreted 'archiving' in different ways: for some it meant "have it all on disk, and it is backed up", or having it on home pages or servers. For most, it meant making and keeping it available on GitHub (or alternatives), for others, it meant depositing software in a trustworthy digital repository[3] such as Zenodo (sometimes in combination with GitHub), 4TU.ResearchData, Figshare, or institutional or field-specific repositories.

3] Trustworthy Digital Repositories have the exclusive mission of preservation for the long term; they guarantee the integrity of digital objects, assign persistent identifiers and have contingency plans in case they cease to exist.

In terms of making research software findable, the most common answers were: through GitHub and through publications (in some cases citing the DOI of the software). Dissemination through social media, mailing lists and other channels (e.g. blogs, curated lists) were also mentioned. Several interviewees reported that they only archive software to meet institutional or funder requirements, because they do not believe it will work in 10 years' time. Some interviewees said that their answers and behaviour would depend on the purpose of archiving the software (verification, reusability, sustainability).

Licensing

Just over half of the interviewees always use a license for their research software: mainly open source licenses[4] (e.g. BSD, GPL, Apache, MIT, Eclipse and EUPL). A few mentioned Creative Commons licenses. The remaining interviewees do not always, or never, license their work. Some said they have not given it much thought. Others just put their software online (without a license) expecting that it can be reused without restrictions, without realising that, as mentioned by another of the interviewees, "software must have a license, or it cannot be reused".

4] This can be problematic, as Creative Commons themselves recommend against using their licenses for software: https://creativecommons.org/faq/#can-i-apply-a-creative-commons-license-to-software

Of the researchers who do not always use a license, some indicated they have trouble selecting the right license, and would welcome help from their institutions. Some mentioned that working with commercial partners or with multiple institutions, each bound to different rules, were barriers when it came to the decision to apply a license to software or knowing which license to choose. However, it was also mentioned that although commercial parties had in the past been wary of open source licenses (especially those with a share-alike clause), this is now changing and commercial partners are starting to see the advantages of open source software.

Documentation

Almost all interviewees agreed that documenting the code they develop is very important, particularly if there is an intention to make the code available for reuse, so that others are able to understand and work with it. Almost all interviewees provide some form of documentation, even if just for their own future reuse of the code. As one interviewee put it, "You will thank yourself if you ever have to reuse your old software again." However, developing documentation is time-consuming according to the interviewees and these efforts need to be offset against other competing priorities, such as writing papers and carrying out data collection and analyses. If the expected target group for the software is relatively small, the documentation provided is less extensive and may only include a 'Readme' file with the necessary libraries. At the other end of the spectrum, for software projects spanning multiple decades, documentation may include a website with versions, information on how to build it in different operating systems, lists of add-ons and related packages, manuals, FAQs, tutorials, community forums, mailing lists and wiki pages.

Dependencies

Interviewees reported that reusing other people's code in a project can save a lot of time and effort, but its benefits depend on the quality and usability of the software. For example, when necessary dependencies are not available, software may not give the same results, or not work at all. The responsibility for this lies with the author of the software. Interviewees explained that, at a minimum, they list the necessary libraries in the documentation (Readme and environment files, wiki pages, etc.). Some use package managers, or integrate dependency management into their pipeline. Source code of dependencies might also be incorporated directly into their own project. Containerisation (e.g. by using Docker) was also mentioned, where the entire computing environment is stored elsewhere and can be run later.

There is also a risk that software projects break down. When software projects depend on many libraries and programs which themselves depend on other code, there is an increased chance that somewhere in the chain something will break. Code may be moved, updated (and no longer be compatible) or be removed altogether. Because of new versions or security issues, platforms may cease to operate or operating systems may become outdated. Looking at longer timescales, hardware will become outdated.

For these reasons some interviewees mentioned that they try to minimize their dependencies, and use only old, stable, well-known libraries. Some researchers for whom software development is more common reported that they actively maintain their projects. For them, factors that play a role in deciding to use someone else's code are: maintenance (is the software actively maintained?), communication (how to get in touch with the developers) and community (what are the options for getting involved in the developers' community?). Some respondents constantly check if dependencies still work using continuous integration and continuous delivery (CI/CD).

Recognition

Many interviewees acknowledged that they felt hindered by a lack of recognition and appreciation for spending time and effort on making software reusable. This includes time spent documenting, keeping software 'alive', and organising the community. These activities are currently not explicitly recognised in formal research assessment and researcher evaluations. Researchers therefore struggle to find time for tasks related to software sustainability, which limits the potential for software reuse, leading to an unceasing growth of unmaintained software. A few researchers felt that there is a direct trade-off between writing papers and writing reusable software. One interviewee expressed a wish to spend more time on reusable software and frustration at being evaluated exclusively on publications. Others mentioned was that academic publishing does not naturally sync with how software development works, and thus much time and effort goes into retrofitting projects into traditionally structured research papers. Improving the practice of proper software citation as well as help in demonstrating the use and reuse of one's research software would contribute to better recognition of developing software as part of standard academic practice.

What is needed from institutions

When asked what help was needed from their institutions, the most popular answers were:

– Specialised training and education for students and staff.
– Support from those with research software engineering expertise
– Help with choosing licenses

Some also mentioned the need for structural funding for much used platforms, such as GitHub and Zenodo; a wish for institutional presence at GitHub; and for institutional libraries to make the connection between the open data movement and open source software.

