Research Software Support

DCC Spring Training Days

This document
https://tinyurl.com/dcc-fair-morning

🗓️ Time and Day

Thursday, 13 July 2023, 9:15 - 12:45

🗺️ Location

3.5 SURF (Hoog Catharijne, Moreelsepark 48, 3511 EP Utrecht), travel instructions

🦸 Code Of Conduct

Participants are expected to follow these guidelines:

- Use welcoming and inclusive language
- Be respectful of different viewpoints and experiences
- Gracefully accept constructive criticism
- Focus on what is best for the community
- Show courtesy and respect towards other community members

If you feel like the CoC has been violated, report it to one of the trainers.

👩‍🏫Trainers

Barbara Vreede - Research Software Engineer at the Netherlands eScience Center
Mateusz Kuzak - Training Program Lead at the Netherlands eScience Center
Fenne Riemslagh - Training Coordinator at the Netherlands eScience Center

⏱️ Schedule

9:15	Intro, icebreaker, document
9:30	FAIR software slides, discussion, mapping exercise (planned software)
10:15	Break
10:30	SMP slides, SMP writing exercise (own software)
11:30	Break
11:45	FAIR vs SMP exercise (flipover/whiteboard)
12:15	Reflection, discussion of software values
12:45	End

📑 Links, Materials, Resources

- Link to Barker et al. paper on Introducing the FAIR principles for research software
- The 5 recommendations for FAIR software made by the eScience Center: https://fair-software.nl/
- A practical guide to software management plans
- You can find a SMP template here. (https://tinyurl.com/smp-templates-dcc)
- Writing A Reproducible Manuscript In R With WORCS, see Youtube tutorial

🧑‍🎨 Roll Call

Name / Pronouns (optional) / Role / Affiliation / social media

Fenne Riemslagh / she, her / Coordinator training program Netherlands eScience Center / Amsterdam
Daphne Jansen / she,her/ RDM specialist / Radboud University
Barbara Vreede / she,her / Research Software Engineer @ Netherlands eScience Center / gh: bvreede
Jet Zoon / she, her / Data Steward Research / Princess Máxima Center for Pediatric Oncology
Inge Slouwerhof / she, her/ Data Steward / Radboud University
Irene Martorelli / she,her / Research data specialist / VU Amsterdam
Andres Ramos / he, him/ Data Steward/ RIVM / gh:aframosp
Mateusz Kuzak / he, him / Team Lead Training Programme @ Netherlands eScience Center
Femmy Admiraal / she, her/Coordinator RDM team/UB Leiden- Centre for Digital Scholarship
Anneke van Houten/ she,her/ Datasteward / HAN University of Applied Sciences
Stefan Kirsch / Data Steward & Research Software Engineer @ Tilburg University
Jolien Scholten / she, her / RDM Specialist / VU Amsterdam
Katherine Marcous / she, her / Data Steward / Radboud University
Thomas Pelgrim/ Data Steward/ HAN University of Applied Sciences
Pedro Hernandez Serrano/he, him/ Coordinator, Data Stewardship Services, Maastricht University / gh: pedrohserrano
Karina Sulim/ she, her/ Data Steward/ UMCG Digital Competence Center / Groningen
Kiana Moghaddam / she, her/ Data Steward/UMCG Digital Competence Center / Groningen
Modhurita Mitra / Research Software Engineer / Utrecht University
Alice Nikuze/she,her/ Data Steward/University of Twente

🧊🔨 Icebreaker

What is your favourite superpower?

Daphne Jansen - Speak all the languages in the world (+1 Jolien
Babara - I would like to be able to stretch time, to make it as short or long as I need it to be
Fenne - talk to animals
Katherine - Teleportation
Irene - multilingual
Andres Ramos - Mindreading
Thomas Pelgrim - Flying (+1! Femmy)
Inge - Photographic memory;
Stefan - Invisibility, I would use it responsibly, except for scaring my friends 😈😉
Anneke being able to morph into different people
Jet - The Pied Piper of Hamelin type superpower.
Mateusz - telepathy
Pedro - Fill the coffee cup with a thought
Modhurita - Housework gets done as soon as I think about it

🗒️ Exercises

Exercise 1: FAIR - Mapping the principles

The FAIR4RS Principles are:

F: Software, and its associated metadata, is easy for both humans and machines to find.

- F1. Software is assigned a globally unique and persistent identifier.
  - F1.1. Components of the software representing levels of granularity are assigned distinct identifiers.
  - F1.2. Different versions of the software are assigned distinct identifiers.
- F2. Software is described with rich metadata.
- F3. Metadata clearly and explicitly include the identifier of the software they describe.
- F4. Metadata are FAIR, searchable and indexable.

