Report on the Workshop on Sustainable Software Sustainability 2021 (WoSSS21)

Software Sustainability Institute

netherlands eScience center

Data Archiving and Networked Services DANS

A workshop organised by:
The Software Sustainability Institute (SSI)
The Netherlands eScience Centre
Data Archiving and Networked Services (DANS)

6-8 October 2021
Online

Report editors:

Shoaib Sufi	University of Manchester
Carlos Martinez-Ortiz	Netherlands eScience Center
Peter Doorn	Data and Archive Networked Services (DANS)
Jessica Meyerson	Educopia Institute
Michelle Barker	Research Software Alliance (ReSA)
Daniel S. Katz	University of Illinois at Urbana-Champaign

Report authors; speakers and workshop contributors (alphabetical by first name):

Adam Jackson	Science & Technology Facilities Council (UKRI STFC)
Alexander Struck	Humboldt University of Berlin
Andrew Sandeman	Loughborough University
Andrew Stewart	University of Manchester
Andy R. Terrel	StoryFit
Ben Companjen	Leiden University
Carina Haupt	German Aerospace Center (DLR)
Carlos Martinez-Ortiz	Netherlands eScience Center
Carly Strasser	Chan Zuckerberg Initiative (CZI)
Carole Goble	University of Manchester
Christina Von Flach Garcia Chavez	Federal University of Bahia
Colin Venters	CERN
Daniel S. Katz	University of Illinois at Urbana-Champaign
Dianne Dietrich	Cornell University
Elena Colón-Marrero	University of Michigan
Emma Irwin	Microsoft
Euan Cochrane	Yale University
Fakhereh (Sarah) Alidoost	Netherlands eScience Centre
Fotis Psomopoulos	Centre for Research & Technology Hellas (CERTH)
Gerard Coen	Data Archiving and Networked Services (DANS)
Hilary Szu Yin Shiue	University of Maryland
Ignacio Blanquer	Valencia Polytechnic University
Jean-Noël Grad	University of Stuttgart
Jesse de Vos	Netherlands Institute for Sound & Vision (NISV)
Jessica Farrell	Educopia Institute
Jessica Meyerson	Educopia Institute
Kelly Rosa Braghetto	University of São Paulo
Konstantinos Repanas	European Commission
Martin Hammitzsch	Helmholtz Centre Potsdam
Meta Keijzer-de Ruijter	Delft University of Technology (TU Delft)
Michael Courtney	Catholic Diocese of Salt Lake City
Michelle Barker	Research Software Alliance (ReSA)
Morane Gruenpeter	Inria
Mustafa Doğan	University of Göttingen
Neil Chue Hong	University of Edinburgh
Nicolas M. Thiéry	Paris-Sud University
Otigbu Austine	National Archives of Nigeria
Pamela Nye	The Westminster Schools
Patricia Falcão	Tate
Paula Martinez	Australian Research Data Commons (ARDC)
Peter Doorn	Data and Archive Networked Services (DANS)
Rachael Ainsworth	University of Manchester
Raniere Silva	City University of Hong Kong
Scott Kirycki	University of Notre Dame

Shoaib Sufi	University of Manchester
Stian Soiland-Reyes	University of Manchester
Tom Honeyman	Australian Research Data Commons (ARDC)
Vicky Rampin	New York University

Table of Contents

1 Introduction

This report is based on discussions and presentations that took place at the Workshop on Sustainable Software Sustainability 2021 (WoSSS21). This edition of WoSSS was a fully online event due to it taking place during the COVID-19[1] pandemic.

The WoSSS workshop series aims to bring together participants from a broad range of communities that are interested in how to deal with software sustainability, primarily from the perspective of scholarly and scientific research.

WoSSS21 was not only oriented to research software developers, researchers who code, and specialists in digital preservation and research infrastructure, but also to policy makers in open science, research funders, and others who wanted to learn about the issues at stake and who have something to contribute. During WoSSS21 we discussed how we could best organise and support the community and emerging infrastructure for software sustainability. This year we paid special attention to software as heritage and compared the challenges of sustaining software in the domains of cultural heritage and research.

Section 4 contains the key recommendations coming from the workshop. Sections 6 through 9 constitute the core of this report. These sections include fully-featured write-ups that go beyond summary abstracts, where some authors have gone into more depth in their writing than others (either prior to or post workshop); these are categorised under a featured subsection.

The discussion sessions have been curated into Subsections 6.3, 7.3, and 8.3. There was a variation in the depth of the discussions and subsequent post-workshop elaboration of the notes, therefore in some cases the discussions are reported as key bullet points and in others they are reported as prose.

The last session of the event ended with a panel discussion and the key points of this panel can be found in Section 9.2.

This report is hosted in Zenodo (DOI: 10.5281/zenodo.7951155) and can also be found on the WoSSS website[2].

[1] https://en.wikipedia.org/wiki/COVID-19_pandemic

[2] https://wosss.org/#reports

2 About the Organisers

WoSSS21 was organised by Data Archiving and Networked Services (DANS[3]), an institute of The Royal Netherlands Academy of Arts and Sciences (KNAW), The UK Software Sustainability Institute (SSI[4]) and the Netherlands eScience Center[5].

More information about the WoSSS organisers can be found on the WoSSS website[6].

[3] https://dans.knaw.nl/en
[4] http://www.software.ac.uk
[5] http://www.esciencecenter.nl
[6] https://wosss.org/partners

3 Methodology

The workshop balanced authoritative views in the space of software sustainability with a strong participatory element.

The workshop was divided into four sessions, covering key topics on software sustainability:

1. Sustaining software in cultural heritage,
2. Open Science & applying the FAIR principles to software
3. Human factors and new development in preserving and sustaining research software
4. Sustaining the community and promoting (human) infrastructures for software sustainability

Each of these sessions featured invited speakers who provided an overview of the topic. The first three sessions also provided space for discussion while in the fourth session a Q&A panel was held where panellists shared their views and workshop participants were invited to participate via an interactive survey. The full programme for the workshop is available on the WoSSS21 website[7].

Some of the discussions are just referenced as key points and related links, whereas others where the discussion participants contributed text during the session and followed up with changes to make the text a coherent piece are stated in full.

After the workshop, the transcript from the presentations and the documents from the discussion sessions were used to produce the first version of this report.

This was then sent for review to the authors and participants to give them an opportunity to expand their summary text and add to their discussion text. A series of reminders were sent and then those who had provided updates from their summaries were classed as 'Featured' and the 'Discussions' sections are either key points or fuller prose depending on how much was provided. The wider group of report editors reviewed the final draft and after updates the final version was prepared.

For each talk, the associated video, a lightly corrected transcript, and any slides are linked from the full programme on the website.

The report editors then looked at the main themes and recommendations from each of the talks, the summaries and featured sections, discussion topic and panel Q&A to put together a list of key points in 4.2 Key recommendations and an associated 4.1 Executive summary.

[7] https://wosss.org/wosss21/agenda

4 Executive summary and key recommendations

4.1 Executive summary

WoSSS21 focused on software in cultural heritage, open science, the FAIR principles, human factors, new developments in research software and human infrastructure. The key recommendations deduced from presentations, discussions, and the panel at the workshop and some of the reflections of authors after the event. We cover them in brief in this executive summary and they are in detail below, in Key recommendations.

There are key stakeholders in sustaining software sustainability efforts. The stakeholders identified include funders, the galleries, libraries, archives and museums (GLAM) sector, those working on applying FAIR principles to software, the communities of practice involved in software and related efforts, centres of excellence in open source and open research as well as organisations. Software sustainability stakeholders come at all different levels and include individual developers, researchers, data stewards and then teams, projects, and domains, all play a part in how software is sustained through their practices and agreed norms and are impacted by the ongoing support of sustaining efforts to produce more sustainable software.

Sustainability is a complex endeavour requiring collaboration amongst stakeholders and a multidisciplinary approach that can mix research practices, technical skills, ethics, and software engineering. Laws can also have an impact on sustainability with legal exemptions around copyright for archives aiding their preservation work. The topic of sustainability is also known as “long-lived software” and it’s important to be cognisant of this body of knowledge to have a full picture of the work done on sustainable software. Tools such as Reprozip that capture software dependencies, and Research Object Crate that capture and relate additional metadata for research workflows are making sustaining software more tractable. The Citation File Format (CFF) and the Journal of Open Source Software (JOSS) highlight software in research and move software closer to being a first class research object (alongside publications) and increase the importance of sustainability.

The FAIR principles of (Findability, Accessibility, Interoperability and Reusability) originated in the data space but they are now being applied to research software. We now have the FAIR for Research Software Principles and these aid in the reproducibility, preservation, and communication of research software and therefore its sustainability.

Human factors are arguably the most important aspect of software sustainability; without skilled people, who have real career paths working in empowering and safe environments that guard against burnout software cannot be sustained. Metrics such as those by Community Health Analytics in Open Source Software (CHAOSS) help evaluate the health of open source efforts to promote equity and inclusion and can be used to monitor project to improve the environment for the people involved. The increase in software oriented roles (e.g., Research Software Engineer, Data Steward, Information Scientist) are a sign of improving acknowledgement and acceptance of software as a key part of research. The Research Software Alliance (ReSA)'s People Roadmap provides an overview of community initiatives in the research software ecosystem. It's excellent that new initiatives are also starting such as the UK Reproducibility Network (UKRN) which promotes open and reproducible research practices and aims to evaluate organisations' performance in terms of open/transparent practices, and this includes the human factors (e.g., credit).

4.2 Key recommendations

The following recommendations have been generated by the workshop organisers, based on the presentations, summaries, discussions in the four sessions, and featured contributions from presenters after the workshop. They are presented in alignment with those sessions and their topics.

The key recommendation is followed by a sub bullet point detailing which stakeholders we believe should lead on this recommendation.

Sustaining software in cultural heritage

1. Collaboration, knowledge sharing, and enthusiasm are essential for driving positive changes in the adoption of sustainable software practices.
    - These should be actively supported and encouraged by funders and organisations.
2. Properly resourcing digital preservation and cataloguing is crucial for improving accessibility and usefulness of historical software code and programmes.
    - Funders, organisations, and projects should recognise the value of these tasks and provide adequate resources to support them (e.g., by supporting the work of Software Heritage[8] archive).
3. Digitisation and data loss are critical issues. They deserve attention, particularly in low and middle-income countries where resources are often lacking. It must recognize that a country's historical records are part of our global heritage, that need preserving for future generations.
    - Richer nations need to allocate the necessary resources to address these issues. International organisations such as the UN and OECD need to take a proactive role in raising awareness and bridging the gap between better-resourced nations and those with a global need.
4. Digital art preservation and conservation is a multidisciplinary craft that requires collecting, organising, and describing both the conceptual and technical aspects of these works for their artistic and historical value. This effort

[8] https://www.softwareheritage.org/

necessitates training in scientific practices, technical skills, ethics, software engineering, and software sustainability.
    - Funders, organisations, and professional bodies should support this work with resources, opportunities to develop curricula, and platforms for collaboration.
5. When developing policies related to software preservation and sustainability, it is essential to consider existing organisational policies and identify any potential overlaps or conflicts (e.g., with data retention policies), to ensure a comprehensive and consistent approach.
    - Memory organisations [9] such as the GLAM (Galleries, Libraries, Archives, and Museums), universities and public research organisation need to take this comprehensive approach.
6. Infrastructure built to support software preservation and sustainability, as demonstrated by EaaSI [10] (Emulations-as-a-Service Infrastructure) should include GLAM professionals in the conversation to ensure that the infrastructure developed aligns with their preservation goals.
    - Infrastructure developers need to include users and use cases from the onset and memory institutions need to have capacity to engage in such conversations.
7. Legal exemptions for copyright for archives are essential in enabling the preservation and access of software in cultural heritage contexts.
    - Memory organisations must lobby governments and policy makers to maintain them to ensure the continued preservation of software in cultural heritage contexts.
8. Despite challenges that can arise when publishers seek to distribute software works, such as codebase contamination (e.g., incompatibly licensed software being bundled together), rectification is necessary to ensure software preservation and access for research and educational purposes.
    - Publishers need to realise the importance of this and put effort towards maintaining the ability to distribute code.

Open Science & applying the FAIR principles to software

9. The FAIR4RS Principles [1] provide a crucial standardisation framework to make software reproducible, preserve it and communication it's difference to data.
    - Individuals, projects, and organisations need to adopt these principles
10. The adoption of FAIR (Findable, Accessible, Interoperable, and Reusable) practices in research software are crucial pillar for modern open science; they can enable increased transparency, collaboration, innovation, and impact.
    - Policy makers need to actively support these FAIR practices in the research community.
11. The risks of developing metrics for FAIR software are software development projects solely optimising for findability, accessibility, interoperability, and reusability as a tick-box exercise, potentially neglecting other crucial software characteristics such as quality, open licensing, modularity, and ease of contribution.

[9] https://en.wikipedia.org/wiki/Memory_institution#Memory_institutions_in_the_Digital_Age

[10] https://eaasi-sandbox.softwarepreservationnetwork.org/eaasi/

- Individuals, projects, and organisations should avoid evaluating software based solely on FAIR metrics.

12. Reproducibility is vital for establishing trust in research. There need to be more opportunities for peer review and career credit for researchers who embrace the software aspects of reproducibility.
    - Publishers, funders, and organisations need to reward such practice.

Human factors and new development in preserving and sustaining research software

13. "Inclusion bugs" are inadvertently excluding people without realising (e.g., ordering meat only for lunch and forgetting about vegetarians), steps need to be taken to mitigate them. The CHAOSS[11] metrics can help identify and address issues related to diversity and inclusion in project and organisational practices.
    - Projects and organisations need to be mindful of “inclusion bugs” and adopt practices and metrics to help avoid them.
14. Toxic behaviours need to be addressed to support a positive culture, productivity, and a respectful work environment.
    - Organisations, projects, and workshops should implement and enforce codes of conduct.
15. Detecting Burnout in colleagues and community members needs awareness and realistic options (e.g., allowing people to take a step back) to foster supportive environments where goals can be met in a more sustainable way. Individuals who may have fewer safety nets, as they maybe more vulnerable to the negative effects of burnout.
    - Leaders, co-workers, and community members must support and take care of each other.
16. Community cohesion is important to maintain progress, motivation, and momentum. Serious differences of opinion can divide communities, diplomacy and compromise should be the first course of action. In situation where compromise is leading to stagnation perhaps this is the time to split the community to allow for different ideas to have the space to grow.
    - Individuals in communities need to focus on finding common group to achieve shared goals with the wisdom to realise when communities should bifurcate.
17. There is a growing demand for Research Software Engineering, Data Stewardship, Information Science, and Data Curation in the research sector. Although the maturity of these roles and career paths may differ across the world, the UK with along with some countries in Europe are currently leading the way, followed by the US.
    - Policy makers, funders, and organisations should be aware of this positive trend and take steps to encourage the development of these roles and career paths, including providing funding and support for projects and training in these areas.

Research Software (across all sessions)

[11] https://chaoss.community/

18. Complex software architecture is acceptable but complicated and chaotic architecture is not sustainable in the long term. To avoid the introduction of technical debt that cannot be repaid or architectural decay and drift,
    - Projects, developers, and technical leads must be mindful of the architecture they are creating to ensure maintainability and scalability.
19. The software architecture research community has been interested in long-lived software for a while, which overlaps with software sustainability. Avoiding duplicating efforts should be a goal.
    - The software sustainability community must collaborate with other fields that are approaching software sustainability from different perspectives.
20. ReproZip[12] captures the environment, libraries, and dependencies of running code to create preservation-ready bundles that enable replay without requiring extensive expert help.
    - Researchers and the creative industries using computation can use such systems to support reproducibility efforts.
21. Reproducibility in research analysis can be enhanced by integrating different parts of analysis written in different systems and platforms into one cohesive whole using workflow languages such as CWL[13] or workflow systems such as Galaxy[14].. This should be enriched with additional metadata that links pipelines to input data, parameters, results, provenance, authorship, experimental information and more. Metadata and workflows can be combined into Research Objects (e.g., RO-Crate[15]) using web technologies such as JSON-LD[16] and Schema.org [17] derived terms to create a bundle that supports active reproducibility and inspection.
    - Project teams, domain, researchers, and developers should consider taking this approach on the research projects they work on.
22. The use and adoption of Research Objects to support the evaluation and reproducibility of results should be encouraged.
    - Publishers should enable this to allow peer reviewers and readers this affordance.
23. Researchers should be actively assisted in adopting open and FAIR practices by being provided with guidance on tools, licenses and best practices.
    - The Open Source centres of excellence such as the FLOSS Competence Center in Brazil [18] serve as an excellent example for middle-income countries and beyond to support the adoption of more transparent practices within research communities.
24. Software policies should be given the same level of urgency and importance as data policies. This is essential in ensuring the secure and efficient management of software assets.
    - Organisations should take note of this to allow the safeguarding of their software assets.

[12] https://www.reprozip.org/

[13] https://www.commonwl.org/

[14] https://galaxyproject.org/

[15] https://www.researchobject.org/ro-crate/

[16] https://json-ld.org/

[17] https://schema.org/

[18] https://ccsl.ime.usp.br/

25. Progress towards establishing software on an equal footing with papers such as the Citation File Format (CFF[19]) and The Journal of Open Source Software (JOSS[20]) should be supported and celebrated.
    - Funders, domains, infrastructure providers and organisations should take note.

Sustaining the community and promoting (human) infrastructures for software sustainability

26. Establishing grass roots research-led organisations that actively identify and address gaps in incentives, training, and organisational performance is crucial to promoting open and transparent research practices.
    - Funders should support these; an example of such an organisation is The UK Reproducibility Network (UKRN[21]) funded by Research England.
27. Human aspects of software sustainability (human infrastructure) are fundamental to ensure sustainability of software. The ReSA people roadmap[22] provides an overview of the landscape of community initiatives in the research software ecosystem.
    - National and international funders, policy makers and organisations need to support these infrastructures in the transition to open science.

[19] https://citation-file-format.github.io/

[20] https://joss.theoj.org/

[21] https://www.ukrn.org/

[22] https://zenodo.org/record/5633318

5 Background to the workshop

The Knowledge Exchange Group (KEG) deserves credits for organising a first workshop on software sustainability in 2015 in Berlin. The results of this workshop were published as a report[23].

This workshop was followed by the first (WoSSS17[24]) in the Hague, organised and co-sponsored by DANS and the Software Sustainability Institute (SSI) in the UK, which renamed the activity as the Workshop on Sustainable Software Sustainability (WoSSS).

A third workshop (WoSSS19) was then co-organized and co-sponsored by DANS, SSI and the Netherlands eScience Center.

WoSSS21 is the fourth in this series of workshops looking at practices in software sustainability that include representations from memory institutions, research software, and research infrastructure.

[23] https://www.knowledge-exchange.info/event/software-sustainability

[24] https://easy.dans.knaw.nl/ui/datasets/id/easy-dataset:75828

6 Sustaining software in cultural heritage

This section describes the first session of the workshop. This session includes featured sections (where the authors have re-visited and expanded the text after the workshop) and summaries submitted for the workshop that cover the topic of cultural heritage. The featured sections include topics on software preservation as collective action, work at the Computer History Museum in the US and preserving documentary history in Nigeria. There are summaries including how software is sustained in context of art, and the work of the Software Heritage Archive in France.

