Research Software Support

DCC Spring Training Days

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

Time and Day

Thursday, 13 July 2023, 13:30 - 17:00

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

13:30	Intro, icebreaker, document
13:45	FAIR software slides, discussion, mapping exercise
14:30	Break
14:45	SMP slides, SMP writing exercise
15:45	Break

16:00	FAIR vs SMP exercise
16:30	Reflection, discussion of software values
17:00	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)

🧑‍🔬 Roll Call

Name / Pronouns (optional) / Role / Affiliation / social media
Shubha Guha / she/her / Data Steward / University of Amsterdam / LinkedIn, GitHub
Renate Mattiszik /DataLibrarian/Data Steward / Saxion UAS / www.linkedin.com/in/renatemattiszik
Barbara Vreede / she, her / Research Software Engineer@ Netherlands eScience Center / gh: bvreede
Fenne Riemslagh / she, her, hers / Coordinator Training Program Netherlands eScience Center / Amsterdam
Dan Rudmann / he, him / Digital Scholarship Librarian / Leiden University / Mastodon
Luc (he/him) WUR
Stephanie van de Sandt / OS/RDM training coordinator / VU
Mercedes Beltrán (she/her)/Data Steward/Faculty of Science, Utrecht University
Mateusz Kuzak (he/him) / Team Lead Training Programme @ eScience Center
Sanne Berends (she/her)/Data manager/University of Groningen
Joost Albers/Coordinator data steward network/ Wageningen University & research
Yifei Zhao / she/her / PhD candidate / Wageningen University
Danny de Koning-van Nieuwamerongen (don't care what you call me) / RDM - WUR Library
Jacqueline Stroet / DIFFER RDM Desk and Library
Margriet Miedema/LCRDM/DCC Spring Training Days/coordinator/linked in
Maarten Hijzelendoorn/Datamanager, Consultant ICT & Research/Leiden University, Humanities

🧊🔨 Icebreaker

What superpower would you like to have?

Shubha - teleportation
Fenne - talk to animals
Barbara - the ability to bend time, and make it as long or as short as I want
Stephanie - the power over time, doing 2 things at the same time and expand time when needed to get stuff done
Luc: freeze time
Sanne - flying
Mercedes:: teleportation :)
Mateusz: cooking
Dan: mind reading
Yifei: Time traveling
Renate: Go back in time
Joost: Flying
Danny: Blink (from dishonoured)
Margriet: patience

🗒️ Exercises

Exercise 1: FAIR - Mapping the principles

The FAIR4RS Principles are:

Also find them in this document FAIR4RS Principles

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 search6.

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.

Jacqueline+Renate

1. R1.1
2. F1,F2,F3,F4
3. A2
4. R1.2
5. I2, (R2)
6. A1
7. I1,I2

Joost & Maarten
1. R1.1
2. F (F1, F2, F3, F4)
3. A2
4. R1.2
5. I2 (R2?)
6. A1
7. I1

Shubha & Dan
1. R1.1
2. F (all)
3. A2
4. R1.2?
5. I2, R2
6. A1
7. I1

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.

Jacqueline+Renate:

1.

Mercedes and Sanne:

1. A1 - A2?
2. R1.2. / R 3 / F2
3. R2
4. F1 / F1.2
5. R1.1.
6. I1 / R3
7. I2
8. F2/F3/F4

Example 3: gammaShiny

gammaShiny is an application that provides enhanced graphical user interfaces for the R gamma package8. 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").

Stephany & Luc
1 -> F1; F3; F4
2 -> R1.1
3 ->

Yifei:
1. F1, F3, A2
2. R1, A1
3. F2; R2; I2;

Margriet
1. F1;F1.2 A1
2. R1.1. F3
3. F3

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.

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.

Please share your SMP here!

Jacqueline+Renate:
https://docs.google.com/document/d/1KfKbSUfCaPFc_KVet1m44UgQZVFg1kNg_7CnsTHhlqY/edit?usp=sharing

Luc & Stephanie:
https://docs.google.com/document/d/1b8dEZH9WvVm4LEFR158xXrVISsu13IxnBI8ttm2JRkA/edit?usp=sharing

SMP Danny & Yifei

Shubha & Dan: SMP - DemoDQ

Mercedes & Sanne:

SMP worcs

Maarten & Joost:
https://docs.google.com/document/d/1TFVlgbOEUWsefpWpGTixPF-4rsDJmG1RsWBnnW3DTTA/edit?usp=sharing

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?