A: Software, and its metadata, is retrievable via standardised protocols.

- A1. Software is retrievable by its identifier using a standardised communications protocol.
  - A1.1. The protocol is open, free, and universally implementable.
  - A1.2. The protocol allows for an authentication and authorization procedure, where necessary.
- A2. Metadata are accessible, even when the software is no longer available.

I: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards.

- I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.
- I2. Software includes qualified references to other objects.

R: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software).

R1. Software is described with a plurality of accurate and relevant attributes.
    R1.1. Software is given a clear and accessible license.
    R1.2. Software is associated with detailed provenance.
R2. Software includes qualified references to other software.
R3. Software meets domain-relevant community standards.

Three examples

The following three examples of software have been described with statements that can be mapped to the FAIR4RS Principles. With each statement, identify the principle(s) it maps to, and the (sub)facet(s) of the principle(s) it addresses. A single statement may map to multiple (sub)facets.

Example 1: Comet

Comet is a command-line tool and desktop application for tandem mass spectrometry sequence database search.

1. Comet is licensed under the Apache 2.0 open source licence.
2. Comet is registered in the bio.tools catalogue of bioinformatics tools, where it has a globally unique and persistent identifier, and rich metadata that includes the identifier and is searchable and indexable.
3. The metadata in bio.tools is independent from the Comet repository, and will stay accessible should the software itself become inaccessible.
4. The publicly accessible project repository on GitHub includes detailed information about the development of Comet.
5. The code includes dependencies to external software packages, such as Thermo Scientific's MSFileReader library.
6. Comet can be downloaded via the browser following the links provided in the metadata using https.
7. Comet uses standard data types from the proteomics domain for its input and output data that are documented in the metadata as functional annotations.

Barbara & Fenne:
1 - F2
2 -
3 -

Example 2: PureGoMe

PuReGoMe is a project aimed at understanding Dutch public sentiment during the COVID-19 outbreak period by analysing real-time Twitter data. It provides a collection of Python scripts and Jupyter notebooks for this purpose.

1. PureGoMe can be downloaded from the project repository, while metadata is accessible independently from the registry.
2. PuReGoMe's GitHub repository has detailed records of the development history.
3. The code includes dependencies to other software, such as various Python libraries.
4. PuReGoMe has a (versioned) DOI from Zenodo (
5. PuReGoMe uses the Apache 2.0 open source licence.
6. PuReGoMe uses standard file formats (e.g., CSV files) for data exchange.
7. PuReGoMe refers to other objects such as websites.
8. PureGoMe is registered in the Research Software Directory that captures the most relevant metadata, including the identifier, in searchable and indexable form.

Alice & Jet & Femmy

1. A2
2. R1.2
3. I2
4. F1.1 & F1.2
5. R1.1
6. I1
7. I2
8. F4

Daphne, Inge, Katherine

1. A2
2. R1.2
3. I2, R2
4. F1, F1.2
5. R1, R1.1
6. I1
7. I2
8. F2, F3, F4

Thomas & Anneke

1 - A2
2 - R1.2
3 - R2
4 - F1
5 - R1.1
6 - I1
7 -I2

8 F4

Example 3: gammaShiny

gammaShiny is an application that provides enhanced graphical user interfaces for the R gamma package. It is used to process in-situ gamma-ray spectrometry measurements for luminescence dating.

1. gammaShiny has been deposited in the HAL French national archive and it has a persistent globally unique identifier, with the HAL identifier of the metadata record and a SWHID, identifying specifically the software artefact on the Software Heritage universal software source code archive.
2. Thanks to the HAL platform, where a licence is mandatory, gammaShiny is under a GNU General Public Licence v3.0.
3. The archived versions of gammaShiny's source code in Software Heritage include a codemeta.json file, identifiable with a SWHID, where other metadata is available including dependencies named in CodeMeta ("softwareRequirements").

Modhurita, Karina, Kiana

1. F1, F1.1, F1.2, A1
2. R1.1
3. F3, F4

Andres & Felix
1.F1, F1.1, F1.2, A1, A2
2. R1.1
3. A1, A2, R2

Pedro & Jolien

1. F1, F1.2, F2, F3, A1.1
2. R1.1
3. A2, I2

Stefan, Irene

1. F1.2
2. R1.1
3. F2, F3, R2

Exercise 2: SMPs in practice

Step 1: Choose a software project

Choose a software project for this exercise, preferably one you are familiar with.

If you don't have a project in mind, you can use one of the following:

Xenon
ESMValTool
RS-DAT
Haddock
worcs
training infrastructure

Find the Practical Guide at tinyurl.com/SMP-guide.