Conclusion

The researchers we interviewed were selected for their experience with applying open science practices to research software. Although they were highly motivated, they still struggled to allocate sufficient time to do so thoroughly enough and to their preferred standards. Many reported feeling a lack of recognition for such work because it is not a primary research product that counts towards formal evaluations. On the topics of licensing and archiving there seems to be much to be gained. Reuse of software was seen as a great time saver by some researchers. Other researchers expressed caution in reusing software, because problems may occur in chains of dependencies. Benefits put forward by the researchers who were interviewed included an increase in collaborations, impact, exposure and software quality.

3] Recommendations

There is increasing recognition of the crucial role software plays in contemporary research[5]. Until recently, however, most attention on the institutional and policy level has been devoted to developing improvements in creating open access policies and promoting FAIR and open data. Specific policies and frameworks to stimulate the development, use and recognition of sustainable research software seem to be in their infancy.

5] 'Recognising the Importance of Software in Research – Research Software Engineers (RSEs), a UK Example', Open Science Monitor Case Study, European Commission 2019, https://ec.europa.eu/info/sites/info/files/research_and_innovation/importance_of_software_in_research.pdf

This report provides a set of recommendations aimed at improving the practice of sustainable research software in the Netherlands. These recommendations are based on findings from interviews conducted with academic staff actively involved in research software development and expert input from the task group.

The interviewees helped us to gain an understanding of the conditions under which researchers in the Netherlands develop research software that is suitable for reuse. The issues brought up in these interviews are, however, not unique to the Netherlands. Similar topics have been discussed extensively over the past few years in various projects and tasks groups, both in national and international settings. These discussions have led to recommendations, policies and guidelines which, to a large extent, are also applicable to the Netherlands. Given the international nature of research, it makes sense to join and/or adopt these international efforts on software sustainability and archiving. The recommendations presented here, therefore, also take into account a review of other relevant initiatives.

We have summarized the recommendations in a set of tables. These recommendations are tailored to the different stakeholders. To effectively drive the desired change, collaborative actions are needed on several levels within the academic landscape as a whole.

The recommendations are set out in terms of six themes: awareness, policies, academic evaluation, reuse of software, training & support, academic evaluation and allocating funds.

Create awareness

A key step in stimulating sustainable research software is to raise awareness among researchers of the concept of software sustainability and associated best practices. This can be achieved by highlighting stories of successfully reused research software and demonstrating its advantages. Research institutions can benefit from and draw on relevant insights, developments and recommendations from national and international initiatives such as the FAIR Software[6] website (fair-software.eu), relevant RDA initiatives (e.g. RDA group for FAIR software), the Research Software Alliance (ReSA), articles[7] and the national community for research software engineers (NL-RSE)[8]. National organisations involved in digital research infrastructure (such as the eScience Center, DANS and SURF) should take on a more active and visible role in supporting the research institutions in their ambitions to develop relevant and high-quality support capacity within research groups and in digital competence centres.

6] The Netherlands eScience Center and DANS: 'Five recommendations for fair software': https://fair-software.nl/

7] Lamprecht, Anna-Lena et al. 'Towards FAIR Principles for Research Software'. 1 Jan. 2020 : 37 – 59. https://content.iospress.com/articles/data-science/ds190026 https://www.iospress.nl/ios_news/towards-fair-principles-for-research-software/

8] NL-RSE (https://nl-rse.org/) represents the community of Research Software Engineers (RSEs) from Dutch universities, knowledge institutes, companies and other relevant organisations.

Develop policies

Another important step research funders and research institutions can take towards promoting sustainable research software is to develop and implement software sustainability policies with a focus on open science. For example, the EOSC Scholarly Infrastructure for Research Software Task Force Report[9] suggests the following: “all research software should be made available under an Open Source license by default, and all deviations from this default practice should be properly motivated.”

It is important to note that not all software is equal, and policies should reflect this. Software is archived or made publicly available with different aims. Requirements and policies should therefore be made suitable for reproducibility purposes as well as for software sustainability.

9] The European Open Science Cloud (EOSC) Executive Board has set up a Task Force to provide recommendations on the policies and infrastructures needed to support research software as part of EOSC. The First Draft of this report is available via https://docs.google.com/document/d/1yObRCR-7COQctpjMdg-rNenRZ-7ZeUZ-uoiyvdpPRK598/edit#.

Stimulate considered use & reuse of software

Ideally, research software should be able to be reused, adopted and extended by the academic community. By doing so, it would prevent researchers from developing research software that may already exist. However, not all software is stable or suitable enough to be reused and maintained. Some software will only be preserved for reproducing research. Researchers should therefore carefully consider which of the research software they develop is suitable for reuse by others, and which part may simply be shared or archived.

Group leaders can stimulate good practices by encouraging researchers to develop software with reusability in mind. Public accessibility and good documentation, in particular, are crucial components for enabling software reuse. Funders can prime researchers for both reusing and developing reusable software by addressing this explicitly in funding applications and required data management plans, or they might consider asking for a separate software management plan.