The discussions sessions which occurred after the presentation in this session cover software preservation policies, challenges, the role of GLAM (galleries, libraries, archives, and museums) and intellectual property.

6.1 Featured

6.1.1 Software Sustainability as Collective Action

Jessica Farrell[25] *& Jessica Meyerson*

As a framing, sustainability does tremendous work for the international community of stakeholders that care about software as both 1) a dependency to make meaning from existing data and scholarship, and 2) as an output in its own right, as a cultural heritage object. It does this work - because sustainability doesn't just imply the maintenance of software over the longer term, sustainability is a way of thinking that begs us to see our relationship to software in multiple dimensions - social, environmental, and economic - as well as across time[26]. In the communities that I facilitate, sustainability frames not just how we preserve software and born-digital records, but also how these goals can align with those of environmental sustainability[27], and how to sustain our own selves as humans with limited attention and capacity but a desire to relieve the tensions and barriers to software preservation.

Drawing from information science, software sustainability practitioners operate within a "records continuum." The records continuum model (RCM) holds both software's active life (fulfilling its originally intended purpose) as well as its potentially long tail of reuse by resisting the false binary of active and archival in the first place. The continuum assumes that from the moment of creation, a record (in this case software) is both an active record and a historical one. Like sustainability, the records continuum model begs us to approach software as part of larger cultural, environmental, political, and legal processes.

Additionally (and importantly), the RCM asks us to consider our relationship to software at different scales of human life including individual, group, community, organisational, institutional, national, and international scales.

How does the growing landscape of software sustainability specialisations, tooling, and processes apply across the software continuum? How do we use the continuum

[25] Links to slides, video & transcript available at - https://wosss.org/wosss21/S1-JessicaFarrell

[26] https://educopia.org/cultivation/

[27] https://bitcuratorconsortium.org/workshop-enacting-environmentally-sustainable-digital-preservation/

model to inform software sustainability activities at each scale of human life? What are the tensions shaping software sustainability activities at the organisational and international scale?

In the Software Preservation Network, we believe that collective action is necessary to situate software and other born-digital material at the various scales in the RCM, and to address the tensions that we see around achieving that goal. But in our work on Emulation as a Service Infrastructure, we have now begun to map the boundaries of collective action - the boundary between what a consortia or a service or an interorganizational community of practice is in the best position to do versus what really needs to take place locally - informed by local user constituencies and institutional realities. Now that SPN has gathered model software preservation workflows/case studies, policies, and advocated for exemptions to US Copyright law -- and the EaaSI platform is real, wrapping up a pilot with additional cloud hosted emulation nodes - we are faced with a new challenge - how do we (re) situate these collective outputs back into the very local, very specific contexts of cultural heritage institutions in ways that enable those same institutions to provide useful, sustainable emulation services?

This is where field-level scale is de-emphasized and the local, interrelational scales come to the forefront. Participatory Archival Research & Development (PAR&D), an orientation/approach to cultural heritage work that speaks to how these communities work, emphasises - “There are more general ways and specific strategies/standards that help to cultivate trust (community archiving, post-custodial stewardship, trusted repository audit certification) but we sustain trust through reflexive practice, by making reflexivity business-as-usual: questioning our assumptions/our rationales, and critical evaluation, of documenting those rationales and making that documentation visible.”

Collective action got us to where we are with the development of these resources and development of communities like the Software Preservation Network and BitCurator Consortium[28]. But when we re-focus on the local context, that is the work of building power for future collective actions. Sustaining and preserving the world’s software requires the commitment of many, many individuals that hold various levels of influence and power. We build power by building trust, and it’s not a very technical activity. It’s actually quite simple - from the constant activity of checking in, chatting with your colleagues, understanding where everyone is coming from, you build trust and collect the ingredients required for successful collective action later. I also see power built through sharing knowledge in spaces like this, in training efforts, and in sharing resources that can be reused.

We must imagine the funding, capacity, and even knowledge that we want ourselves and our organisations to have in the future, to develop pathways to materialise these goals. So, we imagine this future, and we are all collectively working to get there in small ways by sharing knowledge and enthusiasm with our colleagues, and in big ways by holding public events like this one to amplify that activity many times over.

[28] https://bitcuratorconsortium.org/

6.1.2 Software Preservation at the Computer History Museum

Elena Colón-Marrero[29]

The Computer History Museum has collected historic software since the museum's founding in 1979 by virtue of collecting computers. In 2017, the Software History Center was launched with a focus on collecting and interpreting software materials. The museum's collection contains everything from paper tape and punched cards to CD-ROMs, DVD-ROMs, and source code with a heavy emphasis on PC software totalling over 2,500 linear feet (nearly 800m) of material. The biggest limitation to CHM's software preservation work is limited resources: monetary, people, and time.

In 2016, there was very little intellectual control over the software collections. Lack of intellectual control makes it very difficult to do any work such as disk imaging or emulation. As you can imagine at 2,500+ linear feet a massive effort is really needed to describe the collection. Over the course of two years an inventory of the software collection and cataloguing instructions for describing software were created.

The software inventory focused on the items in the collection that were catalogued in some level to determine which items needed more focus. The review of those records informed the creation of an instruction manual for staff and volunteers to use when cataloguing software materials. The software cataloguing manual had to adapt to the limitations of the museum's collections management system, as well as the fields in use by other object types. Due to these limitations no metadata standards were followed due to the lack of mapping abilities.

Digital forensics workstations were created to help the collections department image software materials, but also process any other types of born-digital materials the museum may receive. Sample workflows, naming conventions for files, and folder structures to establish a process from cataloguing to final ingestion of images into our Digital Repository were created.

The collections department created an internal case-study to determine the viability of disk imaging and software emulation on our collection. It was found that on average the process of cataloguing and disk imaging a software package took 2-3 hours of staff time. Software items with significantly less available metadata or number of disks would take less time but involve at least an hour's worth of time. We also attempted to emulate some of our software but found that with limited staff and time it was not something that we could feasibly do or expand on. It just took too much work when that time could be better spent on cataloguing, reference, managing our digital repository, and more.

However, with COVID-19 and the museum closed, a lot of the museum's software preservation efforts are placed on hold. Without access to the building or materials it was difficult to catalogue or image items. Increasing focus of the collection's staff time shifted towards building a Digital Asset Management system and a new Collections Management system, including record clean-up efforts. At the current moment software preservation efforts at the Computer History Museum are stopped until new

[29] Links to slides, video & transcript available at - https://wosss.org/wosss21/S1-ElenaCol%C3%B3n-Marrero

staff and/or the DAM (Digital Asset Management) and CMS (Content management System) migrations are completed.

6.1.3 Preserving our collective documentary heritage in bits, putting a step forward

Otigbu Austine[30]

As the cornerstone of history, Archives are an invaluable national heritage for human society. With proper storage, preservation and access, such records become veritable tools for appreciating the past, understanding, and dealing with the present and projecting for the future. Archives constitute a vital part of the memory of a nation, its people and institutions for cultural growth and development. As the apex archival institution in Nigeria, the National Archives continues to make deliberate efforts to salvage and preserve these records of perpetual value for easy accessibility using different preservation techniques and methods.

Prior to the Covid-19 pandemic, the National Archives of Nigeria deploys traditional methods of records management and archives administration which was essentially paper based in preserving and disseminating information materials to its users. However, the Covid-19 pandemic has driven the National Archives of Nigeria to rethink its archiving methods and processes, thus, embracing automation and digital preservation methods to facilitate records management and archival functions to ensure a more sustainable, simplified, and time saving process of preserving and disseminating our national documentary heritage to its local and foreign users who had limited access to this vital information resource during this pandemic period. While Automation involves understanding and integrating archival functions/tasks performed physically into electronic tools or machines, digitization requires the knowledge and professional use of electronic tools to salvage and preserve the documents in digital format for as long as necessary. However, both require technological and human resources to drive the process.

6.1.3.1 Implementation

Saddled with the task to come up with a workable framework to implement this project, my team and I adopted the Digital Preservation Management Model approach [2,3] to ensure a robust and inclusive process of implementing the project. The Digital preservation management model is a 3-legged approach which clearly outlines three main areas of work (Technology, Organisation and Resources) for the successful implementation of the automation and digital preservation process.

6.1.3.2 Technology

Technology is an integral part of automation and digital preservation, a good understanding of the tangible (hardware) and the intangible (software) aspects of technology to be adopted will be crucial to the success of this exercise. Considering the scarce resource of the National Archives of Nigeria, a greater consideration was given to open source software to ensure future sustainability of this project.

[30] Links to slides, video & transcript available at - https://wosss.org/wosss21/S1-OtigbuAustine

6.1.3.3 Software Deployed - ATOM

T
he Access to Memory (ATOM) 2.4 version was considered suitable for the repository system. This common information storage and retrieval software package was specifically developed by Artefactual Systems under the governance of the International Council on Archives for long-term archiving. It was developed as an open source web application with an idea to enable standardised and controllable creation of different levels of description of archival collections, holding all relevant information about the fonds. Hence, this software program contains general rules for archival descriptions regardless of the type or form of the archival records. Furthermore, it provides means for preparing a very detailed description of records as whole and parts through the following basic entity types and their interactions: Access records, archival description, authority records and archival institutions. Please visit National Archives of Nigeria Online[31] for a better insight of work done so far. Also, software was deployed for security encryption and integrity check purposes.

6.1.3.4 Organisation

At the organisational level, management is committed to the success of this project, to this end; management is drafting a digital preservation policy framework to ensure legal and regulatory compliance. Resources within the scarce capital resources of the National Archives of Nigeria a small amount of equipment was procured to start this project. The equipment constituted: two desktop computers, two scanners, one server, and one router. Considering the volume of information documentary heritage to be digitally preserved, the sustainability of this project seems to be in doubt. Therefore, an alternative source of funding is needed to ensure the continuity of this project and the Archives are open to exploring this.

6.1.3.5 Conclusion

The immediate focus of this project is to digitise our collective national documentary heritage for digital preservation as long as necessary, a key focus is on the endangered archives that may be lost forever if urgent actions are not taken to salvage them. Amid the funding and human resource shortage crisis faced by the National Archives of Nigeria over 1000 endangered archives have been digitised and are waiting to be transferred to the digital repository for easy access by users. Lastly, we shall rely greatly on friends of the institution for support particularly in the equipment and financial support where necessary. This transition project is tentatively to last 24 months, after which it will be reviewed.

6.2 Summaries

6.2.1 Software Sustainability in the context of Software-based Art Conservation

Patricia Falcão[32]

[31] https://nationalarchivesofnigeria.org.ng/

[32] Links to the video & transcript available at - https://wosss.org/wosss21/S1-PatriciaFalc%C3%A3o

Software sustainability and Conservation of software-based art have many similarities and some key differences. This talk will provide my perspective as a conservator and researcher working in the conservation of software-based art and explain my view of the differences and similarities, and where I see space for close collaboration.

Artist's software is as varied as the artists themselves and their teams. Even a small collection like Tate's, that now contains less than 15 artworks, where the earliest is from 2013, includes works created in and for five different Operating Systems, applications created using Java, Delphi or C++ and tools such as Director or Unity. The functions of the software vary between creating randomness in drawings in the work Becoming (2003) by Michael Craig-Martin and choreographing the movement of an oversized puppet in Jordan Wolfson's Colored Sculpture (2016)

A conservator in an institution needs to learn how to preserve both the conceptual and material aspects of these types of works, for their artistic and historical value. This includes documenting the systems used and how to run and calibrate the software, with the aim of ensuring that the artworks that the software creates can be displayed "in perpetuity". This implies the preservation of running systems, in a gallery, for the public. Often there is the need to change the software that runs an artwork, and the role of the conservator is to understand how those changes may impact a work's meaning and ensure that its behaviours are as little changed as possible.

Conservation practitioners are trained within a now fairly long tradition, centred on scientific practices, technical skill and thorough documentation processes. Underlying all these aspects is a code of ethics that guides decision-making when intervening in an object. For more traditional objects, the aim is to prevent change, but contemporary art practice, not just software-based art, has questioned this aim and for any digital or media-based work conservators strive to manage change which, given the dependence of this type of objects on mass-produced technology, is accepted as inevitable.

The field is new, with the first research in the area happening in the early 2000s, but the landscape is changing rapidly, with new specialised degrees opening or about to be opened. The existing practitioners are adopting and adapting practices from Digital Preservation, Software Engineering and Software Sustainability and ensuring that how those practices are applied still reflects the conservation code of ethics.

6.2.2 Software as a first class research output in a FAIR ecosystem

Morane Gruenpeter[33]

Software is a significant and vital component of research. It is integral to all stages of research and can play the role of a tool, a research result, or a research object. Since software source code has been recently recognised as an important asset in the field of scientific research, complementing publications and research data, it is essential to collect and preserve it.

[33] Links to slides, video & transcript available at - https://wosss.org/wosss21/S1-MoraneGruenpeter

Software Heritage (SWH [34]) is the universal source code archive: collecting, preserving, and sharing the largest collection of source code. Software Heritage is now providing the infrastructure for depositing and referencing software source code, in collaboration with national and international open access portals.

In parallel, the RDA, ReSA and FORCE11 FAIR for Research Software working group identified divergences between software and data and the crucial need to translate the FAIR data principles to be relevant for software artefacts. In September 2021 the working group published [4] after a community review.

Finally, the importance of [5] is a common goal to have better recognition and interoperability of software in a FAIR ecosystem.

6.3 Discussions

6.3.1 Software preservation policies

Gerard Coen, Michael Courtney, Dianne Dietrich, Elena Colón-Marrero

6.3.1.1 Key points

- Explore how existing organisational policies apply to software.
- Software outputs produced and their priority in an organisation should guide the production of software specific policies

6.3.1.2 Related resources

Memento Protocol can be.

- Research software sustainability in the Netherlands: Current practises and recommendations[35]
- TU Delft Research Software Policy[36]
- TU Delft Guidelines on Research Software: Licensing, Registration and Commercialisation[37]
- A Research Software Agenda for Australia[38]
- Memento Protocol[39] - useful for looking at prior or archived versions of policies and resources which is useful in this context.

6.3.2 Software preservation challenges specific to the cultural heritage sector

Patricia Falcao, Jesse de Vos, Morane Gruenpeter, Hilary Szu Yin Shiue, Colin Venter, Carlos Martinez

The cultural heritage sector and scientific research share a common goal of preserving software for future use. However, there are different needs within these sectors: in

[34] https://www.softwareheritage.org/
[35] https://doi.org/10.5281/zenodo.4543569
[36] https://zenodo.org/record/4629662#.Y4SOWuzP2WY
[37] https://zenodo.org/record/4629635#.Y4SOb-zP2WY
[38] https://ardc.edu.au/program/research-software-program/
[39] http://timetravel.mementoweb.org/

some cases, it is desirable to keep the exact code and to be able to reproduce the original answers; in other cases source code is seen as a means to that end, and what matters is the visual/experiential result is more important. In the arts, there is also the Mona Lisas, where you want to know details about the artist's process, techniques.

Depending on the specific needs of a particular area, different solutions might be suitable:
In cases where a running copy of the software is necessary, emulation and virtualisation are perfectly valid ways to ensure reproducibility.

In other cases, documentation can play a big role. For example, rather than preserving an actual game, recording a video of that game being played, having the authentic colours, speed and frame rate and other information can be more important and better than a usable game.

Yet another case is where source code is both an object of research and heritage, an example of this is the Apollo 11 software[40]. The Software Heritage Acquisition Process[41] aims to recover and curate landmark legacy source code and address challenges for all codes before 'Archaeology' for software/digital forensic are needed to piece together what the intentions might have been.

These different types of preservation all present a problem: the fact that software preservation is not always part of current practice. Artists are mostly concerned about making the software work for the current exhibition and leave the problem of longevity up to the museum; in science, researchers are mostly concerned about making the software work for the current experiment and leave the problem of longevity up to the next researcher. In both cases there is a need to make preservation part of day-to-day practice (archiving by design).

6.3.3 The role of galleries, libraries, archives and museums in software preservation

Ben Companjen, Euan Cochrane, Mustafa Doğan, Scott Kirycki, Pamela Nye

6.3.3.1 Sustainability

Sustainability is still an issue, there was a Humanities Data Centre[42] in Germany which ended, it was unclear what happened after.

6.3.3.2 Questions

What software should be preserved and why? At what granularity? What counts as software? What about software which contains other software? How far do you go

40 https://archive.softwareheritage.org/swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa;origin=https://github.com/chrislgarry/Apollo-11;visit=swh:1:snp:206c27c0c031c6aac6b5fedddba8fe082dea9836;anchor=swh:1:rev:3913f198f4383d4d638c0485d6aa902ff2f35828;path=/Luminary099/BURN_BABY_BURN--MASTER_IGNITION_ROUTINE.agc

41 https://www.softwareheritage.org/swhap/

42 https://humanities-data-centre.de/

down to preserve? Does all software need to be preserved, when is it ok to not keep something? These are important questions when thinking about what role galleries, libraries, archives, and museums (GLAM) should play in software preservation. In addition, it's important to look at how existing policies around other artefacts map to software.

6.3.3.3 Analogies

The use of analogies and metaphors similar to the language used for preservation of physical artefacts can help us to understand the way that digital objects age, for example digital patinas[43] can give the same impression of ageing of digital objects as physical patinas (e.g., oxidation on an old metal object). The skeuomorph use of terminology from the physical to the digital may not be consistently applied but still useful e.g., the look of a physical object vs the impression of a sound of using a digital object/device.

6.3.3.4 Executability and emulation

The GLAM sector should be pretty capable of storing and preserving bits, but the 'playability' or files and older software is not generally something GLAMs can do. There are some examples of 'playability that exist, e.g., beyond preserving digital objects there is a move to experience them as they were originally intended in their original software environments e.g., as in the Universal Virtual Interactor (UVI[44]) detailed by Euan Cochrane[45] of the Digital Preservation Coalition[46]. There is an archives of New Zealand article on opening objects in different software which is part of this wider report on rendering[47] also from the archives of New Zealand. 'Executability' is something that GLAMs might be encouraged to support. Directing students to begin work on these types of projects to generate interest in the field as well as building out their skill sets.