Jolien, Pedro: Praat Software https://github.com/praat/praat

Femmy, Alice & Jet: RS-DAT

Step 2: Assess the management level needed

Look at the software repository or page in the Research Software Directory. Assess what level of management the software needs: low, medium, or high.

For this, use the following considerations (section 5 in the guide):

- Purpose. What is the current reason or expected end-use for developing the software?
- Reliability. The effect of software failure and/or non-maintenance on:
    - Risk of harm to self or others. This includes injury, privacy violation, bias, and inappropriate content.
    - Reputation. For example to self, institution or other.
    - Research, either your own or of others. This effect could be due to an obvious software failure ("crash") or a hidden one, for example, returning inconsistent numerical results on different operating systems.
- Maintenance. The long-term effort needed to maintain the software as long as it might be used as a standalone tool or dependency. This includes maintenance functions that can extend beyond the lifespan of the original

development project and includes fixing bugs, dependency management, operating system compatibility, and security issues.

This may not be a clear-cut decision, but that is fine. The goal is to get a sense of the level of management needed. Discuss with your group and come to a consensus.

Step 3: Fill in the SMP template

From the assessment made in Step 1, get the requirements that match the level of management needed.

You can find a template here. (https://tinyurl.com/smp-templates-dcc)

Copy the template, and fill it in for your software project.

Note that we do not expect you to complete this template entirely. A good SMP can take a while to write! The purpose of this exercise is to get familiar with the different requirements, prompts, and concepts discussed, and to encounter questions we can tackle together.

Alice, Jet, Femmy: SMP_Alice-Femmy-Jet

Katherine-Daphne-Inge: SMP RS-Dat

Jolien, Pedro: SMP-PedroJolien - not accessible (should be resolved now 🙂)

Stefan, Irene: SMP-StefanIrene

Modhurita, Karina, Kiana: SMP-artscraper

Andres & Felix: DCC software management, exo2

Exercise 3: SMP vs FAIR

Why do we care?

When thinking of the importance of good software stewardship, management, and development practice, many reasons and motivations come to mind. You may think of:

- Accuracy of results
- Reproducibility of results
- Reusability of software
- User friendliness of software
- Software quality
- Recognition and rewards for software developers
- Scholarly archive
- Software sustainability

What are the reasons and motivations that come to mind for you?