While the creation of documentation may be perceived as time consuming, the use of automated documentation generators (such as sphinx or roxygen) can optimize the time spent on this. Documentation is not only crucial for reuse, but it also supports future development of the code by its own authors, as it explicitly provides important information about the code's functionality that may be forgotten as time passes.

Sharing code and software can help to increase researchers’ visibility. More than most activities, developing software lends itself to an open workflow through the use of social software development platforms such as GitHub. This has the potential to attract collaborators and improve the work at an early stage. Research groups and institutions can stimulate the use of software sharing platforms, for example by requiring them in all coding projects. Having an institutional presence on these platforms further helps the visibility of the institution, connects its researchers and encourages the use of the platform.
A further step institutions can take is to organise the formal registration of research software as scholarly output in research information systems. If research software is considered as primary research output, it should be assigned a persistent identifier (PID) and be registered in the same manner as publications.

Provide training & support

Although the researchers we interviewed were generally well versed in sustainable software best practices, many reported that they struggled with licensing choices, with some expressing the need for more support from their institutions. Software licenses are a notoriously tricky subject, but an appropriate open source license is a crucial element of re-usable research software. Research institutions are recommended to develop relevant and sufficient capacity to help resolve licensing conflicts, e.g. between dependencies or between contrasting wishes of different parties in a consortium.

Some interviewees also commented more generally about the need for support in academia. Some of them indicated that specific and more training is needed, starting with students at the BA and MA level. The standard curriculum should be expanded, not in the least for Master's degree programmes in less digitally oriented disciplines (such as languages, criminology or archaeology), as the use of software will only increase.

Researchers need to be trained on various best practices in developing sustainable software. Initiatives and resources such as the Carpentries[10], CodeRefinery, the Turing Way[11], and training given by the Netherlands eScience Center can be taken advantage of here, both by researchers themselves, and by training coordinators at research institutes. Guidelines for sustainable software development should be present in the curriculum and form the basis of programming work in a research group.

Finally, including research software engineers in research teams enables more efficient use of research software development in terms of cost and time (through continued maintenance, solid documentation and outreach via active web-based documentation, for example). Research software engineers can elevate the process by dealing with more technically challenging issues such as dependency management. They can either be hired by research groups or on an institutional level.

10] The Carpentries has been active since 1998 and focuses on foundational coding and data science skills. Their teaching materials are openly available and include basic topics such as programming in Python and version control with git.

11] The Turing Way is a community-driven guide to reproducible and collaborative data science, initiated and funded by the Alan Turing Institute. It contains extensive guidelines on topics relating to software sustainability as mentioned in the interviews above, such as licensing, version control, testing, continuous integration and reproducibility.

Adjust academic evaluation

Academics are still mostly evaluated on the basis of their publication record. Recently, however, there has been a clear shift nationally and internationally[12, 13, 14] to value research output more broadly (e.g. including datasets and software) and more qualitatively, including societal impact. In line with these recent developments, we recommend that research institutions develop appointment and promotion policies that recognise research software as a primary research product. Research institutions should facilitate career opportunities for researchers involved in research software engineering. In particular, we recommend recognising and rewarding Research Software Engineers (RSEs) to thus stimulate the differentiation of specialised and less-traditional academic career paths.

The Organisation for Economic Co-operation and Development (OECD) provides the following statement on RSEs: 'A growing number of people in academia combine expertise in programming with an intricate understanding of research. These Research

Software Engineers may start as researchers who spend time developing software or they may come from a more conventional software development background.

12] VSNU et al. 'Room for everyone's talent: towards a new balance in the recognition and rewards for academics' https://www.vsnu.nl/en_GB/Recognition-and-rewards-of-academics.html

13] San Francisco Declaration on Research Assessment: https://sfdora.org/read/

14] VSNU, KNAW, NWO 'Strategy Evaluation Protocol 2021-2027': https://www.nwo.nl/evaluaties-nwo-instituten

Researchers should be encouraged to cite the software they use and to make the software they develop citable. Software citation elevates software to the level of first-class research output.[15] Citing software helps to advance research and supports proper attribution and credit (similar to that of papers and data).

Software citations and other relevant metrics (e.g. number of public releases, or the size of the community of contributors) could be taken into account when evaluating research and researchers.

15] Katz DS, Chue Hong NP, Clark T et al. Recognizing the value of software: a software citation guide [version 2; peer review: 2 approved]. F1000Research 2021, 9:1257 (https://doi.org/10.12688/f1000research.26932.2)

Allocate funding

Sustainable research software comes at a cost. Time and effort are needed to develop research software such that it can be reused by others, and the software may need to be maintained and/or archived long after a project has finished. Currently, it is difficult to fund such efforts, especially after projects end. One way forward is to explicitly address sustainable software and sustainable software development in funding instruments. This can be addressed through the creation of funding lines that specifically target the development and maintenance of reusable software and long-term software archiving.

To make progress along these lines, more information is needed that sheds light on the financial resources and available infrastructures for long-term archiving and the hosting of sustainable software. On the national level, such information should be gathered by researchers and institutions to clarify the financial needs for future research proposals relating to research software sustainability and the benefits for academic research in general.

