Research Software at the Heart of Discovery

Principles of Academic Software-Sustainability

16/02/2016

Editors:

Dr. Peter Doorn – DANS

Dr. Patrick Aerts – DANS & NLeSC

Dr. Scott Lusher – NLeSC

Data Archiving and Networked Services
DANS

netherlands eScience center

Research Software at the Heart of Discovery

Principles of Academic Software-Sustainability

Summary & Recommendations

This document sets out the need for increased attention to academic software-sustainability and research software practice in general. The key recommendations are:

- Utilise existing organisations and models to develop a coordinated effort in The Netherlands to promote good research software practice and software-sustainability.
- Develop software- and data-carpentry training for new PhDs.
- Promote research software as a citeable scientific deliverable of equivalent value to publication.
- Ensure recognition and career paths for software research engineers.
- Develop academic models for sharing and disseminating research software.

Introduction:

Software is a fundamental component of modern research, without which, twenty-first century science would be impossible. In this era of data-driven and compute-intensive research, software is the toolkit that enables us all. The continued development of science is in part dependent on improvements in research software. Despite the reliance of software in academia, professional practices for its development too often lag behind those in the commercial sector. The pressure on researchers to rapidly publish new results, without the need to develop professional quality code, results in academic research software being fragile, generally not sustainable or usable beyond the lifetime of a given project. As a result, the research community is failing to benefit fully from the potential impact of their research.

The goal of software-sustainability is to ensure that impactful current software functionality continues to be available - improved and supported - in the future. However, there are many issues that accompany this topic. These include the choice of which software is worthwhile keeping and what are criteria to select the tools that require maintenance and/or improving. Who will fund sustainability, for how long and according to which protocols? What will be the means for access, publication, updating, and version control as well as how to train new research software developers and support the careers of existing software research engineers?

The purpose of this document is to raise awareness of these issues and provoke debate about software-sustainability and the importance of research software in general.

Goals:

The intention of this document is to promote:

1. The importance of good software development practice within academic research
2. To promote models of software sustainability to ensure:

    a. Improved access to academic software over time
    b. Reproducibility of scientific results
    c. Prevention of duplicate software development
3. To support the dissemination of the best software across application areas and disciplines
4. To support the careers of "software research engineers" within academia
5. The recognition of research software as a valid scientific deliverable requiring dedicated routes of publication
6. Training opportunities to improve the quality of academic research software

The importance of research software:

The "Science Code Manifesto" (http://sciencecodemanifesto.org/) sets out the importance of research software by stating "***Software is an essential research product, and the effort to produce, maintain, adapt, and curate code must be recognized. Software stands among other vital scientific contributions besides published papers."*** To achieve these goals they propose 5 key principles:

- **Code**. All source code written specifically to process data for a published paper must be available to the reviewers and readers of the paper.
- **Copyright**. The copyright ownership and license of any released source code must be clearly stated.
- **Citation**. Researchers who use or adapt science source code in their research must credit the code's creators in resulting publications.
- **Credit**. Software contributions must be included in systems of scientific assessment, credit, and recognition.
- **Curation**. Source code must remain available, linked to related materials, for the useful lifetime of the publication.

Software-sustainability:

Research data is especially and increasingly dependent on the software environment in which it is created, managed, analyzed and presented. Without the appropriate software, other research outputs such as scientific publications and datasets cannot be viewed or used. Yet, software usually has a short lifespan, and quickly tends to become useless if it is not properly maintained and sustained. As outlined by the UK's Software Sustainability Institute, all research software should be considered as a form of scientific output *"From the grand problems that push the boundaries of human knowledge, to day-to-day research tasks, software has made an invaluable contribution to advancing research. We believe that the full benefits of software in research will only be realized when software is accepted as a valid research output."* (http://www.software.ac.uk).

In sustaining software one should distinguish aspects of archiving and/or accessibility on the one hand from preservation, maintenance and/or regularly updating on the other, which requires more expertise and an infrastructure to deal with it. Software-sustainability is significantly different from basic archiving or preservation. Preserving software in principle entails that the code is stored in a trustworthy place that it is accessibly archived, documented (described by metadata) and can be found by search engines. Archiving does not guarantee that the software will still run at a later point in time.