The Emulation as a Service Infrastructure (EaaSI[48]) is an enabling technology which aims to make the running of preserved software much easier; with pre-configured software coupled with emulated hardware available and the ability to document, install, configure, and share software on emulated computers for those who want to publish an emulated resource. There is also support for exporting disk images and packages containing all the dependencies to allow preservation in local systems. A friendly support forum[49] is also available as well as a sandbox[50], blog[51] and other resources[52]. It's interesting to note that EaaSI evolved from EaaS[53] and EaaSI works with

[43] https://www.dpconline.org/blog/wdpd/the-emergence-of-digital-patinas
[44] https://www.dpconline.org/blog/wdpd/designing-a-uvi-for-digital-objects
[45] https://www.dpconline.org/component/comprofiler/userprofile/1363-ecochrane?tab=10
[46] https://www.dpconline.org/
[47] https://web.archive.org/web/20130207025446/http://archives.govt.nz/resources/information-management-research/rendering-matters-report-results-research-digital-object-r
[48] https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/
[49] https://forum.eaasi.cloud/
[50] https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/sandbox/
[51] https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/news/
[52] https://www.softwarepreservationnetwork.org/emulation-as-a-service-infrastructure/resources/
[53] https://web.archive.org/web/20200518112558/http://eaas.uni-freiburg.de/

openSLX[54] which also evolved from that project. EaaSI sustainability plan depends on the uptake of their services, if enough institutions join them there will always be an interest in keeping the EaaSI up-to-date. Licensing is an important topic when it comes to preservation, EaaSI uses the Code of Best Practices in Fair Use for Software Preservation[55]; it should be noted that most parts of the world don't have fair use.

6.3.3.5 Other tooling

The uniform API access afforded by the Preservation Quality Toolkit (PresQT[56]) will allow easier integration of different repository systems. Other resources exist which could be of use to the GLAM sector such as the Wikidata for digital preservation[57] which holds technical and descriptive metadata on software and file formats. The Open Preservation Foundation (OPF) runs several events[58] which are relevant to tooling in this space. Other tools such as ReproZip[59] that create a self-contained bundle of resources (software, data files, libraries, environment variables and options) can also aid the GLAM sector.

6.3.3.6 Future steps

Research data and software preservation are not the same; additional skills and investment are needed in GLAMs to improve software preservation. There is a need for institutions and efforts to come together (e.g., in projects like EaaSI) as individual institutions cannot provide everything needed to access old software and data. Community is also very important and the GLAM sector engaging with meetings such as the ACM reproducibility meetings[60] will be mutually beneficial to tool creators and users in the GLAM software preservation space.

6.3.4 Intellectual property and software preservation; areas needing careful navigation.

Adam Jackson, Jean-Noël Grad, Neil Chue Hong

In principle, version-controlled codebases are a rich piece of heritage to archive; not only do they contain the code, but also a history of changes and related metadata. However, one may not simply be able to look at the code licence to determine the legal position of the repository. Codebases can easily become contaminated with, for example:

- Accidental use of copyrighted material in a GPL project
- Use of GPL code samples in BSD licensed code
- Sample code from StackOverflow that is licensed under Creative Commons.

[54] https://web.archive.org/web/20201029171201/https://openslx.org/
[55] https://www.arl.org/resources/code-of-best-practices-in-fair-use-for-software-preservation/
[56] https://presqt.crc.nd.edu/
[57] https://wikidp.org/about
[58] https://openpreservation.org/events/
[59] https://www.reprozip.org/
[60] https://reproducibility.acm.org/blog/

The legal compatibility of doing this is often overlooked. In practice, heritage is lost by codebase 'cleaning' practices (e.g., copying a snapshot into a new folder and using 'git init'[61] to make a fresh start.)

These incidents aside, there may still be a tension between preserving the history of the software and complying with licences. It might not be possible to compile (or even understand) the software without a dependency that cannot be preserved due to different licensing terms. Such a dependency could be:

- An external library
- A commercial compiler depending on a defunct licence server
- Use of 'compiler extensions' that are not compliant with the programming language standard
- External datasets e.g., for:
    - Training a neural network or an unsupervised learning algorithm.
    - plagiarism-detection software (the reference material is copyrighted and accessing many papers from a publisher requires their approval and comes with bandwidth limitations).
    - scripts that analyse medical data (patient data is confidential and cannot be archived, although synthetic data can be archived).

There are also grey areas around data, databases, and the formatting of data. Ironically, large/complete datasets may be safer to preserve as creative selectivity would help them qualify for copyright.

Intellectual Property law varies from country to country, and even though there are attempts at interoperability, there may be differences that cause issues with software. An example of this is the international variability in "panorama freedom": laws around copyright of picture assets taken in public areas in different countries.

There are legal discussions going on to enable museums and archives to hold copyrighted works for preservation purposes, but it is unclear whether this would include Zenodo[62].

- Implication that software is uploaded by the author, rather than someone seeking to preserve (e.g., if the original author dies)
- Are repositories like Zenodo which are principally publishers, not archives, able to claim protection under exemptions for archives?

Hard to keep track of permissions when people contribute code. Sometimes old code is not in a version controlled environment, in which case authorship information can be hard to find (e.g., names in source file header comments or in function documentation). During pair programming, two individuals contribute code but only one appears as the author in the version control commit (in git it's common to write "Co-authored-by: name <email>" in the commit message to acknowledge co-authors, and it's recognised by GitHub and GitLab).

[61] https://git-scm.com/docs/git-init

[62] https://about.zenodo.org/

Contributor licence agreements (CLAs[63]) cause barriers to new contributions.

While platforms such as Github have made it easy to add a standard licence file, and resources such as choosealicense.com[64] are available to help with selection, there are still many repositories without an explicit licence. If no licence[65] is provided with a software repository, the default status is no permission to use, modify or share. In this scenario it is difficult to determine what permission is needed or whose copyright claim applies, especially when third-party contributions and project forks are involved. Additional documentation can clear this up.

Some packages have strange complicated/restrictive licences that make things worse from a general software sustainability perspective (e.g., they cannot be used by some institutes!). However, this may not be a *heritage* issue if archiving services have a suitable legal exemption - if you are exempt from copyright, you should be able to ignore a lot of other licence terms.

There may, however, be a heritage issue around contaminated codebases in cleaning up a codebase and making it suitable for distribution, important heritage aspects around the history of the project may be lost before preservation.

7 Open Science & applying the FAIR principles to software

This section describes the second session of the workshop. This session includes featured sections (author enhanced and expanded text provided after the workshop) on the progress of FAIR in mathematics and the EU OpenDreamKit project, FAIR adoption in the Australian Research Data Commons, and FAIR computational workflows. The summaries submitted for the workshop cover the evaluation of a FAIR tool for Earth system models, ELIXIR software management plans and fiscal sponsors in open source.

The discussion session which followed covered measures of FAIRness for software, the importance of FAIR software for those working in research and starting an open source endeavour.

7.1 Featured

7.1.1 glimpse at decades of FAIR struggles and practices in computational mathematics

Nicolas M. Thiéry[66]

7.1.1.1 Executive summary

In the last decades, far before their formalisation, Open Science in general and the FAIR principles in particular have been in effect at the core of the development of Free Software for Computational Mathematics. Despite constant challenges and struggles,

[63] https://en.wikipedia.org/wiki/Contributor_License_Agreement

[64] https://choosealicense.com/

[65] https://choosealicense.com/no-permission/

[66] Links to a video & transcript available at - https://wosss.org/wosss21/S2-NicolasThiery

the situation has been continuously improving, notably through the emergence and propagation of best practices. The recent advances of Open Science, and in particular the recognition of its importance by institutions and policy makers is a major step forward. In this section, we will illustrate these elements through the lenses of the development of computational mathematics software like SageMath.

7.1.1.2 Messages for policy makers

- Given appropriate means, scientists are in general sympathetic to Open Science, when not enthusiasts. In addition, the appropriate best practices vary very much from one domain to the other. **Support and foster FAIR practices. However, don't impose them unless absolutely necessary to counterbalance other higher forces**.
- The FAIR ideas have been around for decades for software. Software raises very specific FAIR challenges; however, it's not just another type of data. **Support FAIR research software as one of the pillars of modern science.**
- Public bodies ought to fund basic scientific software development, and in particular **Fund long term software maintenance, at all scales.**
- Research software development for-users-by-users can work well; however, support from Research Software Engineers makes a huge difference to teach the community base, provide advice and consulting, and achieve highly technical tasks. **Ease access to Research Software Engineers, at all scales.**

7.1.1.3 The story

Computing has always been one of the favourite tools in (pure) mathematics to discover and explore new theories. Thus, as computing devices emerged, they were naturally adopted to compute examples, test conjectures, or even prove theorems, like the classic four colour theorem: computers became the telescope of mathematicians. A telescope made of both general purpose hardware and bespoke software.

At first, it took advanced skills to develop that software, but the scale was limited: for each project a dedicated program would typically be written by one or two persons. Starting from the 70's, with computing capacities and computational mathematics blossoming, the range and the depth of mathematics that could be explored with the computer increased drastically. There was a price however: the software complexity and scale also increased drastically. Rewriting software for each occasion was not sustainable anymore; it had to be **Reused**.

Many development models were explored in the 80's and 90's to achieve that aim, with two main archetypes emerging: in the first archetype -- developed by users for users -- a mathematical community would get together and build a common system aggregating and structuring the development efforts of the community. Examples of such systems include, for example, GAP for Group Theory, PARI for Number Theory, Macaulay for Commutative Algebra, etc. Naturally these systems adopted -- sometimes before they were formalised -- the four principles of Free Software. Indeed, computer exploration is by nature a handicraft where the needs are ever changing. Hence the user needs a toolbox that they not only can reuse as is, but observe with a critical eye, adapt to their own hand and job, with the ability to redistribute their

adaptations. A major strength of the by-users-for-users development model is that codesign is at its root, ensuring that the software meets the user's needs and letting the user craft the tool to their own hand.

In the other archetype, the development was carried out by a dedicated team, the obvious challenge being to fund such a team in the long run. Thus, such systems usually ended up taking the commercial route, hence targeting by necessity a wide audience susceptible to draw enough revenue. These systems, including e.g., Maple, MuPAD, or Mathematica, had a major impact by putting computational mathematics at the fingertips of casual researchers, engineers, teachers, and students. At least when they could afford the licences.

At this point, it should really be emphasised that this text is no more than a glimpse into decades of work by hundreds. A proper history should highlight dozens of other systems of all scales and mixed development models that have supported mathematics over the years. Not counting that mechanising mathematics goes far beyond computation: formal proofs, databases, knowledge management, typesetting, etc.

At the turn of the century there was a growing frustration in the community about the situation: through web searches, conferences, or hearsay you would **Find** many functionality that you would dream to use in your own computations; however, more often than not, they were not **Accessible**, either because they were provided by a system with a licence that you could not afford or that would not run on your computer, or by a bespoke system that was not **Interoperable** with yours. To resolve that tension, many were dreaming of a system that would be simultaneously Free Software and general purpose.
This was a major technical and social challenge given the very limited resources that the community could devote to such an endeavour.

There was hope however thanks to the emergence of:

- Adequate general purpose programming language: up to now most systems had developed their own language to serve the needs of mathematical programming
- A large ecosystem of specialised free mathematical software
- Tools and practices enabling large scale collaboration on free software
- A crowd of open source enthusiasts among potential users

Finally, SageMath -- based on Python -- was started in 2005, and progressively a community of hundreds of developers crystalized around that project, proving the sustainability of the by-users-for-users development model if one **Reuses** whatever can be to focus the energy on the core of the project. Later steps included a tight cooperation with the Jupyter community to outsource the development of the user interface.

To promote **Findability**, the community invested a lot of energy in training workshops, notably dedicated ones for women and minorities, and Question and Answers tools (mailing lists, Ask Sagemath[67]). At a lower scale, interactive use and introspection are

[67] https://ask.sagemath.org/questions/

powerful tools for discovering features, especially when supported by a strong type system that closely models the business objects, and by tutorials and systematic documentation with many examples. That documentation could still be considerably enhanced with a strong network of cross-links; Natural Language Processing might be able to come to the rescue to automatically generate such a network.

Accessibility has been a continuous challenge due to the scale of SageMath, with hundreds of dependencies, some dating from decades: making it easy to install SageMath on personal computers or computing infrastructure required porting to the main operating systems, modularizing, promoting loose coupling between components, and standardising the build systems to help packaging (Debian, conda, pip, ...) of SageMath itself and users' extensions. These efforts and the integration in the Jupyter ecosystem have considerably reduced the entry barrier for users, in research, teaching and engineering, notably through collaborative virtual environments provided by services such as CoCalc[68] or JupyterHub[69]. This in particular supports **basic reproducibility**, by letting users make their computational narratives accessible to anyone online through on-demand virtual environments hosted by services such as MyBinder[70].

Interoperability is at the core of a system with so many dependencies. The challenge comes from the diversity and richness of objects that one wants to manipulate in mathematics and rich APIs (thousands of types of objects each with dozens of methods). Work occurs at many levels:

- Low-level procedure calls, and data handles across components often written in different languages, ideally in shared memory for performance. Favourite tools include Cython, pythran, cppyy, ... Recent languages like Julia often offer helpful facilities in that regard, and the OSCAR Computer Algebra System[71] explores how to exploit them for a tight integration between systems
- Adapters, to let objects in a used component behave as native objects of the calling component
- Data conversion

On the pragmatic side, adaptation and data conversion can be achieved on a case by case basis between two systems, though this does not scale well. An ongoing research endeavour is instead to build common ontologies and adapt / convert between any two components through these common ontologies. For that specific aspect, the FAIR challenges for data and software are intimately related.

7.1.1.4 Conclusions

Like in many other areas, the development of large scale computational mathematics systems in the last decades has been strongly correlated with the advancement of FAIR best practices. These practices enabled a sustainable for-users-by-users development model which best meets the user's needs in research, engineering, and education.

[68] https://cocalc.com/
[69] https://jupyter.org/hub
[70] https://mybinder.org/
[71] https://oscar.computeralgebra.de/

A major trend supported by the FAIR principles is the evolution from an array of competing software to a collaborative ecosystem of software. At a low granularity, such ecosystems offer a fertile ground for innovation, fostering individual ideas and features to sparkle, live and compete; and die when superseded. Death for software artefacts is a necessity: otherwise, technical debt takes over; it's also not so bad thanks to **archival**. Meanwhile, collaboration is the key to innovation at the level of systems, people, and communities.

Implementing the FAIR best practices can involve highly technical long term investments stretching the limits of the by-users development model. This motivated the OpenDreamKit[72] Horizon 2020[73] European Research Infrastructure project (2015-2019; €7.6M) to support the computational maths community. We were pleased to see that institutions and funding bodies nowadays start to appreciate the importance of FAIR principles and of research software. This project was the occasion to confirm the high impact that a few **Research Software Engineers** can have. It was however also the occasion to meet the limitations of project-based and novelty-based funding schemes: how to fund **long term software maintenance**? How to offer **career paths** for the required highly skilled professionals? Inadequate granularity and high management overhead are also impediments to their efficiency.

7.1.2 FAIR adoption

Tom Honeyman[74]

The Australian Research Data Commons (ARDC) is a national facility supported by the federally funded national collaborative research infrastructure strategy (NCRIS) scheme. In this role the ARDC is a provider or co-investor in several areas of digital infrastructure, skills development, and guidance, and has national programs in storage, compute, data, informatics services, skilled workforce, policy, platforms and software.

In seeking to adopt the FAIR for Research Software (FAIR4RS) principles, the ARDC is considering action across several of these programs.

Within platforms that include JupyterHub the ARDC is looking for opportunities to incorporate features or guidance relating to the principles. We are developing an easily deployable JupyterHub-based platform as a service to be deployed within the national ARDC Nectar research cloud. We see this as an opportunity to put the guidance and encouragement to adopt the principles and other best practices relating to research software authorship close to the authors themselves. Similarly, we are encouraging the same with our platform's co-investment projects that also incorporate JupyterHub.

Within our own in-house software development, we will make that software FAIR. We will leverage this as an example of best practice to show partner organisations. Within our co-investment programs we'll be looking to work with project partners who are

[72] https://cordis.europa.eu/project/id/676541

[73] https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-2020_en

[74] Links to a video, slides & transcript available at - https://wosss.org/wosss21/S2-TomHoneyman

developing research software tools to assist them in applying the principles to the software they are developing.

Within our skilled workforce program, we are looking at incorporating FAIR4RS into our training materials, and other guidance materials (used broadly throughout Australia). Based on the ARDC "FAIR data self assessment tool", we are developing a similar tool to aid socialisation of what adopting the principles might look like at different levels. This tool will not be targeting software authors themselves, but rather support staff, managers, policy makers and other roles that impact upon the authors of research software.

Consistent with our existing policy advocacy work with relevant national bodies and research organisations, we will continue to advocate for FAIR outputs (particularly within our national signatory obligations under the OECD recommendations concerning data from publicly funded sources), but direct interested parties to the FAIR4RS principles when considering what actions might apply to research software. That is, we will advocate for FAIR data and software.

Our own policy regarding outputs from co-investments will be updated to clarify expectations for future co-investments. Specifically, where reasonable, they should be applying the FAIR4RS principles to software outputs arising from co-investment, instead of interpreting and applying the original FAIR principles to all outputs.

Finally, under the software program we will socialise and further assist in adoption of the FAIR4RS principles amongst the national software authoring communities that we support or facilitate.

7.1.3 FAIR Computational Workflows

Carole Goble[75]

The FAIR principles (Findable, Accessible, Interoperable, Reusable) [6] have laid a foundation for sharing and publishing digital assets, starting with data and now extending to all digital objects including software [7].

Computational workflows are a special kind of software for handling multi-step, multi-code data pipelines, analysis, and simulations. Their use has accelerated in the past few years driven by the need for repetitive and scalable data processing, access to and exchange of processing know-how, and the desire for more reproducible (or at least transparent) and quality assured processing methods [8]. COVID-19 pandemic has highlighted the value of workflows [9].

Computational workflows encode the methods by which the scientific process is conducted and via which data are created, by capturing precise descriptions of the multiple execution steps and data dependencies needed. Workflow Management Systems and execution platforms handle the definition and set-up of the multi-step specification and the heavy lifting of dependency management, code execution, data,

[75] Links to a video, slides & transcript available at - https://wosss.org/wosss21/S2-CaroleGoble

and control flow, reporting and monitoring. Over 300 workflow systems are currently available[76], although a much smaller number are widely adopted [10].

As first class, publishable research objects, it seems natural to apply FAIR principles to workflows [11]. The FAIR data principles themselves originate from a desire to support automated data processing, by emphasising machine accessibility of data and metadata. Workflows are special kinds of software, but they're also a precise description of a process. As workflows are digital objects that have a dual role as software and explicit method description, their FAIR properties draw from both data [6] and software principles [1,7].

Workflows create unique challenges such as representing a complex lifecycle from specification to execution via a workflow system, through to the data created at the completion of the workflow. As workflows are chiefly concerned with the processing and creation of data they have an important role to play in ensuring and supporting data FAIRification.

7.1.3.1 Properties of workflows that impact FAIR

Although workflows are inherently software, workflow management systems have additional properties that impact how we might apply FAIR principles.

- **Method abstraction.** Workflows have a specification, which is a description of the steps with parameters and inputs and guidance - offering FAIR transparency, and metadata descriptions. We can consider these almost to be like FAIR data because they're descriptive artefacts. On the other hand, we have software, including workflow management systems themselves, as well as the tools and the infrastructure that the individual codes that they're orchestrating and chaining together. Like all software this is related to reproducibility; of being able to run those pipelines and reuse those pipelines. The descriptions refer to method preservation, but software reproducibility is more about software preservation. Alongside these two perspectives of method abstraction and the software that implements the method are the associated objects around workflows: logs of their execution, example data, test data, and services associated with them in order to be able to check whether these workflows are FAIR.

