

Software Management Plans, basic training

Samantha Wittke, CSC – IT Center for Science

Practicalities



The webinar will be recorded, but the discussion and participants' comments will not be included in the recording.



Please keep your microphone muted during the presentation.



You can write questions in the chat at any time. Short questions may be addressed briefly between different sections, but there will be a dedicated discussion at the end of the webinar.



Content

- Research Software: Context & Basics
- Why Research Software Matters?
- Introduction to Software Management Plans
- What Does an SMP Look Like?
- Using SMPs in Practice
- Practical Guidance
- SMPs in Finland / Nordics
- Discussion





**Answer by
Zoom reaction**

Have you heard about SMPs before?

Meet Anna and Ben

Anna – researcher in environmental sciences, mostly working solo

- Small, self-contained project: her own analysis of climate patterns
- Writes scripts (R/Python) to clean and analyze her own datasets
- One postdoc occasionally reuses her scripts
- 3-year project → her own publications



Ben – PhD student in a large social science consortium

- Multi-site study on community responses to climate change, spanning several partner institutions
- Built a shared Python tool together with a couple of PhD students from other partner sites, used to code and combine interview transcripts
- Dozens of people across sites now depend on this one tool and its output database
- 3-year project → coordinated publications across all partner sites



Research Software: Context & Basics

- What is research software
- Who creates it
- How it is developed
- How it is used in research

What is research software and who creates it?

- **Code**: individual instructions that let computers perform tasks
- **Research software**: organized code, documentation, and tools that support research and produce research outputs
- Almost universal: 92% of academics use research software; 7/10 say research is impossible without it
(S.J. Hettrick et al, UK Research Software Survey 2014", [DOI:10.5281/zenodo.1183562](https://doi.org/10.5281/zenodo.1183562))
- Ranges from a short script to specialized or complex software
- Usually **created by researchers themselves**, sometimes developers /RSE or with their help
- Often **built informally** within a small team

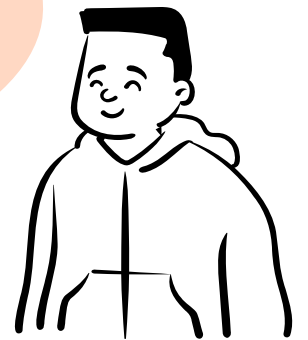
How is research software developed and used?

- **Developed iteratively** alongside research questions and discoveries
- Used by creators, collaborators, and maybe also a wider research community
- Benefits from version control, documentation, testing, and clear licensing
- **Requires planning for maintenance, sharing, and preservation** beyond the project



I add functionality when I need it and make sure it works for me.

I follow our team's development process. As we are working with various data sources the software needs to generalize.





**1-2-3
Chat!**

Mention one thing:

**What could go wrong if software isn't
documented or maintained?**

Why Research Software Matters

- Research quality and reproducibility
- Sustainability and reuse
- How software is shared and cited

Quality, reproducibility and reuse

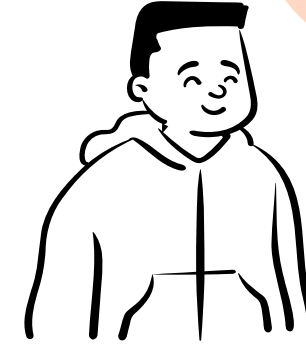
- Undocumented code can't be checked, reproduced, or reused later, often not even by its original author
- Well-organized, documented software can **save the next researcher months of effort**
- High-stakes research depends on **reliable software**: from the black hole image to AlphaFold
- Reproducible research requires **preserved, accessible, and citable software** alongside data



I try to keep things reproducible,
but my main goal is to get my
research published.

Sharing and citing research software

- Like datasets, software can be **stored, shared, and formally cited as a research output**
 - Citation.cff
 - Everyday storage: GitHub, GitLab, ...
 - Long-term, citable archiving: Zenodo, national/institutional repositories , ...
- Tools exist, but not widely known/used – seen as extra, a luxury
- Barriers to sharing: "ugly" code, fear of mistakes being found, feels too niche, ..
- Journals exist to **publish about research software**: Journal of Open Source Software (JOSS), Software-X, ...