Archiving software is like deep-freezing or canning food so that it can be used some time in the future. In this case, the labels on the can or package function as metadata.

On the other hand, sustainability requires the software to remain functional, even over significant periods of time. Software needs to be updated, adapted to new ICT-environments, made user friendly and multi-platform, tested, and certified. Making sure that software continues to be executable requires much more: technical, application and functional maintenance, updates and perhaps even testing and certification. This is more labour-intensive and hence more expensive than archiving, and requires expertise and infrastructure. Any changes to software needed to remain executable as operating systems and hardware evolve must ensure the same reliability of analytical results as older software iterations. If results differ, it must be clear and accountable why they differ.

There are potentially several ways to ensure the sustainability of software, including existing user community support, commercialization or archiving. Providing they result in the long-term availability of the software to the scientific community, they are all valid. It is expected that through the publication of directions, guidelines and other forms of help and support, individual researchers will be able to select the direction that best ensures their software remains sustainable.

There is also no general answer to the question for how long software needs to be sustained or maintained. This period may depend on the discipline, purpose, function, use, etc. and is most likely to be dependent on the community that developed it and the development of new software (replacement).

The importance of software-sustainability in reproducing research:

It is important to recognise the importance of maintaining access to software that has been used in the production of scientific communications. Good scientific practice should lead us to conclude that software referred to in scientific publications needs to be kept and maintained for the purpose of reproducibility and verification of scientific results. In practice, there is very limited requirement for researchers to share the code, algorithms or workflows they utilise in their research. The current movement to require researchers to share data, and particularly to provide access to data described in publications, will not fully support reproducibility without provision for supporting software. In most cases, cheap archiving of software with clear documentation (including version control etc.) should be sufficient. For this aspect of software sustainability it is urgently needed that guidelines are given to all researchers, in particular those that are still in a phase before the actual conception of new software, on coding ethics, good practices and other guidelines to make later use easier, once the software or tools have been created. Guidelines for reliable software development will contribute to proper scientific conduct. Such guidelines may differ in their implementation between scientific domains, but these implementations should be public, open for discussion and continuously adapted to changing situations and technologies.

Potential benefits of a national software-sustainability policy:

- Improved reproducibility of scientific results obtained using research software
- Increased return on investments from scientific projects
- Increase in the pace of scientific discovery by (re-)use of already available code as a start

- Ensuring that data will remain readable, interpretable, usable by the software from which it was generated
- Helping researchers with their research by enabling them to stand on the shoulders of their predecessors
- Keep historic software to read historic documents

Deciding which software should be sustained:

It’s unrealistic and potentially harmful to presume that all academic research software should be sustained. Good research software must be efficient, calibrated, reliable and accessible and be based on excellent standards of code quality utilizing meta-data standards and software development environments. Software that shows no additional value compared to existing solutions, or that has failed to develop a user community, or that is technically flawed, may not need to be retained. This software may still need to be archived and carefully documented if referred to in scientific publications, but only software meeting a new user-driven demand, demonstrating improved efficacy, usability or reliability need be sustained.

The effort required to sustain software in the long-term is dependent on code transparency, use of proper coding practices, internal and external documentation, version control, modularity, etc. It is therefore crucial that a software-sustainability policy recognises the need to improve the basic level of academic code development. In sum, criteria for selecting software to be sustained are:

- Relevance for a specific domain (impact on domain)
- Frequency of use
- Size and Complexity
- Usability
- Maintainability (quality and transparency of code)
- Relevance of the data produced with software (software to support data-stewardship)
- Future replication of published research results

Providing a Software Seal of Approval:

Poor maintainability is typically the result of cumulative minor violations of good software practice in documentation, complexity avoidance strategy, and basic programming practices that make the difference between clean and easy-to-read code vs. unorganized and difficult-to-read code. Maintainability requires adherence to software engineering best practices and technical attributes.