- **Method modularization and composability.** Workflows expect to take various different components in different languages from different third parties and be able to put them together and port them and then to recombine them and port them to yet more hosts. Workflows are compositions of components, including other workflows that can be broken down, versioned, recycled and so on. This requires FAIR to apply at the different levels of abstraction at the description level and the software level, and for the different components that make up the workflows. There are multiple workflow systems in the landscape, which typically are used in an intertwined kind of way. People use a workflow management system, which is a dedicated infrastructure that does that neat

[76] https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems

separation of concerns with respect to modularization, but also abstraction, and execution. But these are typically used also with interactive notebooks, and scripting environments, which perhaps are less clean in this regard, but still running multiple steps.

7.1.3.2 Steps towards FAIR Workflows

The work on defining and improving the FAIRness of workflows has already started. An ecosystem of tools, guidelines and best practices are under development to reduce the time needed to adapt, reuse and extend existing scientific workflows.

The EOSC-Life[77] Project is the European Open Science Cloud Life Sciences cluster, which brings together European Research infrastructures dedicated to building a collaborative space for digital biology. EOSC-Life is building a FAIR data and workflow Commons, not unlike what Tom was talking about earlier about the Bio Commons, and other Commons in Australia. The Life Science infrastructures extensively use computational workflows for preparing, analysing, and increasingly sharing large volumes of data. They have very different kinds of workflow management systems but are all effectively building multi step pipelines and multi step processes to coordinate and execute multiple codes, codes that they may not have developed themselves. Those workflow systems are handling data and processing dependencies and doing other kinds of heavy lifting, typically data pipelines. Alongside the many workflow management systems in use there are dedicated registers and repositories that work with dedicated workflow services. The EOSC-Life Commons must honour this diversity and legacy.

7.1.3.3 Machine processable metadata

A fundamental tenet of FAIR is the universal availability of machine processable metadata.
EOSC-Life has developed a metadata framework for FAIR workflows based on Schema.org[78], RO-Crate[79] [12] and Common Workflow Language (CWL[80]) [13], EOSC-Life have made great efforts to on-board community workflow platforms such as Galaxy[81], snakemake[82], nextflow[83] and CWL to carry and use FAIR metadata for discovery and reuse. Auto harvesting FAIR metadata from different workflow management systems means their onboarding is essential. We use this metadata framework to move workflows around FAIR workflow systems, services and registries. The workflow metadata framework covers:

- Canonical description and common metadata about what the workflows are about using Bioschemas, Common Workflow Language and EDAM. CWL provides a canonical workflow description of the steps of the workflow. The Bioschemas Computational Workflows[84] provides a schema.org metadata

[77] https://www.eosc-life.eu/
[78] https://schema.org/
[79] https://www.researchobject.org/ro-crate/
[80] https://www.commonwl.org/
[81] https://galaxyproject.org/
[82] https://snakemake.readthedocs.io/en/stable/
[83] https://www.nextflow.io/
[84] https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE/

profile about a workflow. The EDAM ontology[85] types the inputs and outputs of the steps of the workflow.

- **Packaging all of the different disparate components** around workflows using RO-Crates [12], called a Workflow-RO-Crate[86] profile. RO-Crate is an implementation of FAIR Digital Objects[87], a key concept in the European Open Science Cloud (EOSC) and the EOSC Interoperability Framework. Packaging the logging and data lineage results of running a particular workflow is to be captured in a further specialised RO-Crate: Workflow-Run-RO-Crate [88] . Workflows produce superfluous noise and detecting signals is hard. A great deal of workflow provenance is fine grained metadata about execution that is easy to collect but actually not useful. Distilling the lineage of critical data products and cleaning out the rest is much harder. The T7 ProvWeek 2021[89] highlighted the discrepancy between workflow provenance [14] and transparency.

EOSC-Life has also developed services to support FAIR workflows using the metadata framework to exchange workflow objects between services, workflow management systems, registries, and repositories.

- **Findability**: The WorkflowHub [90] , a registry of workflows, links into and leverages different workflow management systems, their different deployments and their different repositories. The Hub has a focus on rich metadata using the framework that supports findability and reuse requirements, for example, licensing and provenance of the workflows. Facilities support Workflow RO-Crates, registration processes from Git, curated libraries of workflows, lifecycle support around versioning with git support, and communities of practice curating and sharing workflows.
- **Accessibility:** Once found, workflows need to be accessed - WorkflowHub uses the GA4GH TRS[91] API for a standardised communication protocol to launch workflow executions. As the FAIR principles for decree that metadata are accessible even when the workflow is no longer available, the metadata framework, when completed, yields enough metadata that a workflow is read-reproducible as a method description even if it no longer runs. The RO-Crate packaging means that every workflow in WorkflowHub can be deposited in other long term repositories like Zenodo. The software might not exist, but the description still will.
- **Interoperability**: This principle is the hardest to unpack for both data and software. For workflows, interoperability follows two threads: (i) supporting workflow system interoperability through workflow descriptions independent of the underlying system (e.g., Common Workflow Language and WDL[92]) and (ii) workflow component composability. Workflows are ideally composed of

[85] http://edamontology.org/page
[86] https://about.workflowhub.eu/Workflow-RO-Crate/
[87] https://fairdo.org
[88] https://www.researchobject.org/workflow-run-crate/
[89] https://iitdbgroup.github.io/ProvenanceWeek2021/t7.html
[90] https://workflowhub.eu/
[91] https://ga4gh.github.io/tool-registry-service-schemas/
[92] https://openwdl.org/

modular building blocks like Lego, and these and the workflows themselves are expected to be reused, refactored, recycled and remixed. Thus, FAIR applies "all the way down": at the specification and execution level, and for the whole workflow and each of its components.

- **Reusability**: Composability relates to reuse – that is, adapting [7], a workflow or its component “can be understood, modified, built upon or incorporated into other workflows”. Reuse challenges also include being able to capture and then move workflow components, dependencies, and application environments in such a way as not to affect the resulting execution of the workflow. Independent components operate through API's and metadata standards requiring programmatic access to the metadata of those particular tools, the making of workflow ready tools [15] and being able to create canonical and recyclable workflow blocks. Workflow developers can be both data-FAIR, by using and making identifiers, licensing data outputs, tracking data provenance and so on, and workflow-FAIR by managing versions, providing test data, and sharing libraries of composable and reusable workflow “blocks” [16]. Communities such as BioBB and nf-core are working on reviewing, validating, and certifying canonical workflows. Interoperability and reusability present important obligations on software developers to ensure that tools and datasets have clean I/O programmatic interfaces, no usage restrictions, use of community data standards and identifiers, and that they are simple to install and designed for portability. The components need to be designed as FAIR units that can be FAIR unit tested, FAIR data production and as FAIR workflows for reuse in other workflows.
- **Usability**: As FAIR software needs to be usable and not just reusable; EOSC-Life has also developed services for, e.g., workflow testing (LifeMonitor[93]), execution and benchmarking, using the metadata framework to exchange workflow objects. Packaging using containers, execution standards and API's such as the GA4GH standard for running workflows all feeds into usability, as does dependency management and FAIR unit testing of workflow components.

7.1.3.4 Workflows as functions for FAIR data

As workflows are instruments of data generation, typically dealing with data flow, they should be supporting FAIR data. Thus, we need to test that workflows actually produce FAIR data. Are they licensing data outputs? Do they use community data formats? What usage restrictions do they require? Do they handle identifiers correctly, which is critical for the detailed provenance of the data that goes through those workflows? Again, good design for FAIR data, and reuse and the development of canonical workflows and libraries of validated and curated workflows by communities, as well as best practice and golden examples of workflows, which go through a reviewing process. That really requires training and stewardship and sustainability activities.

7.1.3.5 Challenges

Many challenges remain for describing, annotating, and exposing scientific workflows so that they can be found, understood, and reused by other scientists. Further work is required to understand use cases for reuse and enable reuse in the same or different

[93] https://crs4.github.io/life_monitor/

environments. The FAIR principles for workflows need to be community-agreed before metrics can be considered to determine whether a workflow is FAIR, whether a workflow repository or registry is FAIR, and whether it is possible to automatically review whether a workflow's dataflow is FAIR.

Ecosystem citizenship means that "FAIR takes a village" [17]. Workflows have a community of practice as opposed to all of the software community. Community activism, led by the platforms and registries coming together in a community group like the Workflow Community Initiative[94], is needed to define principles, policies, and best practices for FAIR workflows and to standardise metadata representation and collection processes. Communities of workflow developers are building well curated and canonical workflows that we can address directly in order to be able to improve their practices. There are those working in building the standards and communities for building sustainability and policy around FAIR workflows in our FAIR Commons.

As software developers there are many different challenges to deal with, with FAIR workflows. We still have to define the principles, particularly considering their complex lifecycle of specification and execution and data products, and metrics around the FAIRness of workflows. We need to include the folks that develop the codes that are incorporated into workflows, to code to become workflow friendly, with clean interfaces, avoiding usage restrictions and so on, as well as FAIR workflow making. We need to work on how we can automate the FAIRness in workflows and check the way that those workflows have been developed, so that they adhere to FAIR principles, not just for the workflow, but also for the data that flows through them. It is important for workflow platforms to enable FAIR outputs, like citing correctly the input data and the used software The FAIR Digital Object approach - using RO-Crate and a metadata framework - means we can package links to data and cleanly reference the used software too. When we have ubiquitous PIDs we can begin to build citation metrics too.

We should not forget the FAIR workflow user. We want to encourage people to use well documented, FAIR enabling and FAIR workflows and to credit the makers of them because this is a non-trivial and expensive activity. I really liked the term that was used before, on.

7.2 Summaries

7.2.1 Research software and beyond - ESMValTool: a community and FAIR software for evaluations of Earth system models

Fakhereh (Sarah) Alidoost[95]

Let's imagine I want to analyse a time series of 50 years of air temperature in the past and in the future generated by 10 climate models and visualise the results. So, I developed some lines of code implementing the analysis. The code includes running several tasks: finding and downloading data, checking the data for completeness and correctness, processing the data, and finally storing the results. Also, it creates some

[94] https://workflows.community/about

[95] Links to a video & transcript available at - https://wosss.org/wosss21/S2-FakherehAlidoost

plots that can be used for figures in my publications. It took me some time and effort to develop the code that performs the analysis in an efficient way concerning computational costs. You need to implement a similar analysis in your research, for example analysing air temperature and precipitation simulated by 5 other different models. You can re-use my code instead of re-implementing it from scratch because it is FAIR (Findable, Accessible, Interoperable and Reusable). This is the world where our research is efficient, and our code is sustained as it is usable by others for their own research.

Generally speaking, the first step in using software is to find it. Then, we need to know how to access it. Finally, we can execute the software. However, implementing these steps is not straightforward since the research software is made to run a specific experiment in a scientific domain. Also, developing software can be a challenging task because as researchers, we might not have all the skills needed to write well-described and well-structured code during our research. Therefore, it is impossible to define one solution that fits everyone in all research disciplines. To facilitate this, research communities promote best practices and recommendations around elements related to software. These elements are mainly public repositories, version control systems, licences, community registries, and software quality tools. As implementing FAIR principles changes the way we do science, guidelines on how to treat those elements are often tailored towards our own research field/community. It is also common that best practices of developing software cover other aspects of the research like data and papers.

In my talk, I will introduce a community-driven software: Earth System Model Evaluation Tool (ESMValTool[96]). I will also explain how this software makes it possible to reuse code easily and ensures transparency and reproducibility of research output. Here is a summary:

The World Climate Research Programme (WCRP) provides a platform for international collaborations to better understand climate phenomena and develop useful climate information. The core projects of WCPR explore models that show how our climate system works, how it changes, and what impacts are. Climate and Earth system models are very complex codes that project future climate. Outputs of the models are used as the basis for climate research around the world. To analyse the output collectively, WCRP organises and leads the activities in the Coupled Model Intercomparison Project (CMIP) involving thousands of researchers. CMIP sets standards and experimental protocols and facilitates sharing codes and comparing models' results. ESMValTool is a software that facilitates the assessments of Earth system models in CMIP. The software is built and maintained by a community of scientists and software engineers. The community includes technical and scientific teams that review contributions mainly in the form of codes in climate-related domains. Also, the community is supported by principal investigators and a user engagement team. Discussions, developments, maintenance, and collaborations mostly take place in public on GitHub[97]. The software processes data and runs analyses efficiently regarding computational resources. It also stores provenance and citation information in a user-friendly way. Automated testing through unit tests and review processes

[96] https://www.esmvaltool.org

[97] https://github.com/esmvalgroup

safeguards the quality of research data, codes and publications. In addition, the software is a collection of publicly available scripts with extensive documentation. Moreover, there is an online tutorial that shows not only how to run an experiment but also how to develop your own scripts. In this way, ESMValTool helps others to understand our analyses, makes the results reproducible and facilitates collaborations.

7.2.2 Developing the ELIXIR Software Management Plan for Life Sciences

Fotis Psomopoulos[98]

Data Management Plans (DMPs) are a key element of good data management and are now considered a key element of Open Science practices. A DMP describes the data management life cycle for the data to be collected, processed and/or generated within the lifetime of a particular project or activity. Conversely, a Software Management Plan (SMP) can help to formalise a set of structures and goals that ensure your software is accessible and reusable in the short, medium, and long term. Although it has a management perspective, the main advantage of an SMP is that it provides clear context to the software that is being developed. In that sense, it addresses several aspects of the software development process such as (a) supporting reproducibility and reusability of the software, (b) allowing funding agencies to have a better grasp of the envisioned development process (as well as the achieved milestones), (c) increasing the awareness of the existing community standards that can/should be used, and (d) ensuring that the software can be easily accessed by the wider community.

There are several flavours of SMPs already available in one form or another. The Software Sustainability Institute (SSI) offers a very detailed checklist[99] that is further complemented by an online sustainability evaluation service (SES [100]). Several journals (such as SoftwareX and the Journal of Open Source Software) have checklists that are expected to be filled in by the software authors before any submissions addressing most of the points of an SMP. Finally, there are funding agencies (such as the Wellcome Trust) that expect a research outputs management plan[101] submitted during any application.

A key downside of the SMPs is that they tend to be rather complex, occasionally requiring deep technical knowledge of the software development process. To address these drawbacks, ELIXIR has put together a simplified version of an SMP, tailored for Life Science oriented projects but still general enough to be more widely applicable. The primary goal of the ELIXIR SMP was to encourage wider adoption by Life Science researchers, and be as inclusive as possible to the various levels of technical expertise, while also having an explicit connection to the FAIR principles for Research Software (FAIR4RS WG[102], [18]). A common theme in Life Science researchers is the wide differences in background expertise, with most researchers being self-taught research software developers. Having an SMP with a relatively low barrier in technical knowledge, while maintaining all the best practices expected in research software

[98] Links to a video & transcript available at - https://wosss.org/wosss21/S2-FotisPsomopoulos

[99] https://www.software.ac.uk/software-management-plans

[100] https://www.software.ac.uk/resources/online-sustainability-evaluation

[101] https://wellcome.org/grant-funding/guidance/how-complete-outputs-management-plan

[102] https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg

development, may both encourage wider adoption of these practices as well as increase the awareness of the multiple aspects involved in research software development.

The starting point for the creation of the SMP was the four recommendations for Open Source Software [19]. These recommendations are meant to encourage best practices in research software development. To incorporate the feedback of the ELIXIR community and the best practices used there, interviews were conducted during a dedicated project within the Europe BioHackathon 2019. The common practices and ideas collected from the interviews were structured around the four FAIR Research Software principles (Findability, Accessibility, Interoperability and Reusability), based on the area they were more relevant. Additional care was given towards ensuring that the main low-effort high-gain actions were captured, to ensure that the resulting questions had a low knowledge barrier for the expected users. The first version of the SMP, grouping the practices as described above, was presented in a dedicated webinar organised by ELIXIR in July 2020. Following that, two iterations of revision by the ELIXIR community were conducted. Specifically, developers from the ELIXIR community were invited to review and comment on the questions and corresponding options, ultimately leading to a consensus version of the SMP that incorporated all proposed changes. Currently available as a survey, future plans of the ELIXIR SMP include a human- and machine-readable version, that can be automatically queried and connected to relevant tools and metrics within the ELIXIR Tools ecosystem and, hopefully, beyond.

7.2.3 The Role of Fiscal Sponsorship in Open Software

Andy Terrel[103]

Open source software is very often a work of passion to scratch the author's itch. What happens when that passion goes stale? Just as a pair of young lovers, the software community must evolve its relationship, i.e., someone has to do the dishes.

While early open source projects could organise around a few people with not many resources, today's projects often include hundreds of contributors working for numerous companies. Additionally, the people of the project are spread across dozens of countries complicating many legal issues. To meet this need of maturing software projects numerous fiscal sponsor organisations have been formed. Fiscal Sponsorship manages the project's financial, legal, and organisational resources. It also allows projects with similar organising principles to come together and produce common goods.

In this presentation, we use NumFOCUS[104] as a case study to show the various aspects of the Fiscal Sponsor. NumFOCUS founded in 2012 has grown to sponsor over fifty scientific software projects with a vibrant educational program to help a growing community learn and build new tools.

[103] Links to a video, slides & transcript available at - https://wosss.org/wosss21/S2-AndyTerrel

[104] https://numfocus.org/

7.3 Discussions

7.3.1 Setting measures of FAIRness for software

Michelle Barker, Raniere Silva, Adam Jackson, Peter Doorn, Tom Honeyman, Nicolas M. Thiéry

When a researcher says “I want to keep my research alive. How do I do it?”, it is important to consider that all digital outputs should be made FAIR, including research software. Who will be responsible for ensuring that research software is FAIR?

Emulation is an interesting approach to keep research software alive in the GLAM (Galleries, Libraries, Archives, and Museums) sector, and for e.g., arcade computer games. It might be suitable for some types of research software, such as prototype tools (demonstrating new ideas, methods, and models) and analysis code (capturing one-off analytical decisions and use of methods), which might need to be “preserved” rather than maintained.

There are existing approaches that measure some aspects of the FAIRness of software, although they have some shortcomings:

- The Journal of Open Source Software (JOSS) is quite successful in using a checklist to establish a baseline over a wide range of research areas. But this is not suitable for *all* types of research software.
- Wellcome Trust is developing a “FAIRware[105]” tool for funders to assess grantee data, but there is not yet an equivalent for software. There are a range of FAIR evaluation tools for measuring data FAIRness, but they are not consistent in how they analyse FAIRness.

With the development of the FAIR for Research Software Principles, there is the potential for future requirements requiring research software to be FAIR. When measuring that FAIRness, there is a danger that the demands are set too high or that they become too complex for researchers to obey, and that may result in less FAIR software instead of more FAIR. It is important to measure things that are meaningful.