The number of citations of our software motivates me and helps get further funds for development and maintenance!

I am not a software developer, my code shows that and I am not sure that anyone would find use for my code.



Excursion: FAIR4RS

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 standardized protocols.

- A1. Software is retrievable by its identifier using a standardized 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.

Excursion: FAIR4RS

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.



**1-2-3
Chat!**

**Mention one thing:
How could we support software planning better?**

Introduction to Software Management Plans (SMPs)

- What is a Software Management Plan
- Management levels – not one for all
- Why consider it (even if not required)
- When to consider it

What does an SMP look like?

In plain terms, the creator is asked: **where's the code, what does it do, who else touches it, is it documented, what's the license?**

Core Requirements of a Software Management Plan

Engineering focus



Documentation focus



Project management focus



SMP core requirements, grouped by focus — adapted from Netherlands eScience Center / NWO Practical Guide to SMPs (2022)

What really goes into the SMP?

Management levels:

- **Low** → a lightweight checklist is enough (purpose, version control, basic docs, license)
- **Medium** → suggest the full SMP template (adds repository, testing, citation, maintenance plan)
- **High** → flag for specialist involvement (adds a support plan and risk analysis)

Example: Anna's Software Management Plan (Low management level)



1. Please provide a brief description of your software, stating its purpose and intended audience.

A set of R scripts that clean multi-source climate datasets, run comparative analyses, and generate the figures for my publications.

Intended primarily for my own use and for my postdoc collaborator.

2. How will you manage versioning of your software?

Git repository, hosted on our institute's GitLab during the project; tagged at each major analysis milestone.

3. How will your software be documented for users? Please provide a link to the documentation if available.

A README in the repository explaining how to run each script and what input data format is expected. No separate documentation site is needed at this scale.

4. How will you document the installation requirements of your software? Please provide a link to the installation documentation if available.

An environment/lockfile listing the R package versions used, referenced from the README.

5. What type of licence will your software have?

MIT license, so my postdoc and any future collaborators can reuse the scripts freely.

Based on the sample SMP template for the "low" management level in the Netherlands eScience Center / NWO *Practical Guide to Software Management Plans* (2022), Section 6.3.1, DOI: 10.5281/zenodo.7038280.



Why consider an SMP, even if not required?

For your organization

- **Homogenisation:** comparable, reusable software, shared good practice
- **Counselling capacity:** gives support staff the background to advise well, catches risks early
- Avoids duplicated effort across projects; lets funders verify promised work was done
- Matches what some journals already expect

For a researcher

- Traceability and reusability of their own results
- Better documentation, earlier testing, **FAIR4RS compliance**
- Makes expectations clear early
- **Improves visibility, citability, and reuse** of research software
- Supports engagement with user and developer communities

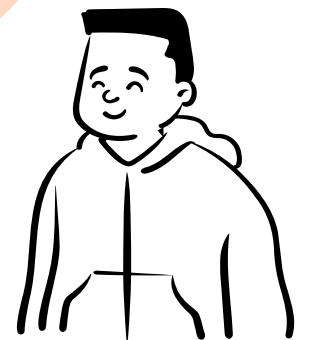
When to write an SMP?

- Best started at the **proposal stage**, alongside the DMP
- Could be funder/organizational guidance for starting a project where software development is envisioned
- Revisited at key moments: new team member, funding renewal, project end
- **Never too late to start**



I created the plan when onboarding my colleague, he got all information in one document.

Our SMP set the stage for the collaboration. Everyone knows what they are getting into and we could have the important discussion early on. E.g. which license to choose.





**1-2-3
Chat!**

A PhD student writes analysis scripts for a project and wants others to be able to reuse the software after the project ends in two years.