Tables with recommendations

To start the discussion on what is needed to encourage sustainable software in practice, we have drawn up recommendations for all stakeholders involved. The recommendations are grouped under themes and there are suggestions for which steps stakeholders could take in order to get results.

CREATE AWARENESS

INSTITUTIONS/ INSTITUTE LEADERS	FACULTIES / RESEARCH GROUP LEADERS	RESEARCHERS
Introduce programmes to highlight stories of successfully reused research software to demonstrate its advantages	Show your researchers the advantages of reusing software by using examples of good practice	Stay informed about relevant findings, developments and recommendations from e.g. FAIR Software, relevant RDA initiatives, reSA and NL-RSE

National organisations in the digital research infrastructure (eScience Center, DANS, SURF) should take on a more active and visible role in supporting research institutions in their ambitions to develop relevant and high-quality support capacity within research groups and in digital competence centres.

DEVELOP POLICIES

RESEARCH FUNDERS	INSTITUTIONS/ INSTITUTE LEADERS	FACULTIES / RESEARCH GROUP LEADERS	RESEARCHERS
Require software developed within funded projects to be as open as possible, as closed as necessary	Develop and implement research software policies and guidelines, including archiving and licensing	Familiarise your faculty/ research group with research software policies and guidelines, including archiving and licensing	Familiarise yourself with research software policies and guidelines, including archiving and licensing

STIMULATE CONSIDERED REUSE OF SOFTWARE

INSTITUTIONS/ INSTITUTE LEADERS	FACULTIES / RESEARCH GROUP LEADERS	RESEARCHERS
Facilitate paragraph on software management e.g. in the institutional DMP-tool Oraganise formal registration of research software (using pids) as a scholarly output in research information systems, similar to the registration of publications	Encourage researchers to consider which part of their research software will be suitable for reuse, and which part needs to be archived to ensure reproducibility of published results Encourage researchers to prefer the use of existing and stable software developed and supported by others over developing software from scratch	Increase visibility by enabling software citation. Deposit developed software on platforms like GitHub or Gitlab. Archive it on Zenodo or Figshare for long-term preservation and for obtaining a PID Increase your research efficiency by using software already developed by other researchers

PROVIDE TRAINING & SUPPORT

INSTITUTIONS/ INSTITUTE LEADERS	FACULTIES / RESEARCH GROUP LEADERS	RESEARCHERS
Facilitate effective and efficient support in the field of software development, sustainability and licensing Provide research software engineering support Include guidelines and training for software development in the curriculum	Train your researchers in good practices for the development of sustainable software Hire research software engineers as part of the research team	Participate in training courses on software development Make use of reseach software engineering expertise

ADJUST ACADEMIC EVALUATION

RESEARCH FUNDERS	INSTITUTIONS/ INSTITUTE LEADERS	FACULTIES /RESEARCH GROUP LEADERS
Acknowledge research software as a principal form of academic output	Reward software development capabilities in your evaluation procedures, according to the new Strategy Evaluation Protocol (SEP) guidelines	Include software citations and other relevant metrics when evaluating researchers

ALLOCATE FUNDING

RESEARCH FUNDERS	INSTITUTIONS/ INSTITUTE LEADERS	FACULTIES / RESEARCH GROUP LEADERS	RESEARCHERS
Explicitly address sustainable software and sustainable software development in funding instruments (e.g. in conditions and evaluation criteria) Require a paragraph on financial needs for software sustainability Develop funding instruments for covering the cost of software archiving after the project has finished	Gather information about the financial needs and required infrastructures for long-term archiving and the hosting of sustainable software	Include the cost of archiving software after the research has been completed in your funding proposals	Include the cost of archiving software after your research has been completed in your funding proposals

Appendices

Appendix 1. Background of this report

This report started with a pitch from SURF to the Advisory Board of LCRDM to investigate software archiving. The pitch was approved in March 2020 and a task group put together in May 2020. The task group comprises research support staff from several universities and research institutions in the Netherlands and staff from SURF, NWO and the Netherlands eScience Center. Based on the discussions among the group, the scope of the report was broadened to software sustainability and reuse, rather than focusing just on software archiving. In the context of software, archiving refers to storing source code without alteration, while sustainability covers various aspects, such as reproducibility of results (running the software to verify previous results) and reusability (using the software in a different context to produce new results).

Pitch proposal:
Which conditions encourage a researcher to archive research software?

1. Purpose and demarcation of the subject

Within the context of open science, researchers are asked by funders to provide them with the data and keep the software that led to the results of the research for at least 10 years. The storage of research data is taking place more and more; the storage of research software is still (very) scarce. Storing research software in a way that makes reusability possible requires considerable effort from the researcher who made the software. Within the current research process, there appears to be few incentives for these activities / costs. However, archiving the research software used is more common within some research areas. We want to investigate why this does take place within these research areas.

2. Why is this of cross-institutional interest?

Many institutions struggle with the issue of how they can stimulate the archiving of research software. Archiving it is not widely used at the moment. Many institutions are thinking about this issue, but the experience is not yet great. Precisely because this is a relatively new issue, a joint approach to obtaining more information about the possibilities for stimulating software archiving is useful.