Measuring FAIRness shouldn’t be binary because some software is “allowed” to be less FAIR and the FAIR for Research Software Principles framework can be used to have a sensible discussion around these cases.

There is not just one type of “Research Software”, there is an enormous diversity. Not all FAIR criteria make sense for everything, e.g., prototype tools are a valid category, which are not made to demonstrate a new idea, method, or model. Most prototypes emerging from research do not go on to become maintained software.

FAIR principles are “aspirational”, not absolute. It is possible to “measure” (or describe) in what respects software is FAIR, and in which respects not (or less). There are many possible tiers of interoperability, and some fundamental limits to what is possible.

[105] https://fairware.metadatacenter.org/

Funding requirements are useful for driving behaviour change, but funders don't only care about FAIR, other criteria matter as well. So, incentives not to set those requirements too high where it might reduce output of useful software. Communities will have different expectations, so top-down requirements are problematic.

There is a general wariness around boiling things down to a single metric. (Goodhart's Law [106]: metrics cease to be useful when they are targets.) A checklist plus commentary is probably useful to funders in most practical scenarios.

Such checklists are useful to initiate discussions within projects, leaving to projects and communities to decide for each item whether it is impossible/relevant/etc in their context.

There is a danger in being too ambitious with the requirements on the FAIRness of research software. The danger is that it will not be obeyed if it is made too complicated for researchers to comply with. Those who set the FAIRness criteria set by funders (or domains, institutions, etc) should be aware of the possibility that it will not be adhered to. The emphasis should be on having a sensible basic and practical baseline.

7.3.2 Why is knowing about FAIR Software important for researchers, research software engineers, data stewards and others

Ben Companjen, Carina Haupt, Paula Martinez, Rachael Ainsworth, Fotis Psomopoulos, Sarah Alidoost, Carole Goble, Meta Keijzer-de Ruijter, Andrew Sandeman

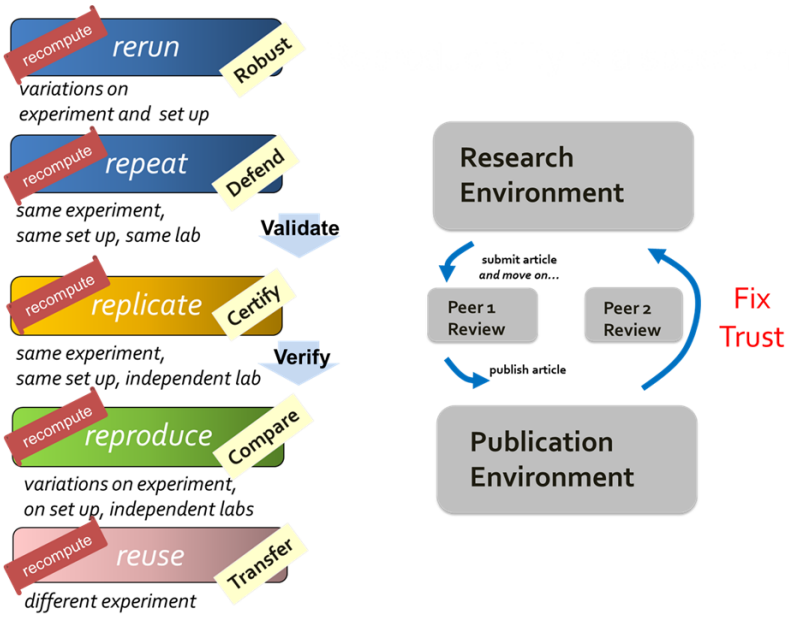
It is often stated that we want reproducible science! However, researchers care more about **transparent** and **reusable** science, and **trusted** science not reproducible science. The IEEE eScience panel on open science raised these points, as does the NASEM NIH workshop [20].

Reproducibility does not resonate with researchers as much because there is less perceived value in publishing reproduction (the focus of journals being on novel science). Researchers usually want to reuse in a different context.

Reproducibility[107] (same data + same methods → same results) is a means to an end, not the end itself. The main goal of reproducibility is to generate **trust**. Reproducibility, transparency, and trust are not defined by FAIRness, but can be facilitated by FAIR. FAIR Software may not lead to reproducibility but can lead to trust.

[106] https://en.wikipedia.org/wiki/Goodhart%27s_law

[107] https://www.nationalacademies.org/our-work/reproducibility-and-replicability-in-science



R*[108]

The intention of FAIR principles was to get a conversation going; it is meant to be a spectrum/journey towards improving things. It was not expected to be policed or to become a cult. Unfortunately, it has become dogmatic, and it is perceived that "75% of FAIR is air". The interpretation of the principles became too abstract.

FAIR principles should provide context on how things should be done, make them part of community norms - not something to be "enforced".

As a community, we need to stop positioning FAIR as labour and burden and stick with benefits outside the researcher and more about the carrot and benefits for them and their scientific workflow, and personal "productivity" - "FAIR is a love letter to yourself[109]".

7.3.2.1 Conclusions:

- Transparency and trust for reviewing works (pre and post publication)
- Reproducibility, transparency, and trust are not defined by FAIRness, but can be facilitated by FAIR
- "75% of FAIR is air"
- What is the value to those who you are trying to get to work in a FAIR way? Focus on benefit, do not add burden - there is a real danger of FAIR fatigue (where you won't be listened to at all) if you become too dogmatic about FAIR.
    - Focus on improving the efficiency of the workflow, conversation on how we are conducting research and how we can do it better, impact
- Unexpected side effects/ramifications on careers, repositories, and institutions that enforcing and policing FAIRness can have

[108] https://www.slideshare.net/carolegoble/what-is-reproducibility-the-r-brouhaha-and-how-research-objects-can-help-236725062
[109] https://www.nature.com/nature-index/news-blog/what-scientists-need-to-know-about-fair-data

- FAIR principles are the start of a conversation about how we can improve research
    - Organise their work
    - Use shared vocabularies
    - Make more efficient workflows (i.e., ways of working)
    - Increasing the impact of work done
    - How to improve trust in results

7.3.3 How to start your own band and the open source analogies

Andy Terrel, Carlos Martinez-Ortiz, Shoaib Sufi

Following Andy Terrel's talk (see 7.6), the analogy of starting a band and starting an open source software project is a very interesting one.

It is very easy to focus on the technical aspects of starting an open source project, such as what type of licence you should use. There are great resources available for educating people in how to pick a licence, such as Choose a License[110] and TLDRLegal[111].

However, it is important to keep in mind that software (just like music bands) has a social aspect. It is important to consider what is the social mission of your software.

There are ethical implications that are involved in developing software, which are easy to overlook. As a hypothetical example, imagine a piece of software that can assess the chance of survival of animals in a shelter: this software could be misused to decide which animals to kill! On a more realistic example, interpretation of weather models - can be misused to deny climate change. Another example is software which might find use in military applications: the original developers of the software might have concerns about such applications.

7.3.3.1 Take home message:

- "Sit down and think about the end of your journey, at the beginning of your journey."
    - Main book recommendation - (claims, success, had achieved things: allows you to think about problems/issues up front) - The Founder's Dilemmas[112].
    - Other books
        - Section on ethics - Things a Computer Scientist Rarely Talks About[113]
- Choices you make early will define your culture (e.g., how shareable / integratable your code is)
    - There is a move beyond just open source licences
    - Increase in ethical clauses
        - Used for good (e.g., non-military use; some will applaud)

[110] https://choosealicense.com/

[111] https://tldrlegal.com/

[112] https://press.princeton.edu/books/paperback/9780691158303/the-founders-dilemmas

[113] https://web.stanford.edu/group/cslipublications/cslipublications/site/1575863278.shtml

        - Used for bad (e.g., racist clauses)
    - Increase in economic clauses (to help sustain the initiating projects)
- Three axes to think about
    - Technical, Legal, Ethical
- Original reasons for getting into open source
    - Advocacy, Openness, Transparency
- Times changed - more complex issues needed to be dealt with[114] [115]
    - Societal issues (e.g., inclusion - beer and pizza is exclusionary to gluten free people) and technical & legal choices are shaped by the society they are part of.
- Important about being intentional about the communities that we build - although there are no easy answers when it comes to societal factors - one has to take those onboard.

[114] https://osaos.codeforscience.org/

[115] https://discover-cookbook.numfocus.org/

8 Human factors and new development in preserving and sustaining research software

This section describes the third session of the workshop. This session includes featured sections (author enhanced and expanded text beyond a summary) on discovering the architecture of scientific software, how software preservation is necessary for reproducibility and how Open Source Center's are supporting open research in Brazil. The summary sections covered making inclusion a core feature of our work, reproducibility with the RO-Crate standard and Common Workflow Language, how to measure sustainability of academic software and concluded with an introduction to the UK reproducibility network.

The discussion session which followed covered how communities could handle burnout, research software roles and the place of software in the scholarly record.

8.1 Featured

8.1.1 The Lost Architectures of Scientific Software and How to Find Them

Colin Venters[116]

Modern scientific and engineering research is highly dependent on software. Its importance in driving forward advances in research in the field of computational science and engineering has resulted in calls for it to be classified as a first-class experimental scientific instrument. However, software as a research instrument has not reached a level of maturity compared with the conventional tools of empirical and theoretical science [21]. Why? Research software is principally developed by end-user developers who have a limited understanding and application of fundamental software engineering concepts, principles, and techniques, combined with a "code-first" approach to development, which is in part driven by the complexity and uncertainty of the problem. This results in research software with suboptimal software design, if any, leading to accidental complexity, technical debt, code smells, and an increase in the risk of software entropy. Similarly, while Research Software Engineering aims to facilitate the creation of well-designed, reliable, efficient software to solve research problems there is little empirical evidence to demonstrate that the research software created is well designed, if at all, understandable, maintainable, and extensible. The consequences of accidental software complexity lead to a range of rotten symptoms, including software rigidity, fragility, immobility, and viscosity that are a pathway to stagnation, decay, and the long-term decline of essential research software investment [22].

Sustainability is generally understood as the capacity of a socio-technical system to endure [23]. While several communities have attempted to address the challenges of achieving sustainability from their different perspectives, there is a severe lack of common understanding of the fundamental concepts of sustainability and how it relates to software systems. As a result, there is no agreed definition of software sustainability or how it might be achieved. While there have been several contributions to formalise a definition of software sustainability, the concept remains an elusive and

[116] Links to a video and transcript available at - https://wosss.org/wosss21/S3-ColinVenters

ambiguous term with individuals, groups and organisations holding diametrically opposed views [24]. This lack of clarity ultimately leads to confusion, and potentially to ineffective and inefficient efforts to develop sustainable software systems.

The future of scientific and engineering enterprise requires a resilient ecosystem of software [25]. Software design is a key component of sustainable software, which starts with software architecture [26]. Software architectures represent the set of structures required to reason about the system, which comprises both software elements, their properties, and relationships [27]. Software architecture is fundamental to the development of technically sustainable software as they lay the foundation for the successful implementation, maintenance and evolution of sustainable software systems in a continually changing execution environment by providing a mechanism for reasoning about core software quality requirements that contribute to sustainability as a first-class, composite software quality [28]. By addressing software sustainability at the architectural level, it allows the inhibiting or enabling of systems quality attributes, reasoning about and managing change as the system evolves, predicting system qualities, as well as measuring architecturally significant requirements. However, the ability to determine sustainability as a core software quality of a software system from an architectural perspective remains an open research challenge, and existing architectural principles need to be adapted and novel architectural paradigms devised. In addition, there is a pressing need for new tooling to fit today's emergent and dynamic environments, where essential research software is explicitly designed for continuous evolvability and adaptability without incurring prohibitive architectural, technical debt [29].

This micro-talk argues that sustainable software is that which is explicitly designed for continuous maintainability and evolvability without incurring prohibitive technical debt and a negative impact on the dimensions of sustainability and presents the results of a case study on the technical sustainability of the MERLIN++ particle accelerator tracking library originally developed in 2000 at Deutsches Elektronen-Synchrotron.

8.1.2 Software preservation is necessary for reproducibility

Vicky Rampin[117]

As supporting research reproducibility continues to take shape throughout various scholarly communities, we've seen several tools arise to help. However, most if not all the current tools for reproducibility were made for short-term replay of research, relying on container technology and having researchers manually configure their computational environments. This is problematic for long-term access to research in many ways, particularly because it's often incredibly difficult (neigh on impossible) to uncover all the dependencies that computational research touches, and even harder to make sure those persist in the long-term (and we know that the computational environment where research takes place directly affects its analytical result; see the case of the Willoughby-Hoye[118] scripts). The more that we care about reproducibility, the more it becomes clear that we rely on software preservation in several ways to enable that reproducibility.

[117] Links to a video, slides, and transcript available at - https://wosss.org/wosss21/S3-VickyRampin

[118] https://arstechnica.com/information-technology/2019/10/chemists-discover-cross-platform-python-scripts-not-so-cross-platform/

In this talk, I'll go over ReproZip[119], which adds automation, extensibility, and preservation of reproducible research to the current landscape of tools. ReproZip has been in development at New York University since 2013, and ReproZip bundles that were created back then are still fully rerunnable and reproducible today, which few other tools can boast (if any).

ReproZip helps researchers make long-term reproducible bundles of their work in two steps:

1. the tracing step: researchers run ReproZip at the same time they run some analyses, program, pipeline, etc., and ReproZip detects and captures the source of everything that the process touches (input data, source code, environment variables -- everything!).

2. the packing step: researchers bundle all that information in a .rpz file. The bundle is generalizable enough to be able to be viewed and reused by several other tools (which improves the sustainability of ReproZip bundles).

ReproZip bundles can then be used to automatically set up the original researchers' computational environment and project workflow on someone else's computer, which is ideal for computational reproducibility. Secondary users can verify the original researcher's work, but also extend it by using their own input data. Because ReproZip packs all the source of every dependency of a script, workflow, etc., as well as the provenance information (for instance, the order in which the original researcher runs multiple scripts), it allows for high fidelity and long-term reproducibility at low cost to the user (in terms of time, resources, and labour).

Among the many benefits of ReproZip, sustainability and preservation is the most unique in the landscape of existing tools. The rpz file generated by ReproZip is preservation-ready from the time it is created, for a few main reasons:

- Flexibility: the rpz file is completely agnostic to the unpacking technology being used, and this has allowed ReproZip to be leveraged in many preservation contexts. ReproZip itself also uses a plugin model such that unpackers can be added and removed as the march of time goes on. This has already resulted in ReproZip being used widely for many use cases.

- Completeness: the rpz file contains all the necessary files to reproduce the packed research, as well as a highly detailed metadata file (config.yml) that lists all the technical and administrative metadata about the computational environment being used, the workflow steps, and other dependencies packed. If for some reason there are no containers or virtual machines in the future, then someone could use this metadata file to rebuild the computational environment and use the files in the bundle itself to reproduce the work.

[119] https://www.reprozip.org/

Reprozip is already integrated in several systems and workflows to support many different use cases, again adding in valuable automation, sustainability, and preservation capabilities. Some of these include:

- Computational science tools (NeuroDocker[120] to minify docker containers, Spot [30] to reconstruct provenance graphs, Model Insertion Checker[121] (part of DARPA's World Modelers program) to trace and pack model execution)
- Peer review (e.g., SIGMOD[122])
- Querying metadata at scale (e.g., WholeTale [31], explore web archives[123])
- Reproducibility component of other platforms (Cloud of Reproducible Records[124])
- Digital preservation infrastructure (e.g., Emulation as a Service Infrastructure[125], ReproZip-Web[126])

ReproZip clearly adds value in long-term access to reproducible research, as well as providing a pathway for the preservation of boutique research software and environments.

8.1.3 The FLOSS Competence Center as an Enabler of High-Quality Open Research Software in Brazil

Kelly Rosa Braghetto[127]

The Free/Libre Open Source Software (FLOSS) Competence Center (CCSL[128]) of the University of São Paulo (USP [129]), Brazil, encompasses activities related to undergraduate and graduate education, research, development, and publicising of free and open-source software. It is composed of formally established centres at the Institute of Mathematics and Statistics in the city of São Paulo and at the Institute of Mathematics and Computer Science in the city of São Carlos, both in São Paulo state. The centre's primary goal is to foster the development, research, and adoption of FLOSS both inside and outside the university, by providing for users and developers high-quality resources and expertise on the various topics related to open-source software.

---

[120] https://github.com/ReproNim/neurodocker

[121] https://mic-cli.readthedocs.io/en/latest/overview/#step-2-trace-your-model-execution

[122] https://reproducibility.sigmod.org/#process

[123] https://twitter.com/anjacks0n/status/1323719989313048579

[124] https://github.com/usnistgov/corr-reprozip

[125] https://twitter.com/euanc/status/1143966909421019136

[126] https://reprozip-web.readthedocs.io/en/latest/

[127] Links to a video, slides, and transcript available at - https://wosss.org/wosss21/S3-KellyRosaBraghetto

[128] https://ccsl.ime.usp.br/en

[129] https://www5.usp.br/

Since its creation in 2008, the centre has been performing a fundamental role in supporting the Brazilian scientific community to embrace openness. The centre offers training, consulting, and hosting to researchers from any field of science, to support the production of high-quality open research software. It aids researchers not only on tools, platforms, and licences but also on methodologies, studies, and best practices in software development. The FLOSS Competence Center team groups some of the main researchers on Software Engineering and FLOSS of the country.

Despite its importance and high impact, the FLOSS Competence Center's scope of action is too small, considering Brazil's large extension and inequalities. There is a lack of country-wide open science practical initiatives and incentives of funding agencies and research institutions, which should provide physical infrastructure, information, and human resources to assist researchers.

Brazil started formal initiatives to adopt open government data in 2011 by creating the Law on Access to Information and co-founding the Open Government Partnership (OGP), which now has 77 other signatory countries. The biennial Open Government National Action Plans have raised awareness and commitments to open science. But it is worth mentioning that Brazil became a leader of the open access movement much before that. The Scientific Electronic Library Online (SciELO[130]), recognized as one of the most important open-access programs in the world, was established in 1997. SciELO's publication model has been adopted by 15 other countries, in a network that contains 1200 open access journals, which publish an average of 50 thousand papers per year.

Concerning the management of open research data, coordinated actions started to take shape only more recently. In 2018, the Brazilian Ministry of Science, Technology, Innovation and Communication (MCTIC) created a Working Group to draft the National Policy for Open Science, to give the guidelines for the national policy for research data management in Brazil. At the end of the same year, Brazil adhered to the GO FAIR initiative through the creation of the GO FAIR Brazil Office, hosted by the Brazilian Institute of Information in Science and Technology, a research branch of the MCTIC. It is one of the first GO FAIR national offices established outside Europe. The first active implementation network in operation in Brazil is in the health domains; other fields are in the process of adherence negotiation.

A few distributed efforts stand out while the national policy is not implemented, mainly on the biggest public research institutions (such as USP) and wealthiest state funding agencies (such as FAPESP[131]). The State of São Paulo Research Foundation (FAPESP) is one of Brazil's main funding agencies for scientific and technological research. FAPESP has been defining policies and fostering initiatives towards open science aligned with what other countries practise, aiming to increase the dissemination and the scientific, social, and economic impact of the Brazilian research it funds.