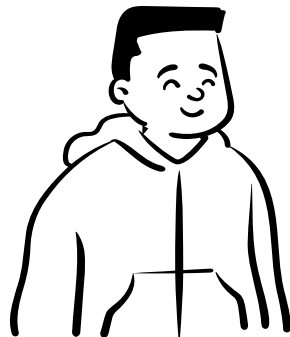
What should an SMP include?

Using SMPs in practice

- Who writes/fills it
- How to fill it/ Where to find the data?
- How to use it throughout a project
- How support staff can help

Who writes the SMP and where to find the data?

- Researchers/RSEs **creating the software** – shared responsibility with supervisors
- Filled in using **existing project documents** : proposal, DMP, documentation, README, ...
- Support staff – data stewards, RSEs – can **guide the process**
- Organizations can **provide a template** (prefilled, with extra guidance)



Our project lead could find all information needed for the SMP in our documentation and repository; we only checked the information in the end.

Potentially needed support for researcher

- Not a one-time form – **revisit at key moments**: new team member, major change in analysis, project ending
- Support staff/Organization can help : **templates, advice** on licensing/storage/DOLs, turning messy everyday practices into a clear plan, **reminders** to fill/update

Our library contacted me when I published the first article and supported me to fill out the SMP for my code. Now I have a document to plan further.



Practical Guidance

- Practical tips
- Common challenges
- Tools, templates, and examples
- Current practices, policies and trends

Practical tips and challenges

- **Start small, reuse** existing documentation, and **make the plan visible** to the whole team
- Common blockers for researchers: not knowing where to start (or that scripts even count as "software"), little time, unclear reuse rules, and turnover as students and postdocs leave
- As organization: Provide **template and guidance**, link to what they already have

