Open Science European Conference

Proceedings of the Paris Open Science European Conference
OSEC 2022

OpenEdition Press

Building the Software Pillar of Open Science

Roberto Di Cosmo

DOI: 10.4000/books.oep.16182
Publisher: OpenEdition Press, Académie des sciences, EDP Sciences
Place of publication: Paris, Marseille
Year of publication: 2022
Published on OpenEdition Books: 13 October 2022
Series: Laboratoire d'idées
Electronic EAN: 9791036545627

Building the Software Pillar of Open Science

Roberto Di Cosmo
Software Heritage/Inria/University of Paris

Thanks a lot for having me here today. It is a real pleasure to open this session about the relevance of software in open science. Let me start by providing a little bit of context. If we look around us, we see software everywhere. It powers our industry, fuels innovations and is essential to academic research. It is the fabric that binds our digital and professional lives together. It is thanks to software that we are able today participate in this conference despite the fact that we are in the middle of a pandemic. However, when we talk about software, sometimes we forget that it does not come out of the blue. It is not just a piece of data that comes out of an instrument. Software is written by human beings in the form of source code, which is a precious form of knowledge. It is actually a very unique form of knowledge, because it is built to be understood by humans and executed by machines. As Professor Abelson from MIT wrote in a beautiful book in 1985, 'Programmes must be written for people to read, and only incidentally for machines to execute[1]'.

What did he mean by saying this? Maybe it is easier to understand if we think of some particular examples of software source code. A piece of source code which is old is a fragment of the source code used on the lunar-landing module of the Apollo 11 mission that allowed us to put a man on the moon. Some of its text is very complicated to understand because it was assembly language of the kind used on the early machines of the 1960s. However, alongside the assembly code we find comments in human language that describe what the software is meant to be doing. This is a message from humans to humans. It is not just a message for a machine. More recently, if you look at programmes that are written with a higher level of programming language, like C, you can find beautiful pieces of software where the language has evolved over time. You have a name for the variable, a name for the function, but again you need comments to understand what is going on, although sometimes even with the comments it is not so easy.

Len Shustek, the founding chairman emeritus of the Computer History Museum, beautifully stated in his seminal 2006 paper on preserving software, 'The access to source code provides us with a view into the mind of the designer[2]'. This is very important to know because again it is human ingenuity that produces all this. It is not

just a tool. It is much more than that. The history of software is quite short, unlike many other disciplines. In the 1960s we had the chance to put a man on the moon, and this was done, by the way, thanks to a woman, Margaret Hamilton. She led the team of engineers who developed 60 000 lines of code used in the mission. Those 60 000 lines of code were enough to send a man to the moon and bring him back. Today, some 50 years later, a Linux kernel with over 20 million lines of code is one of the many components fount in the phones that we have in our pockets that allows us to send a smiley to a friend, or to send a message to somebody.

The reason why we had this lightning-fast growth is of course because software is changing the world we live in, but also because there is the open source software movement that started over 30 years ago. This led to an incredible collaborative effort by tens of millions of developers worldwide to work together and build the incredible software infrastructure we all use today. There is an old saying that we should build on the shoulders of giants, and we are doing so by reusing over and over again many components of previous work done by others in the very same spirit of open science, albeit this movement started much earlier than the term 'open science' was actually getting noticed.

I am particularly stressing this because sometimes you still find people who think that software is just a piece of data, a sequence of zeros and ones. It is not. It is much more. It is very special, very different. Software projects evolve over time. Some software projects may last for decades. The development history of how we changed it, who changed it, what, when and how is key to its understanding. The software we use today exhibits incredible complexity in different forms. It may be complex because it is a big piece of software with millions of lines of code. It can be complex because even a tiny programme may rely to perform its function on a broad spectrum of other subcomponents and subroutines, and each of these dependencies may be developed by a large number of other people. We should bear in mind that the software we use in research is just a thin layer on top of the general incredible set of software components developed by many developer communities around the world. To finish up on this point, again software is the fruit of human ingenuity. You cannot compare software source code to just a bunch of numbers that you got from one of your instruments. It comes from people working together! And even from the legal point of view, it falls under copyright law, unlike what happens with data.

