Opportunity Note

Encouraging a wider usage of software derived from research

The Committee for Open Science's Free Software and Open Source Project Group
November 2019

Abstract

Software is a hybrid object in the world research as it is equally a driving force (as a tool), a result (as proof of the existence of a solution) and an object of study (as an artefact).

This specific status means we need to define strategies, tools and procedures which are adapted to the various issues it raises. These include the citation of contributions to software design and production, the reproducibility of research results involving software and the wider usage and long-term sustainability of the software heritage created.

The place of software in research

The purpose of research is to produce new knowledge which can be conceived of by the human mind in all fields. It is based on scientific methodology, i.e. on the reproducibility of results so as to ensure these can be refuted if incorrect. The advent of computing - the science of efficient information processing - has opened up new pathways for scientists. Like telescopes in their day, computers have increased the scope of the attainable. Above all, the advent of software has made it possible to formalize abstract information processing operations in an unambiguous form so that they can perhaps be implemented by computers and shared within the scientific community and beyond for the benefit of all citizens.

Software therefore plays a triple role:

1. it serves as a **tool** in many areas by effectively processing various types of data to build and test models to support or invalidate hypotheses;
2. it can be a research **result** in its own right acting as evidence of an effective algorithmic solution to a given problem as measured by the capabilities of the computers of the day;
3. it can itself be a research **object**. The scientific community is particularly interested in the modes of software development and the proof of their properties especially regarding societal issues related to transparency and trust in computerized processing.

This means that scientists are increasingly producing research papers summing up their results and also software to support or demonstrate such results. This activity can represent a significant part of their work and must therefore be fairly taken into account when researchers are evaluated by their peers and institutional authorities.

Thanks to the development of digital networks, this software is increasingly built in a collaborative way. This is achieved either by the aggregation of a community of contributors or by reusing an ever-increasing number of software bricks which are themselves also very often built in a collaborative way. Modern software production thus aggregates people with multiple skills whose contributions can be of various natures. Software cannot therefore be reduced to a set of historicized additions of lines of code (or "commits"). The underlying dynamic driven by the various architects and leaders of the scientific software production project is an essential condition for its success.

These modern conditions for the creation of software strongly influence the legal status of the software produced. These specific features need to be taken into account in the definition of adapted technology transfer models which maximize societal impact even outside science.

Important issues

The specific characteristics of software production in research have given rise to several important issues:

1. To maintain software-related research output's reproducibility and refutability properties, the scientific community has to be provided with the means to **reproduce the experimental conditions** that led to such result and to test the algorithms concerned on other datasets. Guaranteeing **permanent access to both the software and the data** it manipulates means it must be possible:

   a) to **refer** to particular versions of the software used as well as their execution environments **on a long-term basis**;

   b) to possess platforms capable of **permanently storing** these versions;

   c) to possess hardware and system environments which allow software to be **re-used identically**. This is a complex scientific problem because the rapid obsolescence of hardware can have a strong impact on the reproducibility of certain types of results.

2. A well-adapted **citation** mechanism needs to be constructed to make sure the visibility and reputation of researchers take the time they spend producing software into account.

3. Implementing a policy and resources capable of **ensuring the sustainability and/or adequate technology transfer** of software products from public research including to spheres outside the scientific field requires:

   a) **reference methodologies** to assess the various possible methods of technology transfer illustrated by case studies and feedback;

   b) an **inventory** of these productions which is as widely accessible as possible.

Focus areas for work

Archiving and referencing

We now have solutions for the permanent archiving of software source codes and the precise referencing of source code versions for traceability and scientific reproducibility. These can be recommended for use by researchers in all disciplines[1].

The citation/reputation system

As explained above, software is the result of a complex process involving design and development activities[2]. It cannot be reduced to a sum of lines of code[1]. An architectural or algorithmic contribution may not appear directly as a formal production of lines of code because source code management systems only make the names of the developers visible. The **creation of traces relating to contributions** is therefore a technical problem (effective means for citation are required) as well as an organizational problem (these contributions have to be materialized within the development

[1] Contributions can be in the form of requests for new functions, anomaly reports following use in a new scientific context, porting to new platforms, ergonomic improvements to the interface, etc.

environment). This all requires metadata quality control procedures which are absent in repositories such as FigShare or Zenodo[3] but are currently being developed for HAL[4].

Also, certain aspects of the legal **status of software products resulting from research** need to be clarified. What are the interactions between the non-transferable and inalienable moral rights of researchers and the transfer or devolution of economic rights which automatically occurs when researchers are employed by a public-sector organization? What are the criteria for recognizing the authorship of contributors? One example would be that a person who has contributed to software by defining the models of the problems to be solved, designing the algorithms, defining the architecture of the software or directing the development work should obviously receive credit for this. However, how s/he also be considered an author in the legal sense of the term when he or she has not produced a single line of code? How does the law for a digital republic apply in the context of research software production?

The technology transfer of software produced by research

In terms of referencing, the software databases produced by research are often internal tools involving a mix of issues linked to referencing and the internal evaluation of researchers which prevents them from being opened up for broader usage. Attempts have been made to set up public databases[5] but this leads to duplicate data entries and data which is not always automatically updated. A **homogeneous foundation of open data** therefore needs to be defined which is possibly supplemented by standardized data for internal purposes. Providing different academic institutions with shared availability for this base would enable them to make a single inventory of the software assets produced by their agents which they sometimes co-own joint rights for.

At present, there is no uniform methodology for the transfer of software produced by research. After identification, it is therefore necessary to define **reference methodologies for technology transfer** based on existing mechanisms (free and/or private licences, creation of foundations or consortia, etc.), and **to share them with the actors concerned** (academic institutions' technology transfer units and SATT).