Since 2017, FAPESP requires from researchers Data Management Plans upon submission of project proposals. The data produced in projects funded by FAPESP

[130] https://www.scielo.br/

[131] https://fapesp.br/

must be stored in institutional repositories, preserving them, and making them openly accessible (subject to the applicable standards and constraints).

Also in 2017, FAPESP started a workforce to develop a state network of open research data repositories, formed by the São Paulo state's six public universities (including USP), the Aeronautics Institute of Technology (ITA), and the Brazilian Agricultural Research Corporation (EMBRAPA). The repository that gathers research data metadata [132] from the network institutions was launched in 2019. The network's infrastructure is also being used to share pseudonymized data of Covid-19 patients (~800K) from collaborating health institutions, contributing to essential research in several countries.

In 2019, FAPESP formalised its open access policy for publications, under which all journal papers that result from FAPESP-funded research must be made publicly available in institutional open access repositories (if this does not violate copyright rules).

Some funding lines at FAPESP already require that all software developed within a funded project be licensed as free software. However, differently from the case of research data, there are no institutional supporting resources (such as guidelines and metadata repositories) to help researchers appropriately develop, preserve, and disseminate their research software. Until now, the sustainability of research software seems to be out of the official working groups' radar.

This scenario highlights the importance of reinforcing and replicating the work that the FLOSS Competence Center is doing. Moreover, it exposes the urgent need for the Brazilian government, research institutions, and funding agencies to finance and cooperate to build appropriate infrastructure and establish models, skills, and policies to sustain the Brazilian research software, along with the ongoing initiatives on research data.

8.2 Summaries

8.2.1 Changing our ways: Making Inclusion a Core Feature

Emma Irwin[133]

As we look to technology to solve some of the world's biggest challenges it's critically important that we understand how the history of racism, sexism, ableism, casteism and other biases has shaped the technology we use today; that while the small steps being made to increase diversity, to address hate and harm online are encouraging, the true potential lies in our ability to move from 'having good intentions' to systematically evaluating, evolving and changing how we build software.

Open source software, which is less diverse than tech overall (despite the notion of openness) sadly, has as many stories of toxicity, exclusion, and harm as it does innovative success stories. There is plenty of research, and storytelling that tells us why. We even have a set of metrics being developed both by a cross-community effort

[132] https://metabuscador.uspdigital.usp.br/

[133] Links to a video and transcript available at - https://wosss.org/wosss21/S3-EmmaIrwin

called CHAOSS[134] and work by the UN to define a framework for digital inclusion. What's still lacking is a way to apply this knowledge and associated actions into our everyday engineering and community practices.

In this talk, I'll share some of my research, the CHAOSS project metrics and early work to embed metrics for inclusion into the everyday with the hope to inspire new ideas to ensure that the solutions we're building for the world, reflect the challenges and potential of everyone in it.

8.2.2 Reproducibility; Research Objects (RO-Crate) and Common Workflow Language (CWL)

Stian Soiland-Reyes[135]

The use of digital methods and computational analysis is now ubiquitous across sciences and research disciplines. However, there is a growing concern that while modern computing accelerates scientific development and progress, it can come at the cost of reduced reproducibility and a difficulty of communicating the methodology to other researchers, particularly through traditional scholarly articles as text and static figures. Research Objects [32] have been proposed as a unit of scholarly communication, gathering raw data, software, results, figures and documents, described and inter-related using Linked Data, and as an aggregation cited by its own persistent identifier.

RO-Crate[136] is a realisation of Research Objects using off the shelf Web standards (JSON-LD) and vocabularies (schema.org), with a developer-friendly lightweight approach and a set of best-practice guides for capturing “just enough” structured metadata, being interoperable with Linked Data technologies, and extensible for domain-specific needs. RO-Crate is being developed as a community-led project, supported by open source tools, and is being adapted for a wide range of different scientific domains and use cases.

8.2.3 On the Sustainability of Academic Software in Software Engineering

Christina Von Flach Garcia Chavez

The increasing adoption of academic software has made modern Science dependent on the technical sustainability of software. Unsustainable development of academic software hinders reproducibility, one of the Science pillars. In addition, according to Howison and colleagues, the non-sustainable development of academic software can lead to a “dysfunctional chaotic churn”, characterised by the existence of several similar projects, with disconnected communities, few users, and a short life cycle, among other anomalies.

---

[134] https://chaoss.community/

[135] Links to a video, slides, and transcript available at - https://wosss.org/wosss21/S3-StianSoiland-Reyes

[136] https://www.researchobject.org/ro-crate/

There are few studies on the technical sustainability of academic software in Software Engineering, especially in the field of static analysis, with a long tradition in the development of tools to support research in different areas.

In this talk, we present the results of an exploratory study on the technical sustainability of academic software in the Software Engineering field [33]. We analysed 60 static analysis software projects with the purpose of characterising its technical sustainability, with respect to publicity (or availability), recognition and life cycle, from the perspective of the scientist (developer or user) of academic software in the context of two important software engineering conferences; 40% of the software analysed either decayed or became lost.

8.2.4 An Introduction to the UK Reproducibility Network

Andrew Stewart[137]

In this talk, I will provide a brief history and overview of the structure of the UK Reproducibility Network (UKRN[138]). I will cover the goals of UKRN's recently funded Research England Development Fund bid, the activities that this funding will support over the next 5 years, and some of the challenges that UK institutions face in transitioning to a more open and transparent way of conducting research.

The UK Reproducibility Network (UKRN) was launched in March 2019 by Marcus Munafò (University of Bristol), with activities coordinated by the Steering Group[139] and an Advisory Board[140]. At its inception, the UKRN brought together several individuals and pre-existing groups who had become increasingly focused on issues related to reproducibility, replicability, and transparency in research. Many of the early interactions took place via social media and revealed the extent to which there were overlapping concerns across disciplines and institutions around openness in research.

The creation of the UKRN provided a way to bring together those individuals and groups to work together towards solutions and advocate for cultural change. The main grassroots activities of the UKRN occur via the local networks, local network leads, and institutional representatives. Each local network (of which there are currently 63), many of which are ECR-led, engage in grassroots activities, such as forming Open Research Working Groups, that cross traditional discipline boundaries and act to promote the aims of the UKRN at their home institution. These activities can include setting up ReproducibiliTea journal clubs[141], RIOT Science Clubs[142], organising regional or national workshops and conferences on transparency and reproducibility in research, organising local training events, and lobbying senior leaders to raise awareness and promote the importance of openness and transparency in research. The UKRN's aims overlap substantially with the aims of the organisations (such as UKRI – including Research England and numerous individual research councils,

[137] Links to a video, slides, and transcript available at - https://wosss.org/wosss21/S3-AndrewStewart
[138] https://www.ukrn.org/
[139] https://www.ukrn.org/steering-group/
[140] https://www.ukrn.org/advisory-board/
[141] https://reproducibilitea.org/
[142] http://riotscience.co.uk/

Wellcome, the Software Sustainability Institute, and Jisc) that form the External Stakeholder Group[143].

In August, Research England confirmed funding for the UKRN project "Growing and Embedding Open Research in Institutional Practice and Culture" over 5 years, beginning on September 1st, 2021. This is a substantial and ambitious project with the overall aim of accelerating the uptake of high-quality open research practices across the 18 institutional members of the UKRN, and ultimately across the sector. There are three main workstreams to the project. The first involves working with academic communities to identify training gaps needed for transparent research practices, the (iterative) development of new training materials to meet the needs of these communities, a series of train-the-trainer events, and the curation of pre-existing and new training materials so that they are freely available to all (and not just to those institutional members of the UKRN). The second workstream involves developing and delivering a framework for the evaluation of institutional practice and learning in open research. Together these first two workstreams will lead to activity in workstream three which involves sharing effective practice across disciplines and institutions. Crucially, none of these workstreams assumes a "one size fits all" approach but rather will operate through collaboration with different research communities (across institutions) and through an understanding that the definition of "research openness and transparency" will mean different things in the context of different disciplines and different research methods.

One of the greatest challenges associated with re-configuring how research is carried out and reported is in terms of the incentive structures that – at an individual level – often does little to encourage the adoption of transparent research practices. Critically, the incentive structure that institutions face also does little to encourage behaviour change at an institutional level. While individuals can be incentivised to change their behaviour through the evaluation of their adoption of transparent and open research practices associated with the processes of hiring, probation, and promotion, institutions can also be incentivised to change their behaviours. These institutional incentives are likely to be financial, and so UKRI (in the form of REF) and the individual research councils have a clear role to play in rewarding institutions that embed research openness and transparency in their local research environments.

8.3 Discussions

8.3.1 Engaging communities that proactively manage burnout

Neil Chue Hong, Jess Farrell, Jean-Noël Grad, Colin Venters, Carlos Martinez, Kelly Braghetto

8.3.1.1 Key Points

- Academic leadership need to be aware of software issues
    - Working with legacy code can be slow and impact perceived productivity

[143] https://www.ukrn.org/stakeholders/

        - There is no one tasked with managing code complexity (refactoring code or improving modularity by separation of concerns).
    - If the focus is solely on publication rather than quality software, quality will be impacted as it takes time to write sustainable, documented, and tested code.
    - Important and complex features used by key users can make code bases harder to work with; the time and complexity aspects need to be acknowledged when estimating new work
    - Leaders should look out for burnout, an environment where people can self-report comfortably about burnout is also vital
    - The incentives for writing good code are a work in progress without sympathetic leadership; the change to where this becomes important won't happen.
    - Some funders in some countries (e.g., a local funding agency in Sao Paulo, Brazil) are starting to mandate software be open source; leaders need to be cognizant of the funder requirement and allow time to make sure they are met.
- Create an environment which fosters community and support without adding unrealistic key performance indicators (KPIs)
    - Set realistic expectations
    - Be aware of the differences of online and in person work and how burnout might manifest
- Aware communities which allow community leads to identify early signs of burnout and allow community members to step back when they need to.
- Avoid fragmentation of a community by fostering good communication.
    - If different groups have different needs better to work together otherwise there will now be two systems that need maintaining

8.3.2 The state of research software roles

Hilary Shiue, Peter Doorn, Shoaib Sufi

8.3.2.1 Key Points

Software roles included Research Software Engineering, Data Stewards, Digital Humanities, and Information Science.

- The Maturity of research software engineering/sustainability is different in different parts of the world.
- The UK and Europe have more developments in general than the US or other countries, which is reflected in the job market, available positions in relevant institutions.
- However, even if there are positions available, it is notable that they are often project-based, instead of established positions within institutions (although this is changing)

8.3.3 What should software's place in the scholarly record be?

Vicky Rampin, Adam Jackson, Carina Haupt

Software should be on a par with other research materials and should be with other research materials.

Software and data are put in different places than articles; this is mainly for practical and legacy reasons but maybe this needs to be re-visited.

The ability to cite software is getting better (see the adoption of the Citation File Format (CFF[144])). However fundamental questions about what and where to cite software are still in progress.

Software publications can be seen as a hack to get credit for software, although rather than writing about software in a normal journal the Journal of Open Source Software (JOSS[145]) is a good middle ground as the peer review is on software and the 'paper' is a paragraph telling you that the software exists.

There are questions around how effort on software projects is measured and rewarded. Currently rewarding software is not really established. The way funding works also is a problem; with innovation supported far more than maintenance.

It is worth considering why software should be included in the scholarly record. There are two main purposes, recognition, and reproducibility.

Metrics create an incentive not to dilute the recognition people can get.

Software that primarily lives in non-public places, e.g., institutional version control repositories is quite invisible to the public scholarly record (even if "available on request"). Licensing tends to be somewhat relaxed around this kind of private development – and can become a problem when publishing larger software that has dependencies on this type of software. Making sure software has a licence will clear up any issues with licence compatibility.

Where code is heavily licence-encumbered or even cannot be distributed as source, it is still useful to have precise versioning and CHANGELOG summary information such that at the very least the code's metadata can be included in the scholarly record.

[144] https://citation-file-format.github.io/

[145] https://joss.theoj.org/

9 Sustaining the community and promoting (human) infrastructures for software sustainability.

This section describes the fourth session of the workshop. This session includes summaries on mapping people-related activities in the research software community, humans as infrastructure, ARDC software sustainability efforts in Australia, software sustainability and the European Open Science Cloud, the German RSE association and how the Chan Zuckerberg initiative is supporting open source in science.

A panel discussion on research software infrastructure followed the presentations in this session and the results and comments on them are included in the last section.

9.1 Summaries

9.1.1 The People Roadmap: Mapping people-related initiatives in the research software community

Michelle Barker[146]

The People Roadmap is a Research Software Alliance (ReSA[147]) consultation to map the landscape of research software community initiatives focused on people-themed issues, as part of ReSA's mission is to bring research software communities together to collaborate on the advancement of the research software ecosystem.

28 organisations, community initiatives and/or projects in the research software community have been profiled as part of the creation of the People Roadmap, to facilitate identification and opportunities for accelerating efforts to address major issues related to personnel challenges. The People Roadmap aims to increase understanding on how to create an environment where research software personnel are recognised, have appropriate skill sets and access to inclusive communities, within policy and infrastructure environments that support their work.

The People Roadmap was conceived in response to the evolution of a range of research and software areas, including 1) the rise of open science (which includes open software), 2) increased understanding of the need for advanced digital skills in the research community (including research software engineering) to achieve the aims of open science, and 3) the development of the Research Software Engineering movement to recognise and support the Research Software Engineers who are responsible for the development and maintenance of a significant amount of research software.

9.1.2 The fundamental part of software: the human infrastructure

Martin Hammitzsch[148]

[146] Links to a video, slides, and transcript available at - https://wosss.org/wosss21/S4-MichelleBarker

[147] https://www.researchsoft.org/

[148] Links to a video and transcript available at - https://wosss.org/wosss21/S4-MartinHammitzsch

Infrastructures are facilities, services, and installations needed for the functioning of communities. Whereas facilities and installations can be an array of equipment set up for use, facilities are also abilities and aptitudes to perform activities and to serve particular functions. In this regard, services are assistance and help, they are abilities set up as work and duties done for others. Following these definitions, we look at different aspects of the fundamental part of research software – the human infrastructure – and what it takes to ensure its sustainability and thus software sustainability.

The human infrastructure of research software involves communities, on the one hand, those having a demand for software, and on the other hand, those meeting these needs. Ensuring measures that support both necessitates firstly, common policies, guidelines and procedures that community members can follow; secondly, awareness, training and exchange that enable community members; thirdly, work capacities that community members can make use of; and, finally, funding that allows members of the communities to take advantage of support measures as well as to provide these support measures.

The measures comprise community building and the sustaining of these communities. They are manifold and, among others, e.g., involve the elaboration and adoption of best practices with policies, the passing on of knowledge and skills with training, the initiation and promotion of collaborations and exchange with hacky hours and fellowship programs, the giving of advice with consulting, and the implementation of software with research software engineering.

In some places these measures are implemented in a structured manner and with success, in other places they depend on the voluntary commitment of individuals and groups. Sharing resources across wider geographical areas with a coordinated and structured approach would, however, be advantageous in Europe across the board. The cooperation between those involved regionally and locally has already established itself nationally and internationally at the working level. But the collaboration between European countries in a structured approach seems like the next logical step to address the shared needs with a coordinated and efficient use of resources.

We will look at some of these aspects from a human infrastructure and community perspective that is expecting trusted, reliable, and supported infrastructures containing multi-purpose and tailor-made software for their research work. Finally, we aim to address unsolved issues in understanding research software as (human) infrastructure and in managing, implementing, operating, maintaining, and developing it as such.

9.1.3 What we are doing towards software sustainability

Tom Honeyman[149]

The Australian Research Data Commons (ARDC[150]) is a national facility supported by the federally funded national collaborative research infrastructure strategy (NCRIS)

[149] Links to a video and transcript available at - https://wosss.org/wosss21/S4-TomHoneyman

[150] https://ardc.edu.au/

scheme. In this role the ARDC is a provider or coinvestor in several areas of digital infrastructure, skills development, and guidance, and has national programs in storage, compute, data, informatics services, skilled workforce, policy, platforms and now software.

The software program is a new initiative from the ARDC signalling an interest to move into research software. The program is operating under a strategic aim to achieve recognition of research software as a first-class output of research. As part of this work, a draft "National Agenda for Research Software" was released in June this year to stimulate discussion amongst partners in action towards this strategic aim, and to consider what contributions they might make. Based on responses to the draft agenda, the ARDC is now starting to initiate activities tied to the framework laid out in the agenda.

The agenda builds on three high level actions to "See, Shape and Sustain Research Software" which consider:

- (See) the visibility and availability of research software,
- (Shape) the application of software engineering best practice to shape research software for easiest and broadest meaningful reuse, and
- (Sustain) the maintenance and longevity of relevant research software infrastructure (including particularly the workforce of maintainers and developers as a form of soft infrastructure).

To turn these three high level actions into a tractable set of tasks, they are each further broken down into consideration of the necessary infrastructure, guidance, communities and advocacy work needed to make them possible, easy, normal and codified. This gives an overall set of 12 interrelated actions that form the agenda. Further to this, national stakeholders are characterised and then mapped to these actions.

The ARDC is considering activities across all 12 actions in the agenda, but the early focus will be on areas of infrastructure and communities.

We will be considering what is possible under the status quo (i.e., areas of infrastructure) by commissioning three reports, corresponding to three forms of infrastructure (informatics infrastructure, software assets as infrastructure, and soft infrastructure or human capital). The three reports will:

- characterise the national informatics landscape that supports software assets,
- the existence and distribution of Australian supported research software, and
- the existence and distribution of the workforce supporting the development and maintenance of research software infrastructure.

These three reports will form a baseline measure of our national capacity in research software.

To drive development and normalisation of best practice, we are looking to build or support national communities of practice focussing on researchers, support staff, and career research software engineers. We will also look to seed the development of

local, regional, and national communities that bridge the worlds of research and software engineering.
To ease change, our initial areas of guidance development will focus on easing the adoption of best practice software publishing and software citation, as well as socialising and facilitating adoption of the nascent FAIR4RS principles.

Our initial work in advocacy will focus on code availability, but we are anticipating expanding to include credit for new software, and structures to support maintaining research software.

9.1.4 Research Software Sustainability in the European Open Science Cloud (EOSC)

Konstantinos Repanas and Ignacio Blanquer[151]

The European Open Science Cloud (EOSC [152]) aims at providing European researchers a virtual environment for open access to services to store, share, process and reuse research data and other research digital objects, such as software. Scientific software is both a key asset for the reproducibility of science and a research output that is re-usable to create new knowledge. We present the EOSC initiative and how research software preservation, reuse and quality enhancement is increasing in importance and focus, covering the future plans of the EOSC Association Task Forces and how they will address the sustainability of Research Software.

The presentation will cover first the efforts of the working groups of the EOSC Executive Board that have now ended their term. The activity in these Working Groups have produced numerous pieces of work, and the presentation will stress the links to research software sustainability in the Working Groups of Architecture and Skills and Training, focusing on the recommendations of the "Scholarly Infrastructures for Research Software" report in the former and the profile of the Research Software Engineer in the latter.