Now that we have established the general picture, let's focus on the fact that software is fundamental in science too. People have started to notice how software is now essential in all disciplines. It is not just a matter of computer science: most of the research software we see is not written by computer scientists. It is written by

colleagues in many other disciplines. Today I think it is important to bring forward the message that when we are talking about open science, we really need to recognize that there are at least three essential pillars: of course open access to the articles published by our colleagues, and unfettered access to the data that is used in our experiments, but this would not be complete without a third essential pillar, which is the source code of the software which is used to manipulate, create and keep this data. It is the software pillar of open science.

Software in research is a multi-faceted object. It can be a tool used by somebody to create and analyse data. It can be the outcome of a research effort as proof of a result or because it embodies new algorithms or revolutionary data structures, or it can even be the object of research to see how software is built correctly. No matter which facet we look at, we need to have access to the *source code* of the software and so open source, which you could call something like open access to the source code, but it is much older than that, is really necessary. It is necessary to avoid reinventing the wheel and to accelerate scientific discovery, and for that we need to keep the history of all the source code built to enable reproducibility of research results, And this is essential to make it easier to accept the result of research because you can access the tools used to get these results.

If we look at the academic world, what kind of needs can we identify around software and source code? Depending on who you are – you may be a researcher, you may be a team leader or responsible for a laboratory, you might be running a big research organisation – you will need to have places where you archive and reference the software you are using in an article to make sure somebody else can find the same result you did. You want to get credit of course for what you did if somebody is using your software. You might want to reproduce a result from a colleague or build on top of it. These are all the kinds of things you need if you are a researcher. If you are head of a laboratory or of a team you usually need to produce a report, know what software is developed, maintain a webpage and track the software contributions. If you are a research organisation you need to know what software you are using, and what software you are contributing to, because it is important to have technology transfer in order to get an idea of what your impact on society is, because software built in research, as we will see later, is not just for research. Sometimes it has a direct impact on society. This is also needed to establish a funding strategy and to use for career evaluation.

In order to address all those needs there are many, many things that need to be done. I would like to start with what we could call the easy part. Of course, we need an archive, that is a place where you can actually store software and be sure that

you will be able to retrieve it later on. This is not what you can do by using the typical code-hosting platforms that everybody uses to develop software: projects stored there come and go, and even the platforms themselves come and go, they are not archives.

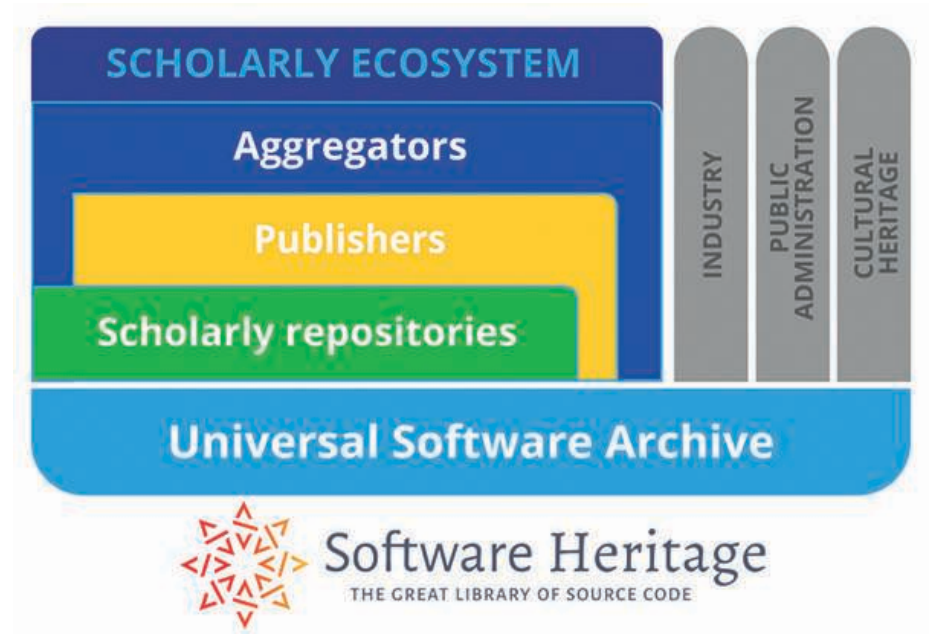
We need a way of referencing exactly, precisely the software artefact that we are interested in using to make sure that when we run it again, we can reproduce a result. Then we need to provide a proper description and proper metadata to make it easy to discover and reuse these artefacts. Finally, and this is touchier because it is connected to evaluation, we need to find the proper way to actually credit the contributions of those contributing to research software, which is not the same as just referencing a piece of software.