3. Deliverables

We want to provide practical recommendations for institutions to stimulate the archiving of research software among researchers. We want to base these recommendations on researching best practices in the field of research software archiving within research areas where this is already applied, at home and abroad. In doing so, we will focus on the following questions for the researchers:

- What do they see as the purpose (s) of archiving their research software?
- Is extra effort required to proceed with archiving? If so, how much extra?

- Do they feel appreciation for the software they have archived? If so, how? –
- In their view, how could the archiving of research software be stimulated even more?

4. Necessary expertise and competences

The following expertise and competences are required in this task group:

- research support employees who are or will be involved in issues related to archiving research software
- employees eScience Center and DANS because of their extensive knowledge in the field of software archiving

For this research contact will be sought with researchers / research software engineers who are currently archiving their research software. It is therefore important that the employees involved in this study have access to this group of researchers.

5. Estimated duration and schedule

The start date is May 26, with a lead time of 8–10 weeks. We will work with a 2-weekly meeting of 2 hours, with discussions and elaborations by email in the meantime.

6. What has already been known or done and why does this information not suffice?

Research into the possible incentives for researchers to archive the developed software has not been carried out a lot, because this is a relatively new area.

Appendix 2. Interview guideline

Research funders want to stimulate researchers to archive the software they created for their research. This desire has the following background: the research carried out must be reproducible; both the data and the software should be described and archived, so other researchers are able to reuse the software to check the results. Research software that can be used by other researchers can speed up research. Also, for this purpose, the software needs to be stored (archived) somewhere accessible by others, and needs to be described. At the moment, a lot of research software created, is not ready for the purpose of reuse by other researchers. However, some researchers (research groups) already store/archive their software for reuse. This is the group we target specifically with this questionnaire. We want to find out what actions are taken to make the developed software reusable, what researchers gain from writing reusable software and where they store (or archive) the software. How do they make the software findable for others?

By research software we mean all the software code (including scripts, apps, models, tools, algorithms etc.) that is being used to produce research results, but also software that is the research result in itself.

Q 1. Job title
Q 2. Research area
Q 3. You use the software for:
Q 4 What type of software do you generally use in your research?
- Software developed by myself/my support group
- Software developed by other researchers I know in my research area
- Any software I can find on GitHub that fits my needs
- Combinations of types mentioned above
- Other

Q 5 The software you develop will be used by:
- Yourself
- Your research group
- Other researchers in the field you work in
- Anybody who wants to

Q 6. When you develop research software:
- You decide in advance whether to make it reusable for other researchers in advance
- You usually take the decision to make it reusable afterwards
- It depends, sometimes you decide in advance, sometimes afterwards
- Many times, you want to make it reusable, but you fail to succeed

Q 7. Do you usually associate your research software with a license? What license type(s) do you use?
Q 8. Is the software you develop documented? Do you provide different types of documentation? If so, please explain
Q 9. Do you use version control when developing software? And do you release new versions of your software after your research is completed?
Q 10. Do you write tutorials for your software? If so, for whom do you write these?

Q 11. Do you document the format of the input and output data(sets) for which your software was developed? If so, how do you do that?

Q 12. How do you manage the dependencies (e.g. packages, libraries) of your software?

Q 13. What are the considerations for you to develop reusable research software? Why do you do it, or not?

Q 14. Do you use software developed by other researchers yourself? If yes, can you describe for what purpose?

Q 15. When you develop reusable software, what is your practice and approach? Is this different from developing software you will not share? Please elaborate

Q 16. Does it take extra effort to make the software reusable? If yes, please elaborate.

Q 17. Do you archive your research software after you developed your code? If yes, what tools do you use?

Q 18. Do you experience difficulties in archiving your software? If yes, can you elaborate?

Q 19. When you archive reusable software, how do you make it findable for other researchers?

Q 20. Do you feel you gain personally from writing reusable research software? If yes, please elaborate

Q 21. Do you experience limiting factors when you develop reusable software? If yes, please describe these

Q 22. What kind of support would you like to receive from your institution in the area of research software development and archiving? Please elaborate

Q 23. Is there anything else you want to bring up on the subject of software reusability and software archiving?

Appendix 3. A summary of the answers per question

Q 1. JOB TITLE

Professor | 3
Group leader | 2
Assistant professor | 5
PhD | 7
Post Doc | 2
Data scientist | 1
Software developer | 3
Scientific staff | 1
Researcher | 1
Advisor | 1
Unknown | 6

Q 2. RESEARCH AREA

Interviewees indicated to work in the following research areas:
Science: 9
Life Science: 14
Software development: 7
Humanities: 3
Other: 1
Combinations: 3

Q 3. YOU USE THE SOFTWARE FOR:

- Create models, generate and analyse data
- Develop software, and support the researchers in their group with development
- Research on software itself
- data processing, analysis and modelling
- pipelining/workflow

Q 4. WHAT TYPE OF SOFTWARE DO YOU GENERALLY USE IN YOUR RESEARCH?