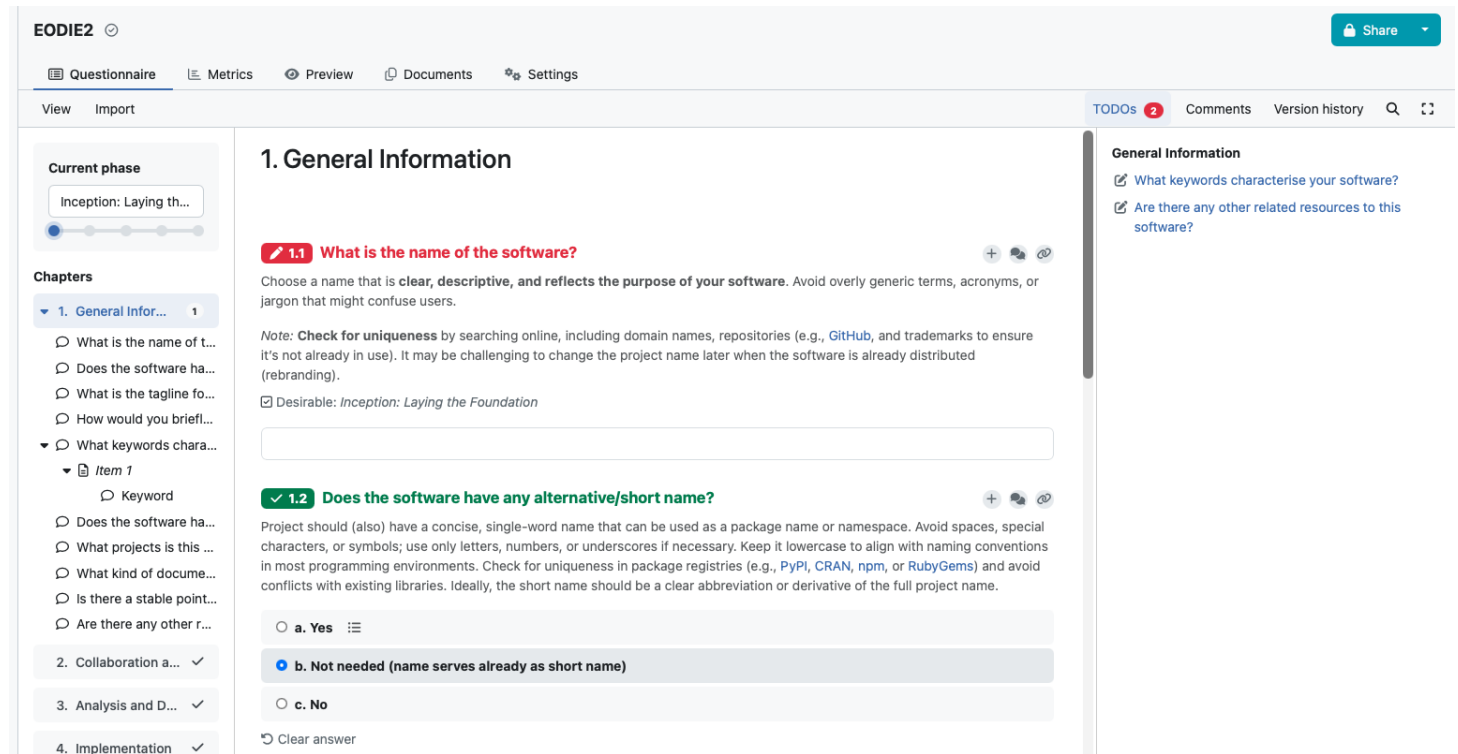


I did not feel like I had time to write an SMP, but now it saves me and my colleague time when someone has any question about the plan for my code.

ELIXIR SMP Wizard

<https://smw.dsw.elixir-europe.org/>

- Interactive questionnaire for filling SMP
- Add your own template, dropdowns for license etc
- Share with team, export
- Mark TODOs, add comments and see version history
- Wizard is in active development!



The screenshot shows the EODIE2 software questionnaire interface. The top navigation bar includes 'Questionnaire', 'Metrics', 'Preview', 'Documents', and 'Settings'. A 'Share' button is in the top right. The left sidebar shows the 'Current phase' as 'Inception: Laying th...' and a list of 'Chapters' including '1. General Information', '2. Collaboration a...', '3. Analysis and D...', and '4. Implementation'. The main content area is titled '1. General Information' and contains two questions:

1.1 What is the name of the software?
Choose a name that is **clear, descriptive, and reflects the purpose of your software**. Avoid overly generic terms, acronyms, or jargon that might confuse users.
Note: Check for uniqueness by searching online, including domain names, repositories (e.g., GitHub, and trademarks to ensure it's not already in use). It may be challenging to change the project name later when the software is already distributed (rebranding).
☒ Desirable: *Inception: Laying the Foundation*

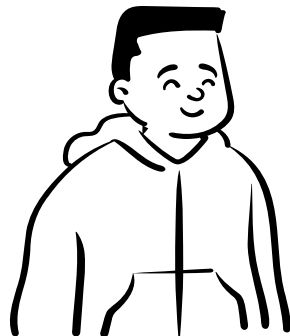
1.2 Does the software have any alternative/short name?
Project should (also) have a concise, single-word name that can be used as a package name or namespace. Avoid spaces, special characters, or symbols; use only letters, numbers, or underscores if necessary. Keep it lowercase to align with naming conventions in most programming environments. Check for uniqueness in package registries (e.g., PyPI, CRAN, npm, or RubyGems) and avoid conflicts with existing libraries. Ideally, the short name should be a clear abbreviation or derivative of the full project name.

Options for 1.2: ☐ a. Yes, ☒ b. Not needed (name serves already as short name), ☐ c. No

A 'Clear answer' button is at the bottom.

Tools, templates and Examples

- Institutional or funder-provided **SMP templates**, and guidance from organizations such as the Software Sustainability Institute, see resource slide
- GitHub/GitLab/etc for **everyday code storage**;
- Zenodo or institutional archives for **longer-term, citable archiving**
- Provide examples from other, similarly sized projects to adapt, rather than starting from a blank page



Looking at other SMPs for different sized projects helped us to decide what information is needed and to what detail it should be shared.