The Software Sustainability Institute (UK) provides an online guide on “Developing maintainable software” written by Steve Crouch: http://software.ac.uk/resources/guides/developing-maintainable-software including a “maintainability checklist” containing the following topics:

- Can I find the code that is related to a specific problem or change?
- Can I understand the code? Can I explain the rationale behind it to someone else?
- Is it easy to change the code? Is it easy to determine what I need to change as a consequence? Are the number and magnitude of such changes small?

- Can I quickly verify a change (preferably in isolation)?
- Can I make a change with only a low risk of breaking existing features?
- If I do break something, is it quick and easy to detect and diagnose the problem?

Developing a "Software Seal of Approval" (SSA) may be a useful mechanism for certifying quality software, which should be more eligible and easier to sustain. Such as quality seal could be modeled after the successful Data Seal of Approval (see: http://www.datasealofapproval.org), An SSA would need to be developed in an international context and to be based on a widely accepted checklist of criteria in order to gain acceptance as an useful quality hallmark.

Providing software-training:

As stated, a fundamental of software-sustainability is the need to adhere to high quality coding standards and development protocols. It is therefore necessary to provide easy, clear and obvious guidelines on how to build software so that anyone starting to write from scratch will adhere to some basic rules, for example regarding simple version control.

There are a growing number of PhD candidates developing new software to manage and exploit growing data resources across all research disciplines. Unfortunately, they often lack formal training in software development and testing (software carpentry). This can result in poorly developed software or "PhDware" as it is sometimes referred to, which is difficult to sustain as future developers find it hard to build on. PhDware is often poorly documented and lacks scalability due to inefficient coding. Therefore, a key component of software-sustainability and good research software practice, is to provide basic software carpentry training to all PhD candidates who would benefit from it. Software-carpentry.org (http://software-carpentry.org/) provides the basic model to deliver that training. Making it available, perhaps via the Graduate Schools, should be a key action.

Supporting the careers of software research engineers in academia:

For research software to have impact, it needs to be reliable, calibrated and accessible. The best research software is well engineered to ensure longer life and is also generic and reusable, so that it can function across research domains. However, software of this nature is not developed by accident. The increasing focus on data-driven and computing-intensive research practices has naturally resulted in an increased need for new scientific software development.

In the past, academia has attempted to employ commercial software engineers to provide the skills and manpower to develop necessary software. However, despite being trained in best practices and employing modern approaches to design, coding, testing and maintenance, software engineers in academia have often failed due to insufficient knowledge of the application domain or the research process itself.

There is clearly a growing need in academia for a new breed of digital researchers able to combine expertise in programming and software development with an intricate understanding of research. The Netherlands eScience Center has pioneered the concept of the "eScience Research Engineer" who are scientists with a strong drive to perform scientific research whilst also being very knowledgeable of the capabilities and use of advanced e-infrastructures. The comparable concepts of the "Software Research Engineer" or "Research Software Engineer" are beginning to be recognised, partly due to the advocate group “The UK Community of Research Software Engineers” (http://www.rse.ac.uk/index.html). Compared to typical post-docs, the software/eScience research engineers place additional focus and energy to produce technologies that are efficient, calibrated, reliable and accessible. They work according to coding standards, perform unit tests and dedicate sufficient effort to support applications with documentation and training.

Research Software Engineers tend to develop from one of two directions, either:

- Beginning their careers as researchers who, by chance or design, spend time developing software to progress their research.
- Beginning from a more conventional software-development background but are later drawn to research by the challenge of developing software to enable discovery.

The importance of the research software engineers is to ensure that academic software improves in quality and is therefore more sustainable. They are the key human component necessary for the professionalization of academic software development. They will also provide training across research to raise basic coding standards.

Publishing traditional peer-reviewed papers may be important to research software engineers, but they are primarily judged on the quality of the tools they develop. For this reason, in some cases, their personal goals do not fit well with the demands of the traditional academic system. The typical university system supports well those who write numerous scholarly publications. The researchers attracted to the research software engineer role are driven by developing excellent software applications, often at the expense of writing scientific papers.