- Software developed by myself/my support group | 4
- Software developed by other researchers I know in my research area | 0
- Any software I can find on GitHub that fits my needs | 0
- Combinations of types mentioned above | 3

- Other | 3
    - An important one. Programming experiments, data maintenance, and reproducible handlings and visualization of complex data is crucial. Without software there would be very little research.
    - main focus
    - *no answer*

Q 5. THE SOFTWARE YOU DEVELOP WILL BE USED BY:

- Anybody who wants to | 20
- Other researchers in the field you work in | 19
- Your research group | 25
- Yourself | 18

Q 6. WHEN YOU DEVELOP RESEARCH SOFTWARE

- You decide in advance whether to make it reusable for other researchers in advance | 1
- You usually take the decision to make it reusable afterwards | 5
- It depends, sometimes you decide in advance, sometimes afterwards | 16
- Many times you want to make it reusable, but you fail to succeed | 1

I always/often aim to make it reusable for other researchers | 1
I sometimes aim to make it reusable | 2
COMBINATIONS:
- You decide in advance whether to make it reusable for other researchers in advance + It depends, sometimes you decide in advance, sometimes afterwards | 2
- You decide in advance whether to make it reusable for other researchers in advance + Many times you want to make it reusable, but you fail to succeed | 1

no answer: 1

Q 7. DO YOU USUALLY ASSOCIATE YOUR RESEARCH SOFTWARE WITH A LICENSE? WHAT LICENSE TYPE(S) DO YOU USE?

19 out of 32 researchers indicate to license their software. BSD is mentioned the most (6), then GNU (L)GPL (4)x, Apache (3x), CC-BY (4) (not relevant for software), MIT (2), Eclipse (1), EUPL (1).
5 out of 32 researchers indicate to license their software occasionally.
1 researcher wants to do it but is still searching for the right license.
7 researchers indicate not to license their software.

Several interesting remarks were made:
– Some researchers indicate that commercial parties consider the use of the GNU GPL license as unwanted. A chance in that behaviour however is being detected; commercial parties start to see the benefits of open source
– One researcher struggles with the ownership of the software; technically the university is the owner.
– 2 researchers indicate not to license their software because their software can be made available on request
– Two researchers mention not liking commercial parties making money off of their software.

Q 8. IS THE SOFTWARE YOU DEVELOP DOCUMENTED? DO YOU PROVIDE DIFFERENT TYPES OF DOCUMENTATION? IF SO, PLEASE EXPLAIN

Most researchers indicate to provide limited documentation. This type of documentation consists of
– inline information in the code (mentioned 8 times)
– information in a wiki (mentioned 4 times)
– information in a readme file (9 times)
3 Researchers provide documentation occasionally.
Extended documentation is provided by 6 researchers.

4 researchers mention the use of automatic generated documentation/ integrated documentation. All major languages have this facility, for instance R has Roxygen.

The constraints in time and budget are mentioned as a reason for limited documentation. Also, the limited need of documentation is mentioned, as the users of the software are familiar with this type of software.

Q 9. DO YOU USE VERSION CONTROL WHEN DEVELOPING SOFTWARE? AND DO YOU RELEASE NEW VERSIONS OF YOUR SOFTWARE AFTER YOUR RESEARCH IS COMPLETED?

25 out of 32 researchers apply version control on their software. This is research software that is used for longer periods of time after the research is finished. Only 2 researchers indicate not to apply version control after the research has finished.
3 researchers don't apply version control, and 1 sometimes.

GitHub is widely used by the researchers for their software.

Q 10. DO YOU WRITE TUTORIALS FOR YOUR SOFTWARE? IF SO, FOR WHOM DO YOU WRITE THESE?

Tutorials are written by only 11 researchers. Lack of time is mentioned regularly. Also 11 researchers indicate never to write a tutorial, 5 researchers occasionally. 1 researcher would like to but didn't.

Some researchers indicate that the users that could make use of the system do not need a tutorial.

Four researchers have made video instructions. Two mention teaching their software to others.

Q 11. DO YOU DOCUMENT THE FORMAT OF THE INPUT AND OUTPUT DATA(SETS) FOR WHICH YOUR SOFTWARE WAS DEVELOPED? IF SO, HOW DO YOU DO THAT?

23 out of 31 researchers describe the input and output data. The description types used are:

– Examples	1
– Elaborate description	3
– Readme files	4
– Inline in the code	2
– Use standard data	3
– Dummy file	1

(not all researchers indicated the type of description)
5 out of 31 researchers do not add a description, 2
4 researchers did not answer the question or were not asked.

Q 12. HOW DO YOU MANAGE THE SOFTWARE DEPENDENCIES (E.G. PACKAGES, LIBRARIES) OF YOUR SOFTWARE?

29 answers
This question triggered a lot of researchers with an extensive answer. It is obvious that this is a difficult area, with lots of challenges. Researchers have different ways of coping with this problem:

- The use of standard libraries
- The use of containers (Dockers)
- Active maintenance
- Description of the used libraries and packages
- The use of built-in methods for dependency management in the software tools (for example in Python)
- Minimalize the dependencies
- The use of old computer equipment

One researcher indicates to expect the problem to grow in the coming years, because of the use of cloud technology. He/she sees problems occur as soon as 3 months after the last update.
Also, interesting to mention is the practice of CI/CD (continuous integration, continuous deployment), which spots dependency issues as soon as possible. Specifically mentioned is the tool TravisCI.

Q 13. WHAT ARE THE CONSIDERATIONS FOR YOU TO DEVELOP RE-USABLE RESEARCH SOFTWARE? WHY DO YOU DO IT, OR NOT?