We will then cover the EOSC Association and the Infrastructure for Quality Research Software Task Force, which addresses several key aspects for software sustainability. We will present the aims and the expected results of this Task Force.

9.1.5 (Inter)National Community Efforts by the German Association of Research Software Engineers (de-RSE)

Alexander Struck[153]

The talk provides an overview of past, current and future efforts within the de-RSE community to create sustainable (human) infrastructure and working environments. Half a decade ago, first conferences, workshops, calls and national working groups appeared where focus had shifted from research data to software. The UK RSE

[151] Links to a video and transcript available at - https://wosss.org/wosss21/S4-KonstantinosRepanasAndIgnacioBlanquer
[152] https://eosc.eu/
[153] Links to a video and transcript available at - https://wosss.org/wosss21/S4-AlexanderStruck

conference in 2016[154] certainly affected or ignited community efforts abroad. Similar efforts by German organisations for the Digital Humanities and Natural Sciences[155] supported national efforts to gather a community around research software. The alliance of all German research organisations initiated a working group and later published well-received guidelines for software aspects. The same year 2016 saw its first call for funding applications to improve software and infrastructure.

During the following years, conferences and workshops had tracks on software, and a community started to emerge. Discussions about lobbying for RS-Engineering and Engineers also led to the foundation of the de-RSE society, and a year later in 2019, the first (inter)national conference took place with about 200 participants.

The past years also saw workshops on software policy, publishing, licensing. Software Carpentries received attention. The de-RSE community flourished, as measured in society members, mailing-list size, and twitter followers. More and more local chapters appear and effort is spent on international collaboration.

Members of the national community are currently involved in efforts to instantiate a European Software Sustainability Institute. Several members took part in a larger DFG-sponsored workshop and published a position paper on our environment. Others and the speaker are involved in the FAIR4RS WG. There we try to find a usable definition for research software, how the FAIR principles need to be amended to be applicable to our software and how we could improve adoption.
Unfortunately, recognition of software as a research result is still lacking as well as recognition of RSEs as valuable human infrastructure for excellent research. More work is needed here to initiate changes towards sustainable environments for RS-Engineers and for the software being written or re-used.

9.1.6 Supporting the creators and maintainers of essential open source software

Carly Strasser[156]

Most open source software for science is undervalued and lacks funding for maintenance, growth, development, and community engagement—especially after the initial phase when it's linked to original research. The Chan Zuckerberg Initiative's Essential Open Source Software for Science program [157] supports open source projects that are important for biomedical research via funding for maintainers, community development, and software upkeep.

[154] https://society-rse.org/events/rse16/

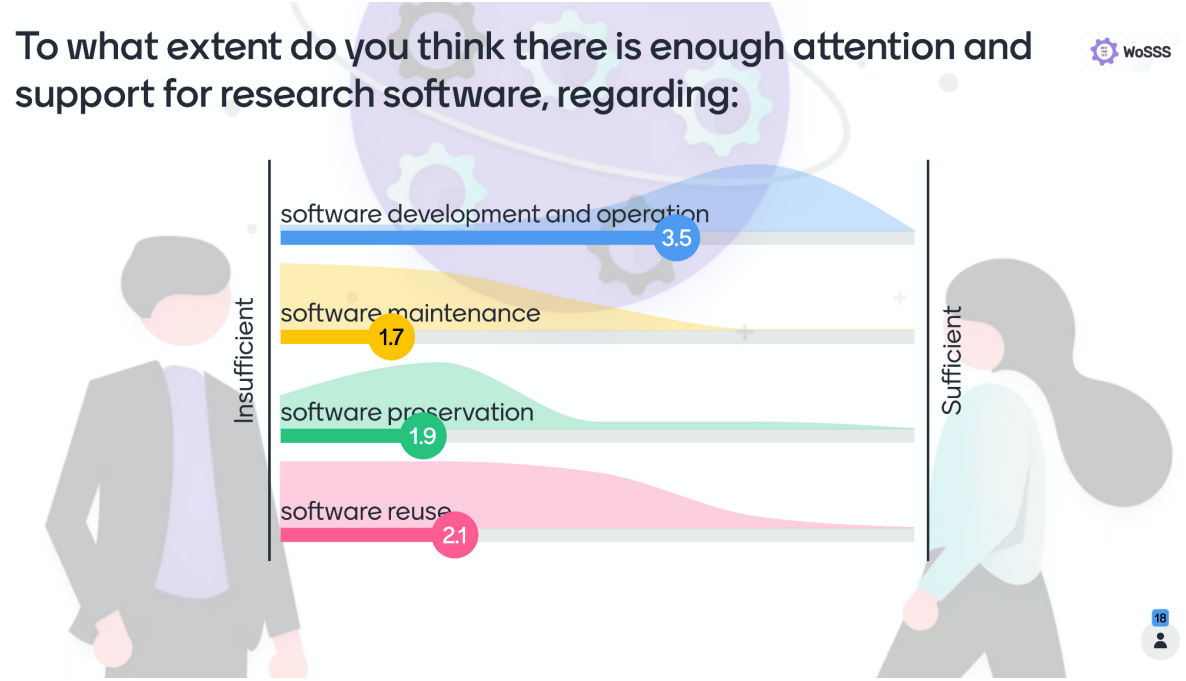
[155] Examples include FORGE 2016 - "Future of Reproducible Geoscience Experiments" in Potsdam, October 2016 and Helmholtz Open Science Workshop "Access to and reuse of scientific software" November, 2016 - https://web.archive.org/web/20190926060607/https://www.hzdr.de/db/Cms?pOid=47484&pNid=243

[156] Links to a video and transcript available at - https://wosss.org/wosss21/S4-CarlyStrasser

[157] https://chanzuckerberg.com/science/programs-resources/open-science/

9.2 Panel on research software infrastructure

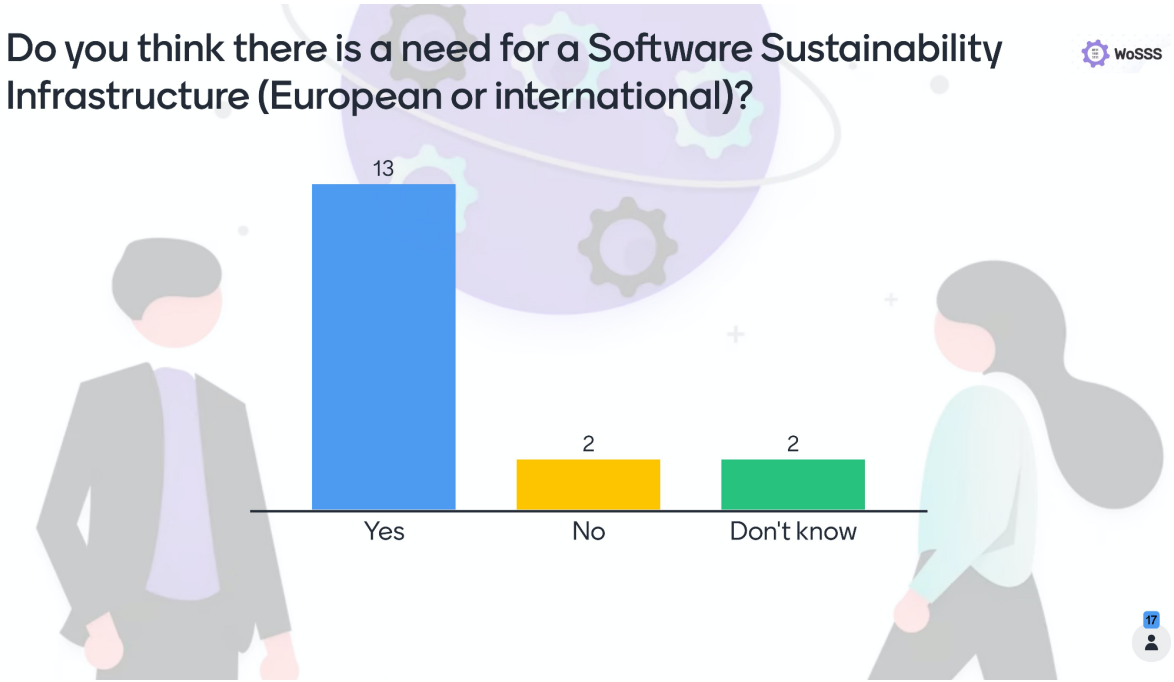
The panel discussion[158] made use of a Mentimeter poll to take the opinions of the audience into account, to which the panellists (the speakers in Session 4) commented. About twenty people took part in the Mentimeter voting, although not everybody voted every time.



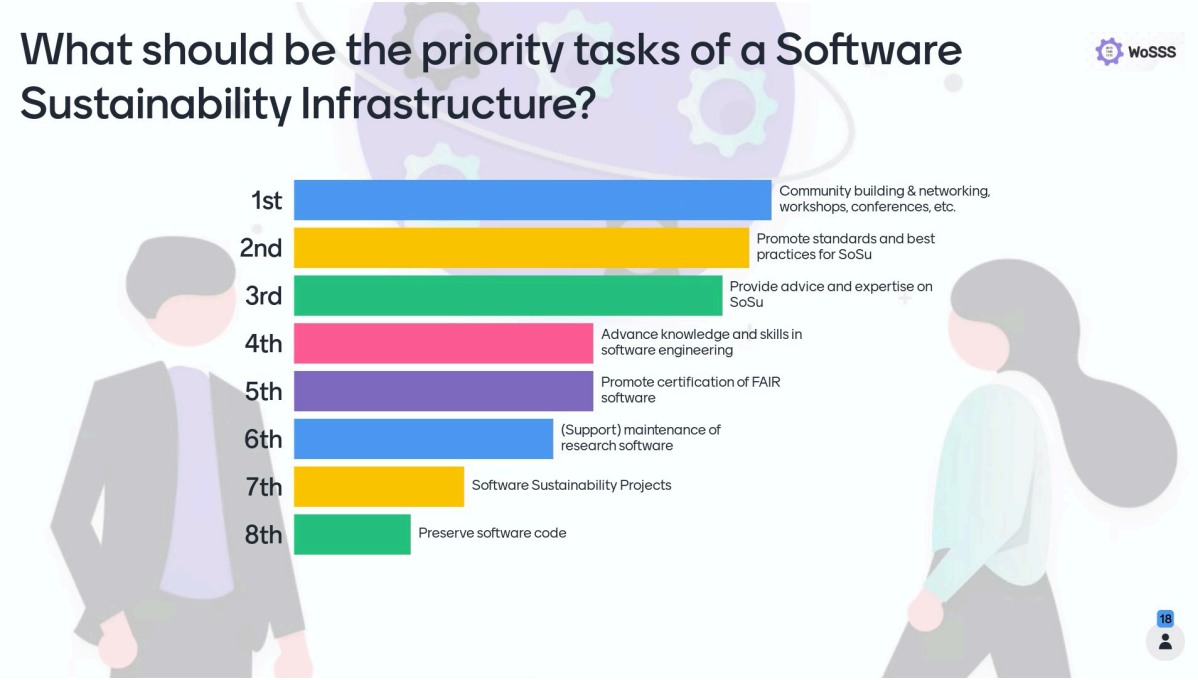
Comments: this question was not about the workshop, but about how people experience the attention for the subjects mentioned in the graph in their work situation. It is very clear that most voters think the attention for software maintenance, preservation and reuse is significantly less than for software development and operation in general. The lack of attention to maintenance is a trend that has been identified for a long time, and although many initiatives are currently aiming to improve this situation, the focus is still more on developing new things rather than reusing existing software and infrastructure.

[158] A video of the panel is available - https://www.youtube.com/watch?v=36_nLoP120c&list=PLXAvKzjdTsrxFqbjWtxHjfJc0RN6jMwZg&index=29

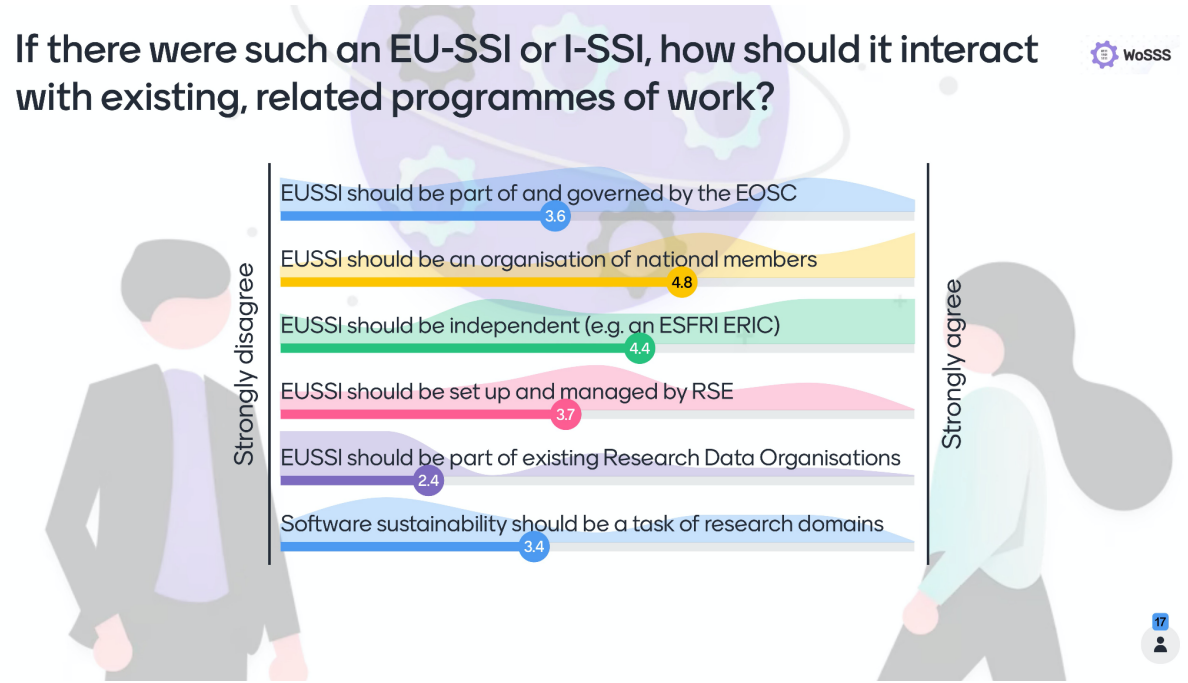
Comments: This question was about norms in the future, 10 years from now. While it seemed that an International/European SSI, FAIR practices, RSE careers and curricula exposure for software/data topics all resonated with participants it was interesting to note the difference in distribution. FAIR practices had the most responses closer together, there was quite a difference of thought on an international/European SSI, while RSE careers and curricula exposure had similar distributions highlighting the majority thought they would be more established practices although there were still some who thought these were a longshot. This is an area that needed further investigation, you would have thought that those attending WoSSS21 would all rate these possibilities highly, but an exploration would help decide whether respondents did not believe these were the right practices 10 years from now or whether they deemed them important but the aspiration unrealistic.



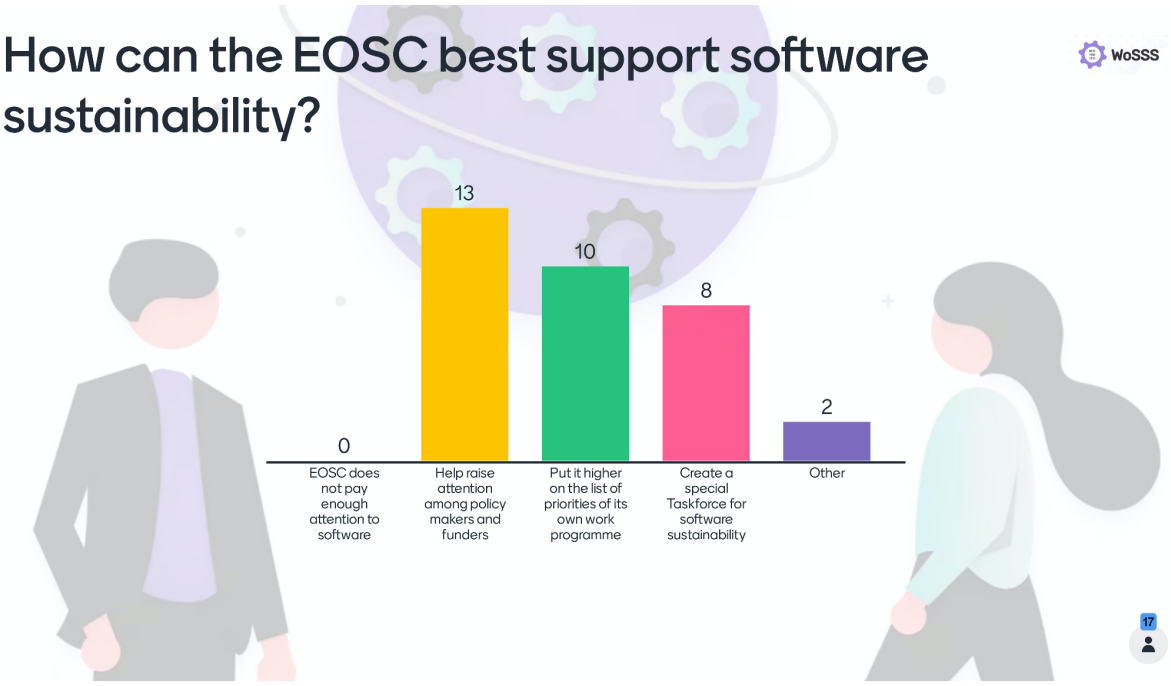
Comment: Given the previous result, the answer here was intriguing, it's clear that a need is seen for a European/international Software Sustainability infrastructure effort, perhaps this points to it being unrealistic that such an institute will exist 10 years from now. The reason also needs further investigation; this could be due the lack of belief that this could happen, all the way to a different model of how this should be done to the fact that in 10 years such an organisation may have achieved its goal and not be so relevant anymore. In any case further investigation is warranted into these intriguing results. One point which should be stressed is the definition of infrastructure for the purpose of software sustainability; section 9.2 gives further insight on this point.



Comments: Community building, promotion of standards and expertise and advice were the top three priorities as seen by respondents for any Software Sustainability Infrastructure. It's interesting to note that activities related to community building are ranked at the top, while activities related to coding were seen as the lowest priorities. Another interesting comment was that "FAIR certification" was perhaps a strong wording but having tools to validate compliance with FAIR principles automatically would be useful. Further investigation about how important these tasks to do are would help establish whether this was a case of prioritisation purely or also importance and reshaping what an infrastructure would offer.