If we want to do this, a starting point is to have a look at what kind of infrastructures we need to support these kinds of cases in research software. Software is all over the place, so there are many ecosystems involved in software. There is a scholarly ecosystem, the one we are interested in today, and there are many others involving, for example, industry, public administration, cultural heritage and so forth. As analysed in a report published in 2020[3], after six months of work by a very broad European working group, you can identify scholarly repositories, like open access repositories, as well as publishers and aggregators that are all collecting and exchanging data. Their mission used to be focused on publications , and is now extending to data, but what about software? If you want to properly address software you need to base your work on a universal software archive that connects the scholarly ecosystem with all the other ecosystems and ensures that we get archival and reference, not just for the thin layer of source code that is developed in the research ecosystem, but for all the other components that are necessary to make it work.

The Software Heritage Initiative we launched some seven years ago comes into play here by archiving everything that is available around the world. It provides the common layer on top of which the scholarly ecosystem provides this added value that comes from curation, description, citation, credit and so forth. Here in France, we have been working for many years to implement this vision. There is a real workflow that can be used by researchers today in France to automatically archive their software and the software they need, not necessarily just theirs, into the Software Heritage archive[4]. They can also deposit proper metadata, which is curated, into the HAL national access portal[5]. This allows you to get a beautiful presentation of a software project with proper attribution, with reference to the institute that funded it, with a precise description of how to cite it and with a pointer that brings you to the universal archive, which is

Software Heritage, that provides you a full view of this software as another piece of research results and not just as a bunch of zeros and ones.

Fig. 15 : Research Software Infrastructures: Overall Architecture



Source: EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS, "Scholarly Infrastructures for Research Software", Website (Publications Office of the European Union, 7 December 2020), http://op.europa.eu/en/publication-detail/-/publication/145fd0f3-3907-11eb-b27b-01aa75ed71a1/language-en.

This example shows that these interconnections can be successfully established, and is also showing how to do it properly. If I may, I would like to make a call here to other people, other organisations, and other countries to join forces in the same kind of initiative. Let us try to avoid the major risks and major mistakes we have been taking and making in other areas, for example, falling into the trap of balkanisation. We should not build a multiplication of different and incompatible infrastructures and silos all over the place. Unfortunately, it is a big temptation for everybody to "just build his own archive", but then the result is that you will end up with duplicated objects in different archives with different identifiers and then you will need to spend a lot of money, time, and effort to try to build a federation after the fact instead of before the fact. We should avoid using closed or for-profit platforms that we cannot control, and we also should avoid using project money to fund operations, which are a completely different issue, in the research sector.

That was a look at things that can be done at the infrastructure level, but let me take a step back and focus now on the broader policy issues we also need to address. If we really want to have software playing its main role in open science, it is important to have policies for the dissemination and reuse of software developed for doing research. We really need to set the default to open source for research software. Open source creates value: look at the industry sector, where it is creating billions in value, and you will see that it is not incompatible with technology transfer. We just need to adapt our traditional way of doing technology transfer to open source.

We also need a framework for evaluation and recognition of researchers because unfortunately many countries still spend time developing beautiful high-quality software that is needed for research, yet it does not count in a research or engineer's career, and this needs to change. However, when we do this for evaluation, we need to avoid the mistakes that we have already made in the publication systems: in particular, we must avoid relying only on quantitative indicators, which are even more damaging in software than in other places. We also need to address the issue of the sustainability of open source on the technical, organisational, and financial levels.

There is, however, good news here because awareness is rising. In 2018, some 40 experts from all over the planet came to Paris to work on the Paris Call on Software Source Code as Heritage for Sustainable Development[6]. If you look at this Call, which was published on the UNESCO website back in 2019, one of the points it puts forward is the need to promote software development as a valuable research activity and recognise it in the careers of academics if they produce high-quality software. More recently, the report on Scholarly Infrastructures for Research Software from a working group created under impulsion of the European Commission called to make research software available as open source unless there are strong reasons not to do[7]. Moreover, in the recently published Open Science Recommendation from UNESCO[8] there is a call to use only non-profit, long-term infrastructure for open science and to operate on a community-based scale.