A wide variety of considerations were mentioned. Many researchers indicate not one reason, but 2. These were the considerations mentioned:

– Ideological motivations, like passing on knowledge and contributing to society
– Use of open source software written by others
– Re-use of the software by the own research group
– To help other researchers
– Cooperation with other researchers; goodwill
– Tracking mistakes in the software
– Open research (verification)
– Internal policies
– Funding requirements
– Enabling testing of automated analyses

Q 14. DO YOU USE SOFTWARE DEVELOPED BY OTHER RESEARCHERS YOURSELF? IF YES, CAN YOU DESCRIBE FOR WHAT PURPOSE?

27 researchers out of 29 use software developed by others themselves
Only 1 researcher does not use software developed by other him/herself
1 researcher does it occasionally

Re-use is:

– To save time
– Not reinventing the wheel
– Standard software that is being used in the area of research
– Cooperation with other researchers
– building on top of other research already been done
– use expertise from others, that is less available

Q 15. WHEN YOU DEVELOP REUSABLE SOFTWARE, WHAT IS YOUR PRACTICE AND APPROACH? IS THIS DIFFERENT FROM DEVELOPING SOFTWARE YOU WILL NOT SHARE? PLEASE ELABORATE

15 researchers say it takes a certain amount of effort to make the software available to other researchers; they refer to better documentation, different development standards, more comments.

11 researchers say it does not take too much effort. Some say, they always develop with re-usage in the back of their minds, which explains why they don't need to make any extra effort. Or they create a bit of extra documentation. For some, the research area requires open software.

For 4 researchers, making the software reusable takes a huge effort.

Q 16. DOES IT TAKE EXTRA EFFORT TO MAKE THE SOFTWARE REUSABLE? IF YES, PLEASE ELABORATE.

20 researchers feel that making research software reusable takes a lot of time. Extra documentation, testing, extra description of the functions, structure of the code, dependency checks.

5 researchers say it takes a little bit of extra time.

4 researchers say it does not take much time or none, to create reusable software.
2 researchers claim it takes a bit of time in advance, but that evens out in the long run, because of the time benefits later.

2 researchers did not answer the question.

Q 17. DO YOU ARCHIVE YOUR RESEARCH SOFTWARE AFTER YOU DEVELOPED YOUR CODE? IF YES, WHAT TOOLS DO YOU USE?

23 out of 29 researchers say they archive the code. Most of them use GitHub/Gitlab, a lot of them in combination with Zenodo. Some other archiving tools are mentioned only once: 4TU, Figshare, institute repository, CRAN, Bitbucket.

4 researchers say not to archive their software.
1 researcher saves all software to his own, personal archive.

For 1 researcher this question was n/a.

GitHub is widely used. The only problem with GitHub is the fact that all software on GitHub must be open. If research is conducted for commercial industries, GitHub cannot be used.

TUE and CWI have installed Gitlab for that purpose. Other institutes have their own repositories in place. For cooperation with students, Bitbucket was suggested. The connection GitHub – Zenodo is highly recommended. Figshare is mentioned only once and was indicated as not very user friendly.

Remark: GitHub cannot be used for archiving, as GitHub is a commercial platform and gives no guarantees for long term storage in case they cease to operate. GitHub is very useful for making the software findable and accessible.

Q 18. DO YOU EXPERIENCE DIFFICULTIES IN ARCHIVING YOUR SOFTWARE? IF YES, CAN YOU ELABORATE?

17 researchers do not experience too many difficulties. Some feel Zenodo is a good tool for this and getting to know the tool takes a little time.

4 researchers say this does not apply to them (yet).
1 researcher makes a specific comment, that archiving tools that consist of multiple modules in separate repositories is hard

Another researcher indicates that Gitlab is not yet recognized by all students or researchers. They download the software and create their own versions on their own laptops. Integrating these versions afterwards with Gitlab is impossible.

Q 19. WHEN YOU ARCHIVE REUSABLE SOFTWARE, HOW DO YOU MAKE IT FINDABLE FOR OTHER RESEARCHERS?

28 researchers indicate to take actions to make their software findable. The means they use are:
- GitHub. GitHub is indexed by Google (mentioned 13 times)
- DOI on Zenodo
- refer to the code in the publication
- a self-made wiki page or a website for a project
- twitter and other social media
- CRAN
- Software mentioned on specific fora on the internet
- Project-based mailing lists
- Personal interaction with other researchers

Only 1 researcher indicates not to take any action. 1 researcher did not answer this question, another one mentioned it to be not applicable.

Q 20. DO YOU FEEL YOU GAIN PERSONALLY FROM WRITING REUSABLE RESEARCH SOFTWARE? IF YES, PLEASE ELABORATE

4 researchers indicate not to gain from writing reusable research software. One of them even states that it is a disadvantage, as he is not judged on open software, only on papers and education. And open software takes time.

17 researchers feel it benefits them. Reusable software benefits them in:
– Reputation/recognition/exposure
– Collaboration
– Goodwill
– Reproducibility of research results
– Time saving
– Testing
– Increase of citations

7 people indicated to have limited expectations on their personal gains. It takes them time and effort, but they expect not much in return, for instance because their field does not value sharing as much as other fields
2 researchers mentioned that writing reusable software was ingrained in the work they do. They have no other option.
For 1 respondent the question was not applicable.