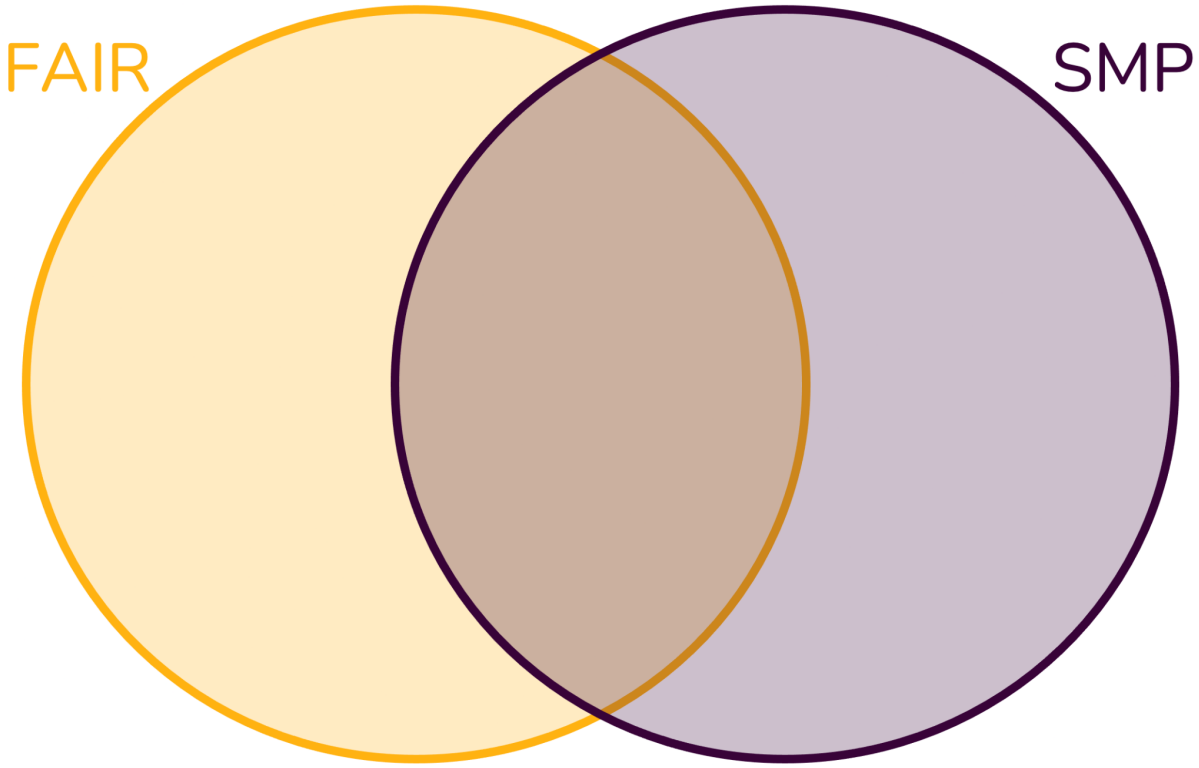
Mapping motivations

(NB: If you are not yet familiar with the FAIR principles for research software, take a look at the chapter on FAIR software before doing the following exercise.)

The motivations we listed above were important considerations for the development of Software Management Plans as well as the FAIR principles for research software. They do not match completely, however. Each has its own focus and emphasis.

In the following exercise, we will map the motivations we listed above to the ultimate goals of SMPs and the FAIR principles.

Draw a Venn diagram with two circles: SMP and FAIR:



Map the motivations listed above to the Venn diagram.

- Which motivations are covered by SMPs, which by FAIR, and which by both?
- Do you find any motivations that are not covered by either?
- What does this tell you about the relationship between SMPs and FAIR?

Jolien, Pedro (we didn't have pen for the sticky notes)

- FAIR - SMP: Interoperability of the code
- FAIR ∩ SMP: Scholarly archive, Software quality, User friendliness, Reusability of software, Reproducibility of results, Clear licencing agreements for usage
- SMP - FAIR: Software sustainability, Recognition & rewards for developers, Accuracy of results

🗒️ Collaborative Notes

Research Software

Definition of research software and distinction from software in research:

- Everything that is created with a certain research goal in mind = research software
- Everything that else (OS, packages, etc.) = software in research

- Simple examples:
    - Python isn't research software, but Python code written during research is RS
    - Excel isn't research software, but a macro developed during research is RS
    -

Zenodo Retention period: Items will be retained for the lifetime of the repository. This is currently the lifetime of the host laboratory CERN, which currently has an experimental programme defined for the **next 20 years at least**.

Licences:

- Permissive (like MIT “do what you want with it”)
- Viral (like CCs because you need to share alike
- https://choosealicense.com/
- Check your university licence policy

🧑‍🏫 Tips (things we can improve)

- The workshop was generally on a high level/policy level, which was good and useful in and of itself. I would however like to see a more practical workshop as well that talks about specific techniques and technologies (such as the Zenodo integration for GitHub). I think Research software management is still a continuum and as you said, it is up for the institution to decide what the minimum requirements are and a lot of it is still up to the researchers how far they want to go. So learning about techniques (such as the R packaging course) is I think more or at least equally helpful than getting a high-level overview. Of course you cannot cover everything in just one morning, but it would be great to quickly discuss what methods there are out there and also recommend specific best practices and ressources where to learn them.
- Working in a collaborative document with 20+ people is not ideal
- Moving back and forth in the document to check the exercise text, the FAIR principles and the place where we took notes did not work very well. Maybe keep the FAIR principles in a linked document so we can skip back and forth between browser tabs.
- For the SMP exercise we got stuck in finding out how the software and related documentation were structured (RS-DAT), as it was listed on a platform with a lot of other projects and tools. We did not really get to the part of actually answering the SMP questions. Perhaps give away the first answer about management level and move to the SMP more quickly +1
- Would have liked an exercise on setting up a software env with steps to cover the set of principles and guidelines in relation to software management.
- Miss the instructions to get into the building (probably not in the email)
- SMP is a new hot topic in Research data/ software management. Next time it would be nice to have more time dedicated to the SMP exercise and if possible provide a template which includes some instructions

🦸 Tops (things you liked)

- Barbara is clearly an expert and knows what she was talking about!
- The hands on component was great
- I love the level of the workshop and the discussions +1
- Nice that we had so many discussions!
- Well-organised workshop, with practical exercises.
- Glad that the SMP was included
- Great organisation as always!
- Enthusiastic trainer +2

- Knowledgeable instructor and assistants!
- The last exercise inspired some good discussion.
- Nice trainer with good level knowledge.
- Training vs breaktime is nice. Good to step away from the computer, let things sink in and take the opportunity to network a bit with colleagues from other institutes.
- Made me think about many different aspects of research software development
- Perfectly fitted my level of expertise - not too demanding for less software-oriented participants, but still in-depth discussions
- I learned some practical things like the fact that you should archive your software to Zenodo or a public registry and that there are automated synchronisation tools!
- Loved the part on licences, very pragmatic and informative!