Implementation of these high-level recommendations has started already in France. If you look at the French 2nd National Plan for Open Science[9] there is now a chapter fully dedicated to software, which is on par with publication and data. Among the many recommendations is for creating a charter for research software policy at the national level and for recognising software development. This second point is already implemented, as you will see in the software awards just after this session, and there are many other significant provisions that I do not have time to delve into right now.

There would have been so many other things to say, but I needed to pick some that I think are very important. If you look at the road ahead, of course we need to spend more energy, money and time on building proper infrastructure for research software and recognising software as a key enabler of research, not just as a tool. This has many implications and I think Professor Lucke's talk will address some of these in the next presentation. Then of course we need to connect with the scholarly ecosystem, by linking software with publications and data. Here the role of the publishers is fundamental, but I would ask the publishers to take the time to consider that software is a noble research output, not just a piece of data, so you need to use specific infrastructures and identifiers.

Last but not least, at the institutional level we need representation and support, the same way this has been done for the other aspects of open science. We need an office in charge of the strategy around research software and open source, and not just in terms of technology transfer. We need to help our colleagues to do proper funding, governance and so forth. Finally, as for incentives and recognition in evaluation, valuing quality research software is possible, but again, beware of quantitative indicators. We do not want to have an s-index, as a software index, like the h-index in publication. There are other ways of doing this. For example, one is the software awards ceremony you will attend today. We really need to build the software pillar of open science together. The time has finally come. As always in academia, change takes time, but I firmly believe that together, if we work coherently together, we really can make it.

To go a step further

"The next step would be to see European or international collaboration in which academic institutions and research organisations would actually put together their effort and build on the common ground on archiving software, on referencing software, on providing proper metadata, on credit and citation, and on finding the right way to credit researchers for what they do in software development."

"The national awards given on the OSEC occasion represent the first time that at the highest institutional level we are putting in the limelight the people who have spent an incredible number of hours building the software that is essential for research today."

"Outside the fields of academia there are too many areas where software was seen as just a tool, and you do not have much respect for something that is just a tool. When you realise software is more than just a tool then you start noticing that you should

pay attention. We therefore now need that impulse. We have vice presidents for open science in universities, however, we do not have a vice president for open source in academia."

"There are many awards for free software in general, but this is the first time we are setting up an award for free software in research at the level of a ministry of research. The objective is to put the spotlight on the importance of software for research. It is a way of giving recognition to researchers who have done an incredible job over a very long time without proper recognition in academia. Software is very broad. It is used all over the place. Of course, we have prizes for software in other places, but we need them in academia because we need to build the software pillar of open science."

References

1. Harold Abelson, Gerald Jay Sussman, Julie Sussman, *Structure and Interpretation of Computer Programs*, The MIT Press, Cambridge, 1985.
2. Len Shustek, "What Should We Collect to Preserve the History of Software?", *IEEE Annals of the History of Computing* 28, nº 4 (October 2006): 112-111, https://doi.org/10.1109/MAHC.2006.78.
3. EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS.
4. "Software Heritage", accessed 24 March 2022, https://www.softwareheritage.org/.
5. HAL Science Ouverte", accessed 16 March 2022, https://hal.archives-ouvertes.fr/.
6. Software Heritage, "Paris Call: Software Source Code as Heritage for Sustainable Development", 2019, https://unesdoc.unesco.org/ark:/48223/pf0000366715.locale=fr https://unesdoc.unesco.org/ark:/48223/pf0000366715.locale=frhttps://unesdoc.unesco.org/ark:/48223/pf0000366715.locale=fr.
7. EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS, "Scholarly Infrastructures for Research Software".
8. UNESCO, "UNESCO Recommendation on Open Science", 2021, https://unesdoc.unesco.org/ark:/48223/pf0000379949.
9. Ministère de l'Enseignement supérieur, de la Recherche et de l'Innovation, "Second French Plan for Open Science", *Ouvrir la Science* (blog), 29 June 2021, https://www.ouvrirlascience.fr/second-national-plan-for-open-science/.