Although their unique combination of skills is extremely valuable, their often-reduced publication record means they lack a formal place in the academic system. The Research Software Engineer works with researchers to gain an understanding of the problems they face, and then develops, maintains and extends software to provide the answers. However, they are sometimes excluded from being named in research papers despite playing a fundamental part in developing the software used to create them and generally lack the metrics needed to progress their academic career, like papers and conference presentations, despite having made a significant contribution through software.

Promoting citeable peer review for research software:

Presently there is no clear scientific credit for the creation of software within the existing academic evaluation. There is a clear need to develop and support alternative ways to review and disseminate software. At present it is often possible to publish new research software in domain specific journals or a few dedicated research software publications (Journal of Open Research Software; Journal of Software: Practice and Experience; Nature Toolbox; Scientific Programming; Original Software Publications; Journal of Statistical Software; and SoftwareX). Promoting publication of software in this manner, and developing additional channels, will help software gain suitable recognition and provide citations necessary to support the careers of software research engineers.

The Software-Sustainability Institute as role model:

The importance of software-sustainability and research software in general is being championed in the UK by the "Software Sustainability Institute" (http://www.software.ac.uk/). We are indebted to this organisation for numerous discussions on the subject and their work promoting these issues.

In their own words, *"The Institute is a leading international authority on research software sustainability, working with researchers, funders, software engineers, managers, and other stakeholders across the research spectrum. The use of software in research continues to grow, and the Institute plays a key role in helping the research community to help itself. We have built a network of over 60 Fellows (http://www.software.ac.uk/fellows) from across research disciplines, championed research software and software career paths to stakeholders, worked with over 50 projects to improve their codes (http://www.software.ac.uk/consultancy/consultancy-testimonials), written guides (http://www.software.ac.uk/resources/guides) on all aspects of software sustainability read by over 50,000 people, and organised training events for thousands of learners. Our mission is to cultivate better, more sustainable, research software to enable world-class research ("Better software, better research"). The Institute is based at the Universities of Edinburgh, Manchester, Oxford and Southampton, and draws on a team of experts (http://www.software.ac.uk/about/people) with a breadth of experience in software development, project and programme management, research facilitation, publicity and community engagement."*

Developing an equivalent initiative in the Netherlands, incorporating research software expertise, data archiving knowledge, an e-infrastructure presence, funding bodies and policy makers as well as representation from researchers should be a priority. Such an organisation could be formed in a "virtual" manner including representation from multiple existing parties.

About Data Archiving and Networked Services (DANS)

DANS promotes sustained access to digital research data. For this, DANS encourages scientific researchers to archive and reuse data in a sustained form, for instance via the online archiving system EASY (easy.dans.knaw.nl) and DataverseNL (dataverse.nl). With NARCIS (narcis.nl), DANS also provides access to thousands of scientific datasets, publications and other research information in the Netherlands. The institute furthermore provides training and consultancy and carries out research on sustained access to digital information.

Driven by data, DANS ensures the further improvement of access to digital research data with its services and participation in (inter)national projects and networks. Please visit dans.knaw.nl/en for more information and contact details.

DANS is an institute of KNAW and NWO.

About the Netherlands eScience Center (NLeSC)

The Netherlands eScience Center (NLeSC) is the national hub for the development and application of domain overarching software and methods for the scientific community. NLeSC develops crucial bridges between increasingly complex modern e-infrastructures and the growing demands and ambitions of scientists from across all disciplines.

The application of digitally enhanced scientific practices, referred to as eScience, is a fundamental toolbox for all researchers and is a prerequisite to ensure the Dutch knowledge sector remains competitive and the greatest return can be achieved from scientific investments. In support of this goal NLeSC funds and participates in multidisciplinary projects, with academia and industry, with optimized data-handling, efficient computing and big-data analytics at their core.

NLeSC is a joint initiative of the Dutch national research council (NWO) and the Dutch organisation for ICT in education and research (SURF).

www.eScienceCenter.nl