Current practices, policies and trends

- **Still emerging** and not standardized – practice varies a lot by field and institution
- National funders and open science policy (the FAIR principles) are increasingly extending from data to software
- Growing **recognition of software as a citable research output**, and a growing number of specialist roles (like RSEs) supporting it

Artificial Intelligence (AI) for coding and the SMP

Benefits

- Can help **draft** documentation, READMEs, a first pass at SMP answers
- **May lower the barrier** to good practices for less technical researchers
- Can **speed up filling** in a template by drafting from existing repository content

Challenges

- Code a researcher didn't fully write is code they may not fully understand – testing and documentation matter more
- **Licensing/provenance risk:** AI output can reproduce licensed code without attribution
- New question for SMPs: should "**will AI be used, and how**" be documented itself?



Hands-on workshop on tools and practices for reproducible research software development for researchers



Want to learn about git and collaborative git, on GitHub, Reproducible research, Social Coding, AI Assisted Research Software Development, tools for documentation and testing and modular code development ?

Join the [CodeRefinery workshop](#) on September 22 - 24 & September 29 - October 1

Audience: researchers of all domains, levels and preferred programming languages who write code in their research.

Location: Online, some locations may offer in-person events (check the [event page](#)), and feel encouraged to join as a team with your colleagues or [bring your own classroom](#).

The event is **free of charge**. More info and registration on the [CodeRefinery Workshop site](#).

In case of any questions, please contact support@coderefinery.org

Resources

- TU Delft: https://tu-delft-dcc.github.io/docs/software/fair_software/software_management_plan.html
- 5 min to write a maSMP: <https://doi.org/10.5281/zenodo.10374839>
- Software Management Plans - Current Concepts, Tools, and Application: <https://datascience.codata.org/articles/10.5334/dsj-2024-043>
- Software sustainability institute SMP checklist: <https://zenodo.org/records/2159713>
- The Turing Way on Software Management Plans: <https://book.the-turing-way.org/reproducible-research/rdm/rdm-smp/>
- eScience Centers practical guide to software management plans: <https://zenodo.org/records/7248877>
- EVERSE collection on SMPs: https://everse.software/RSQKit/software_management_planning
- Self assessment for FAIR research software: <https://fairsoftwarechecklist.net/v0.2/>
- 10 quick tips for making your software outlive your job: <https://arxiv.org/abs/2505.06484>

Acknowledgement

- CSC Research Data Management Competence Center
- CodeRefinery
- ELIXIR STEERS

Open discussion

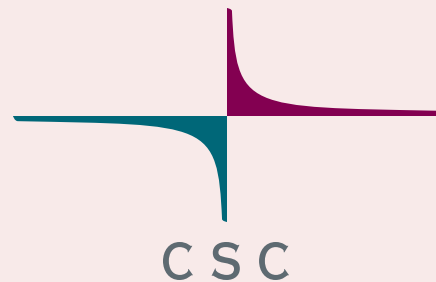
How is the SMP support organized and done in your organization?



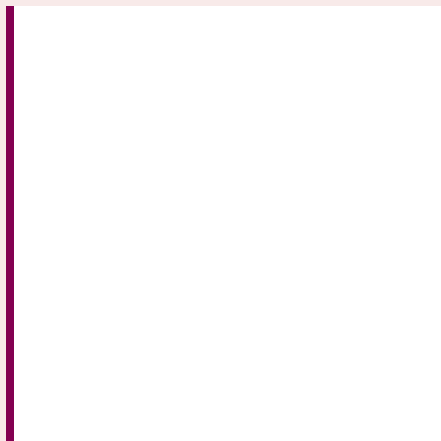


Menti

Scan the QR code or go to menti.com and enter the code. There is also a direct link in the chat!



Thank you!



Follow us

[LinkedIn](#)

[Instagram](#)

[Facebook](#)

[YouTube](#)

[csc.fi](#)