Q 21. DO YOU EXPERIENCE LIMITING FACTORS WHEN YOU DEVELOP REUSABLE SOFTWARE? IF YES, PLEASE DESCRIBE THESE

Generally, time and effort required for writing reusable software are mentioned. In research, time is always a limiting factor. Anything that adds to that, is a problem.

Also mentioned as limiting factors are:
– A conservative approach of the institute or co-researchers, or the need to convince these other researchers of the usefulness
– Licenses
– Users not used to work with software
– Requirements of the institutes for working with specific programming languages
– Different compute environments
Not enough academic recognition

2 researchers do not experience any limiting factors. For 1 researcher the question was not applicable.

Q 22. WHAT KIND OF SUPPORT WOULD YOU LIKE TO RECEIVE FROM YOUR INSTITUTION IN THE AREA OF RESEARCH SOFTWARE DEVELOPMENT AND ARCHIVING? PLEASE ELABORATE

A lot of different things are mentioned here, but 2 things stand out:
– Training/education for students and staff in programming of reusable software
– Help in licensing

OTHER ASPECTS MENTIONED WERE:

On infrastructural components:
– Availability and information on repositories
– More infrastructural components that suit research, for development and archiving, including the long term finance of these environments
– Test environments
– Integrated GUI based software for support on version control and archiving

On policies and documents:
– Policies on software development
– Support on creating paragraphs on archiving and reusable software design in project plans
– Guidelines for description of dependencies
– Easily accessible information on aspects of reusable software, guidelines, expertise and tools

Other supporting aspects:
– Information on who wrote what software in the university
– Collection and analysis of bibliometric data, based on NWO specifications
– Long term support for good software
– Help in documentation of software
– More scientific programmers
– Long term finance

And last, but not least, upgrade the appreciation for software development in general.

Q 23. IS THERE ANYTHING ELSE YOU WANT TO BRING UP ON THE SUBJECT OF SOFTWARE REUSABILITY AND SOFTWARE ARCHIVING?

Different topics came up. Things to look for, things to rethink and some questions or wise remarks.

Things to look for:
- Education on data management/software management
- Support needed for archiving software
- Educate in documentation of software
- We need mechanisms for quality control
- Besides clinical data, take other data into consideration as well
- Sharing software code is scary; a mentality change is needed
- We need a new culture of sharing
- Computer scientist should be evaluated on software, rather than on papers
- Citations of software are not used wide enough
- Teach people how to write software
- Create more awareness
- Publish research software
- We could learn practices from software development companies, like elements of 'extreme programming'

Things to rethink:
- There is now a lot of pseudo-openness: where researchers throw everything on GitHub, without it being really reusable
- The increasing complexity of dependencies (because there is a lot of re-usable software) will become a problem in the future.
- We are approaching a bottleneck where development of new hardware becomes difficult due to the complex software ecosystem, but on the other hand we cannot continue with new software because no new hardware will be available.
- Reuse of software saves * no * time and is not a smart thing to do. It creates more dependencies than necessary which can be a big problem
- If people use software from another person, there is the risk of not completely understanding that software, which can create mistakes
- We shouldn't get tied to single for-profit companies; we don't want our data held hostage
- For recognition purposes we could track in which other projects your code is used, through 'reverse depends and reverse imports'. R has this function; GitHub is starting to do that too.

Questions and wise remarks:
- The software is only as good as the repositories that still exist
- Software can be perfectly re-usable but never archived and vice versa
- The words: "Legacy software" are often used in a negative sense, but it is positive. Rule of thumb: the older the program, the longer it will be useful.
- Not all reusable software is forever; start small
- Can or will the library support software archiving?
- Software is underrated in academics
- It's only science if you write it down. Otherwise, you're just messing around
- When reviewing, always ask for the archived code. That helps establish a new culture
- It is problematic, even immoral, to be dependent on (expensive) software as a research institution

Colophon

zenodo: 10.5281/zenodo.4543569

AUTHORS AND CONTRIBUTORS

Tom Bakker | 0000-0002-6465-5589
Maria Cruz | 0000-0001-9111-182X
Saskia van Eeuwijk | 0000-0002-1322-7429 | *initiator task group*
Vera Sarkol | 0000-0002-8950-3178
Barbara Vreede | 0000-0002-5023-4601

Bart Aben | 0000-0002-7918-1590
Patrick Aerts
Gerard Coen | 0000-0001-9915-9721
Bob van Dijk | 0000-0002-2129-7013
Peter Hinrich | *member consultative group*
Lena Karvovskaya | 0000-0001-7777-5603
Meta Keijzer
Jacko Koster | 0000-0003-1079-4927
Jason Maassen | 0000-0002-8172-4865
Miriam Roelofs | *chair task group*
Jan Rijnders | 0000-0002-9397-1302
Alfons Schroten
Laurents Sesink | 0000-0001-7880-5413 | *liasion Advisory Group LCRDM*
Chris van der Togt | 0000-0002-7373-4625
Jurgen Vinju | 0000-0002-2686-7409
Petra de Willigen

DESIGN | Nina Noordzij, Collage, Grou