The long-term preservation of research software heritage

While software production is a research activity in its own right, software maintenance is not[6]. There is no guarantee that software will last once the scientific question which led to the software being created has been resolved, if the existing software no longer provides the new results researchers may hope for or if its designers move on to new projects. However, software that is no longer an object of research may nevertheless remain an effective tool to help other teams produce results or could even be used by companies as a development tool or used in an industrial or commercial context. The question of the maintenance and durability of software resulting from research must therefore be anticipated by both designers and users[7].

For users, it is essential to identify all the software that can play a strategic role in their business processes and to questioning the designers to ensure that maintenance and possible improvement services can be guaranteed. For designers, knowledge of usage by different categories of actors and its relative criticality must make it possible to evaluate the resources users would be willing to implement to maintain the software in operational conditions (porting to recent systems and environments, managing dependencies with third-party software, debugging) and support its evolution.

Often software developers do not possess sufficient tools to deal with such issues in their work and not enough is known about project support mechanisms. These can be flexible mechanisms such as the use of a consortium (see SSI[2] or ReSA[3]) or a foundation to raise funds and host dedicated staff or research structures directly providing manpower (engineering time). They may also involve the creation of a dedicated company or transferring publishing to an existing company either as the initial objective or in addition to previous schemes[4].

Pooling resources

Several past experiments have shown (such as the Depsy project [5]), that it is not efficient to consider the creation of the tools necessary for the above-mentioned purposes as development projects which need to be financed in their own right. The objective must be the creation of an **infrastructure** which is:

- **unique** to avoid any unnecessary or duplicated work even if the infrastructure is based on a distributed architecture hosted locally in multiple institutions;
- **sustainable** and **public** since the example of Google Code, a free software development platform which closed in 2015, demonstrated that the private sector cannot guarantee long-term sustainability.

Adding an access portal to this infrastructure would make it possible to conceptually and functionally link the different end objectives of preservation, cataloguing, referencing and dissemination/innovation transfer by addressing the different target user groups. These could be academic and industrial personnel who want a solution for their requirements, those who contribute to it or wish to do so, those who explore the stored data for scientific reasons, authority institutions and even the general public.

Work relating to these focus areas presented herein must be part of a sustainable framework for the allocation of human and financial resources as this is the only way to guarantee a return on the investment of the committed resources.

Recommendations

A number of recommendations can be made regarding the focus areas for work set out above. We need to:

Recommendation n° 1: Become involved **internationally** and work to set up **collaboration projects** on the subject.

Recommendation n° 2: Make **the specific nature** is recognized and not as "just data" particularly in the context of discussion about the notion of FAIR data.

Recommendation n° 3: Promote **archiving and referencing** best practices for research software.

Recommendation n° 4: Construct **a consensual definition** of a "**contribution**" to research software.

[2] One of the Software Security Institute's aims is to provide software engineering expertise to research software developers to increase their software's sustainability: https://www.software.ac.uk/

[3] The Research Software Alliance (ReSA) is a group of people involved in the production of software for research who want to see this activity fully recognized at the academic level. Their site lists the scientific advances made explicitly possible by the use of software: https://www.researchsoft.org/.In the United Kingdom the UK Research Software Engineer Association (UKRSE) is working with similar objective: https://rse.ac.uk/

[4] Like for example the transfer of the development of the Gazelle interoperability test platform to the company Kereval.

[5] The Depsy project had a sole funding envelope of $675K. No funding was planned for its long-term existence and when the project ended this automatically led to the end of its maintenance and evolution.

Recommendation n° 5: Build **tools** which integrate this notion of a contribution to be able to effectively credit authors/designers for their software contributions.

Recommendation n° 6: Promote a shareable **standardized metadata schema** for the software with a view to opening up software metadata derived from research.

Recommendation n° 7: Encourage academic institutions to share research software **metadata**.

Recommendation n° 8: Define a **common strategy** and **procedures** for evaluating open source software making it sustainable and encouraging technology transfer.

Recommendation n° 9: Encourage and facilitate the creation of **"legal toolboxes"** to ensure the long-term preservation of free software resulting from research.

Contact and distribution

The authors can be contacted by writing to roberto@dicosmo.org or francois.pellegrini@labri.fr

[1] Roberto Di Cosmo, How to use Software Heritage for archiving and referencing your source code: guidelines and walkthrough, https://www.softwareheritage.org/save-and-reference-research-software/

[2] Pierre Alliez, Roberto Di Cosmo, Benjamin Guedj, Alain Girault, Mohand-Said Hacid, Arnaud Legrand, Nicolas P. Rougier, Attributing and Referencing (Research) Software: Best Practices and Outlook from Inria, https://hal.archives-ouvertes.fr/hal-02135891v1

[3] Zenodo. https://about.zenodo.org/policies/

[4] Please see the description of the moderated depositing of research software in Software Heritage via HAL: https://www.softwareheritage.org/2018/09/28/depositing-scientific-software-into-software-heritage/?lang=fr

[5] Sophie Nicoud, Après PLUME: FENIX (Fiches d'Évaluation Normalisée Issues de l'expérience), https://resinfo.org/les-newsletters-de-resinfo/NewsLetter-3/Apres-PLUME-FENIX-Fiches-d-Evaluation-Normalises-Issues-de-l-eXperience

[6] Anna Nowogrodzki, How to support open-source software and stay sane, Nature n° 571, pp. 133-134, July 2019, doi:10.1038/d41586-019-02046-0. https://www.nature.com/articles/d41586-019-02046-0

[7] Dalmeet Singh Chawla, The unsung heroes of scientific software. Nature n° 529, pp. 115-116, January 2016, doi:10.1038/529115a. https://www.nature.com/news/the-unsung-heroes-of-scientific-software-1.19100