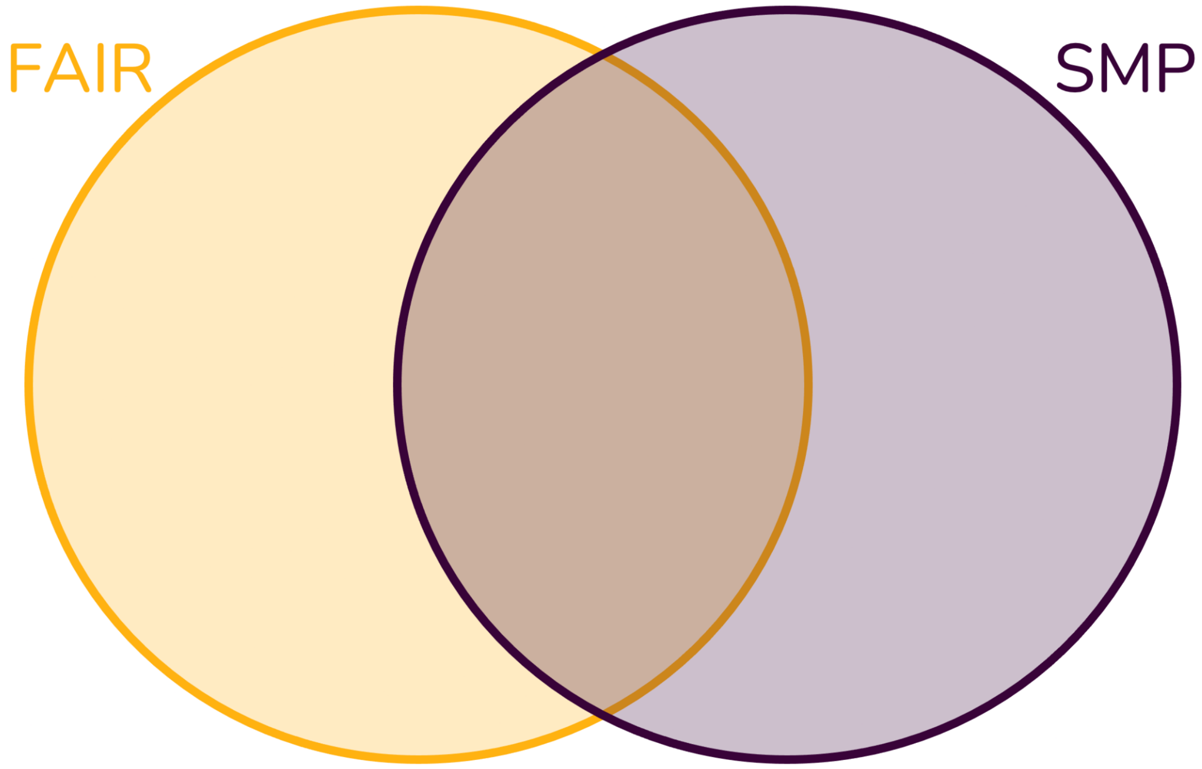
Mapping motivations

(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?

🗒️ Collaborative Notes

Intro

- Dependencies are an important aspect of software for FAIR
- Distinct from data as creatively generated, living thing

Findability

- Github is not a stable archive, use Zenodo for persistence/PID
- Community registries

Accessibility

Interoperability

- DOI/URL/clear pointing to other digital objects

Reusability

- Licensing
    - University policies needed on what open source software licensing to use because the institution/employer owns the software

FAIR is not a quality checklist - quality is not an aspect of FAIR - instead it is about usability and maintainability. Also nothing about safety and security.

- FAIR as spectrum, not switch

Software Management Plans

- Reproducibility not mentioned in the FAIR principles
- Why? Technical choices become more clear, provides time to develop objects, critical reflections on methodology
- Start with the Purpose
- Who is documentation for?
- Some questions require narratives & may be institution specific
- Add assessment rubric
- Low/Medium/High category determined in conversation with the purpose, reliability, and maintenance of software

🧑‍🏫 Tips (things we can improve)

- Add your message

🦸 Tops (things you liked)

- Add your likes here 😍
-