Comments: The highest scoring options included a future European/International Software Sustainability Infrastructure were an independent organisation such as an ESFRI or ERIC or an organisation with national members. Looking at the distribution of scores it's clear to see that an organisation of national members had most respondents voting highly for this option whereas there was a stronger difference of opinion for the independent entity choice. The need for shared ownership of efforts in this space was highlighted by the lowest score going to bases such an infrastructure at an existing research data organisation. As an example, the position of DANS is that data organisations see the importance of such infrastructures existing, even though it feels that it does not necessarily have the skills to lead it. The key issues around leadership relate back to the stakeholders for software sustainability: who has ownership, accountability, and responsibility for software sustainability? A combination of contributions from research domains (which are most familiar with the tools themselves) and an overarching body (to aid coordination and consensus building) could be a good approach.

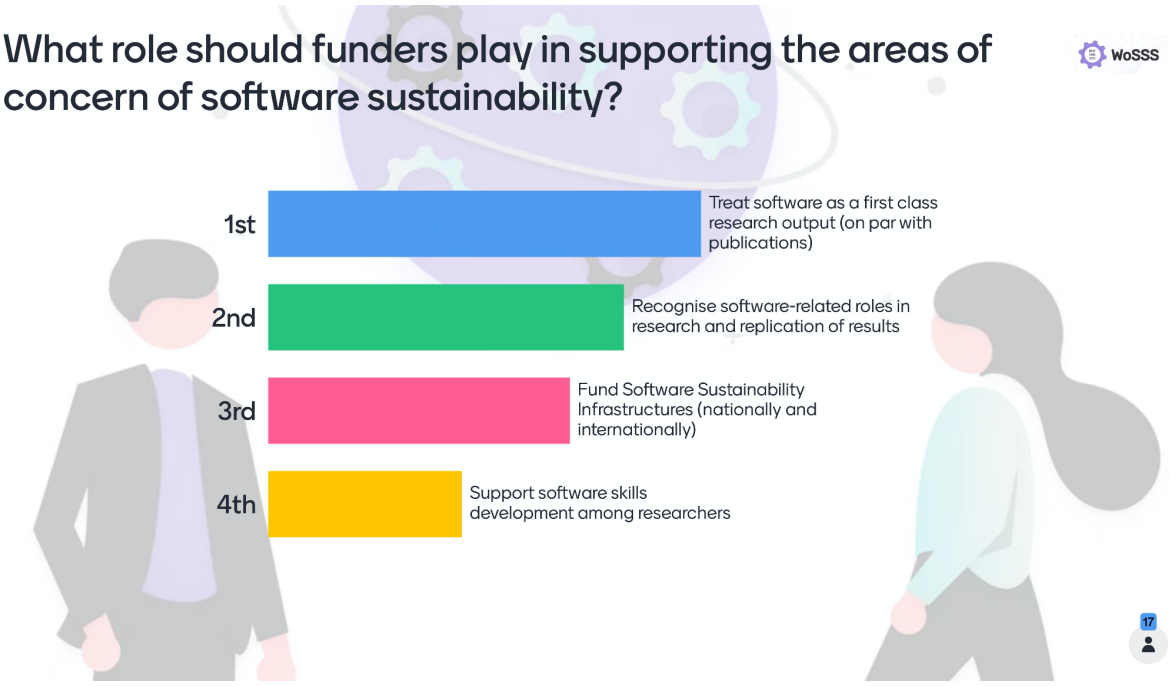


Comments: Raising attention among policy makers was clearly the most popular option. Other options would be for EOSC to have its own software sustainability and RSE strands.

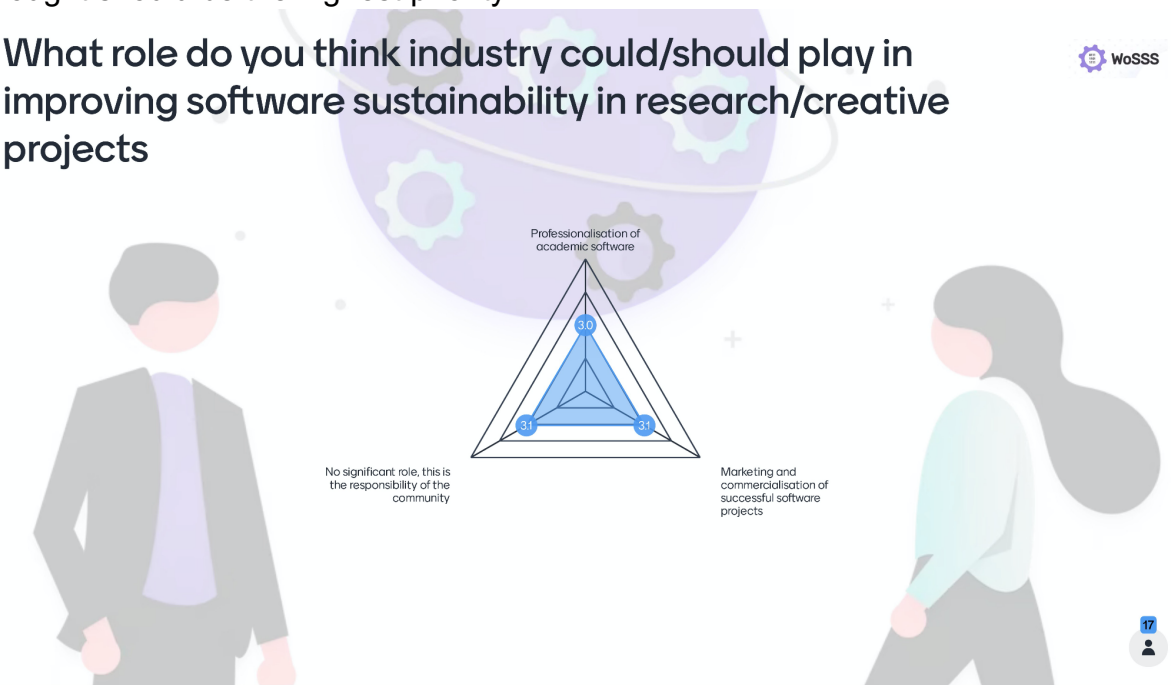
Comments: two of the most important initiatives that were mentioned as relevant to software sustainability were The Carpentries[159] and The Turing Way[160].

[159] https://carpentries.org/

[160] https://the-turing-way.netlify.app/



Attendees thought that treating software as a first class research output should have the highest level of priority in how funders can support software sustainability; reflecting the issue of recognition and the need for change in research culture around this topic. After this came the need to support the recognition of software related roles in research and replication of results, highlighting the importance of supporting Research Software Engineers, Data Stewards and others as being integral to a healthy research ecosystem. Interestingly, funding infrastructure and training came after these two and we know how much focus both have amongst the community; this really highlighted the importance being placed on the two topics which respondents thought should be the highest priority.



It's interesting to note that there was an even spread of opinion around industry's role in improving software sustainability between helping with professionalisation of research software, marketing and commercialisation of successful projects and

playing no significant role as this is the responsibility of the community. Given that the first two criteria concern industry playing a part overall, there is a positive feeling about industry being involved with research software.

10 References

1. Chue Hong NP, Katz DS, Barker M, Lamprecht AL, Martinez C, Psomopoulos FE, et al. FAIR Principles for Research Software (FAIR4RS Principles). 2022 May 24 [cited 2022 Dec 6]; Available from: https://zenodo.org/record/6623556

2. McGovern N. Digital Preservation Management Model Document | Digital Preservation Management [Internet]. [cited 2022 Nov 28]. Available from: https://dpworkshop.org/dpm-eng/workshops/management-tools/policy-framework/model-document.html

3. Internationaler Archivrat, International Council of Archives, editors. ISAD(G): General international standard archival description; adopted by the Committee on Descriptive Standards, Stockholm, Sweden, 19-22 Sept. 1999. 2. Ed. Ottawa: International Council of Archives; 2000. 94 p. (ICA Standards).

4. Hong NPC, Katz DS, Barker M, Lamprecht AL, Martinez C, Psomopoulos FE, et al. FAIR Principles for Research Software (FAIR4RS Principles). Res Data Alliance RDA. 2021 Jun;32.

5. Union PO of the E. Scholarly infrastructures for research software : report from the EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS. [Internet]. Publications Office of the European Union; 2020 [cited 2020 Dec 15]. Available from: http://op.europa.eu/en/publication-detail/-/publication/145fd0f3-3907-11eb-b27b-01aa75ed71a1/language-en

6. Wilkinson MD, Dumontier M, Aalbersberg IjJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Sci Data. 2016 Mar 15;3:160018.

7. Katz DS, Gruenpeter M, Honeyman T. Taking a fresh look at FAIR for research software. Patterns [Internet]. 2021 Mar 12 [cited 2022 Dec 6];2(3). Available from: https://www.cell.com/patterns/abstract/S2666-3899(21)00036-2

8. Reiter T, Brooks† PT, Irber† L, Joslin† SEK, Reid† CM, Scott† C, et al. Streamlining data-intensive biology with workflow systems. GigaScience. 2021 Jan 29;10(1):giaa140.

9. Maier W, Bray S, Beek M van den, Bouvier D, Coraor N, Miladi M, et al. Freely accessible ready to use global infrastructure for SARS-CoV-2 monitoring [Internet]. bioRxiv; 2021 [cited 2022 Dec 6]. p. 2021.03.25.437046. Available from: https://www.biorxiv.org/content/10.1101/2021.03.25.437046v1

10. Wratten L, Wilm A, Göke J. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. Nat Methods. 2021 Oct;18(10):1161–8.

11. Goble C, Cohen-Boulakia S, Soiland-Reyes S, Garijo D, Gil Y, Crusoe MR, et al. FAIR Computational Workflows. Data Intell. 2020 Jan 1;2(1–2):108–21.

12. Soiland-Reyes S, Sefton P, Crosas M, Castro LJ, Coppens F, Fernández JM, et al. Packaging research artefacts with RO-Crate. Data Sci. 2022 Jan 1;5(2):97–138.

13. Crusoe MR, Abeln S, Iosup A, Amstutz P, Chilton J, Tijanić N, et al. Methods included: standardizing computational reuse and portability with the Common Workflow Language. Commun ACM. 2022 May 20;65(6):54–63.

14. Khan FZ, Soiland-Reyes S, Sinnott RO, Lonie A, Goble C, Crusoe MR. Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv. GigaScience [Internet]. 2019 Nov 1 [cited 2019 Nov 4];8(11). Available from: https://academic.oup.com/gigascience/article/8/11/giz095/5611001

15. Brack P, Crowther P, Soiland-Reyes S, Owen S, Lowe D, Williams AR, et al. Ten simple rules for making a software tool workflow-ready. PLOS Comput Biol. 2022 Mar 24;18(3):e1009823.

16. Andrio P, Hospital A, Conejero J, Jordá L, Del Pino M, Codo L, et al. BioExcel Building Blocks, a software library for interoperable biomolecular simulation workflows. Sci Data. 2019 Sep 10;6(1):169.

17. Borgman CL, Bourne PE. Why it takes a village to manage and share data [Internet]. arXiv; 2022 [cited 2022 Dec 7]. Available from: http://arxiv.org/abs/2109.01694

18. Lamprecht AL, Garcia L, Kuzak M, Martinez C, Arcila R, Martin Del Pico E, et al. Towards FAIR principles for research software. Data Sci. 2019 Jan 1;Preprint(Preprint):1–23.

19. Jiménez RC, Kuzak M, Alhamdoosh M, Barker M, Batut B, Borg M, et al. Four simple recommendations to encourage best practices in research software [Internet]. F1000Research; 2017 [cited 2022 Dec 7]. Available from: https://f1000research.com/articles/6-876

20. National Academies of Sciences E, Division H and M, Services B on HC, Policy B on HS, Health R on G and P, Forum NCP, et al. Enhancing Scientific Reproducibility in Biomedical Research Through Transparent Reporting [Internet]. National Academies Press (US); 2020 [cited 2022 Dec 9]. Available from: https://www.ncbi.nlm.nih.gov/sites/books/NBK555201/

21. Goble C. Better Software, Better Research. IEEE Internet Comput. 2014 Sep;18(5):4–8.

22. Booch G. The Accidental Architecture. IEEE Softw. 2006 May;23(3):9–11.

23. Becker C, Chitchyan R, Duboc L, Easterbrook S, Penzenstadler B, Seyff N, et al. Sustainability Design and Software: The Karlskrona Manifesto. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. 2015. p. 467–76.

24. Venters C, Lau L, Griffiths M, Holmes V, Ward R, Jay C, et al. The Blind Men and the Elephant: Towards an Empirical Evaluation Framework for Software Sustainability. J Open Res Softw. 2014 Jul 9;2(1):e8.

25. Wiese I, Polato I, Pinto G. Naming the Pain in Developing Scientific Software. IEEE Softw. 2020 Jul;37(4):75–82.

26. Durdik Z, Klatt B, Koziolek H, Krogmann K, Stammel J, Weiss R. Sustainability guidelines for long-living software systems. In: 2012 28th IEEE International Conference on Software Maintenance (ICSM). 2012. p. 517–26.

27. Bass L, Clements P, Kazman R. Software Architecture in Practice. 3rd ed. Addison-Wesley Professional; 2012. 640 p.

28. Software Engineering: Principles and Practice, 3rd Edition | Wiley [Internet]. Wiley.com. [cited 2022 Dec 9]. Available from: https://www.wiley.com/en-gb/Software+Engineering%3A+Principles+and+Practice%2C+3rd+Edition-p-9780470031469

29. Northrop L. Modern Trends through an Architecture Lens. Carnegie Mellon Univ. 2018;54.

30. Salari A, Kiar G, Lewis L, Evans AC, Glatard T. File-based localization of numerical perturbations in data analysis pipelines. GigaScience. 2020 Dec 2;9(12):giaa106.

31. Ludäscher B. Computational Reproducibility vs Transparency: Is it FAIR enough? :45.

32. Bechhofer S, Buchan I, De Roure D, Missier P, Ainsworth J, Bhagat J, et al. Why linked data is not enough for scientists. Future Gener Comput Syst. 2013 Feb 1;29(2):599–611.

33. Costa J, Meirelles P, Chavez C. On the sustainability of academic software: the case of static analysis tools. In: Proceedings of the XXXII Brazilian Symposium on Software Engineering [Internet]. New York, NY, USA: Association for Computing Machinery; 2018 [cited 2022 Sep 29]. p. 202–7. (SBES '18). Available from: https://doi.org/10.1145/3266237.3266243

11 Appendix A - WoSSS21 Agenda

All times stated were in BST which is UTC+1; the afternoon sessions were timed to make it easier for those in the Americas to participate, the morning sessions were timed to make it easier for those in Oceania.

11.1 Session 1

October 6 2021, afternoon - Sustaining software in Cultural Heritage

14:15-14:30 Connect to Zoom & informal time
14:30-14:40 Opening WoSSS21 & Session 1
14:40-14:50 Icebreaker
14:50-14:55 Welcome to WoSSS21
14:55-15:20 Plenary session 1A
- Jessica Farrell - Software Sustainability as Collective Action
- Elena Colón-Marrero - Software Preservation at the Computer History Museum

15:20-15:25 Break
15:25-15:50 Plenary session 1B
- Otigbu Austine - Preserving our collective documentary heritage in bits, putting a step forward
- Patricia Falcão - Software Sustainability in the context of Software-based Art Conservation

15:50-15:55 Break
15:55-16:20 Plenary session 1C
- Morane Gruenpeter - Software as a first class research output in a FAIR ecosystem

16:20-16:25 Break (5 mins)
16:25-16:35 Introduction to breakout discussions
16:35-17:20 Break-out discussions & note taking
17:20-17:40 Networking
17:40-17:55 Brief report back from breakouts
17:55-18:00 Wrap up and close Session 1

11.2 Session 2

October 7 2021, morning - Open Science & applying the FAIR principles to software

07:45-08:00 Connect to Zoom & informal time
08:00-08:10 Welcome to WoSSS Session 2
08:10-08:20 Icebreaker
08:20-08:45 Plenary session 2A
- Nicolas M. Thiéry - A glimpse at decades of FAIR struggles and practices in computational mathematics
- Tom Honeyman - FAIR adoption

08:45-08:50 Break
08:50-09:15 Plenary session 2B
- Carole Goble - FAIR Computational Workflows
- Fakhereh (Sarah) Alidoost - Research software and beyond ESMValTool: a community and FAIR software for evaluations of Earth system models

09:15-09:20 Break
09:20-09:45 Plenary session 2C
- Fotis Psomopoulos - Developing the ELIXIR Software Management Plan for Life Sciences
- Andy Terrel - The Role of Fiscal Sponsorship in Open Software

09:45-09:50 Break
09:50-10:00 Introduction to breakout discussions
10:00-10:45 Break-out discussions & note taking
10:45-10:50 Break
10:50-11:05 Brief report back from breakouts
11:05-11:25 Networking
11:25-11:35 Wrap up and close Session 2

11.3 Session 3

October 7 2021, afternoon - Human factors and new development in preserving and sustaining research software

14:15-14:30 Connect to Zoom & informal time
14:30-14:40 Welcome to WoSSS21 Session 3
14:40-14:50 Icebreaker
14:50-15:15 Plenary session 3A
- Emma Irwin - Changing our ways: Making Inclusion a Core Feature
- Colin Venters - The Lost Architectures of Scientific Software and How to Find Them

15:15-15:20 Break
15:20-15:45 Plenary session 3B
- Vicky Rampin - Software preservation is necessary for reproducibility
- Stian Soiland-Reyes - Reproducibility; Research Objects (RO-Crate) and Common Workflow Language (CWL)

15:45-15:50 Break
15:50-16:30 Plenary session 3C
- Christina Von Flach Garcia Chavez - On the Sustainability of Academic Software in Software Engineering (due to COVID-19 was not able to present)
- Kelly Rosa Braghetto - The FLOSS Competence Center as an Enabler of High-Quality Open Research Software in Brazil
- Andrew Stewart - An Introduction to the UK Reproducibility Network

16:30-16:35 Break
16:35-16:45 Introduction to breakout discussions
16:45-17:30 Break-out discussions & note taking
17:30-17:40 Networking
17:40-17:55 Brief report back from breakouts
17:55-18:00 Wrap up and close Session 1

11.4 Session 4

October 8 2021, morning - Sustaining the community and promoting (human) infrastructures for software sustainability

07:45-08:00 Connect to Zoom & informal time
08:00-08:05 Welcome to WoSSS Session 4

08:05-08:15 Icebreaker
08:15-08:55 Plenary session 4A

- Michelle Barker - The People Roadmap: Mapping people-related initiatives in the research software community
- Martin-Hammitzsch - The fundamental part of software: the human infrastructure
- Tom Honeyman - What we are doing towards software sustainability

08:55-09:00 Break
09:00-09:40 Plenary session 4B

- Konstantinos Repanas and Ignacio Blanquer -
- Research Software Sustainability in the EOSC
- Alexander Struck - (Inter)National Community Efforts by the German Association of Research Software Engineers (de-RSE)
- Carly Strasser - Supporting the creators and maintainers of essential open source software

09:40-09:45 Break
09:45-10:45 Panel Q & A
10:45-10:50 Break
10:50-11:05 Networking
11:05-11:15 Wrap up and close Session 4
11:15-11:30 Next steps and close of WoSSS21

Last page intentionally left blank.

Photo by Ricardo Gomez Angel on Unsplash