Understanding Software Management Plans

Summary

- What is a Software Management Plan?
- Who are they for?
- What's in a minimal Software Management Plan?

- Why is this important? Things change
  - Plans help manage change
  - Provide structure and baseline for prioritisation

Software Management Plan (SMP)

www.software.ac.uk

A statement of intent around how you will manage your research software

- Documents what is expected to be produced
- Defines who is responsible for what
- Details the processes used for different stages

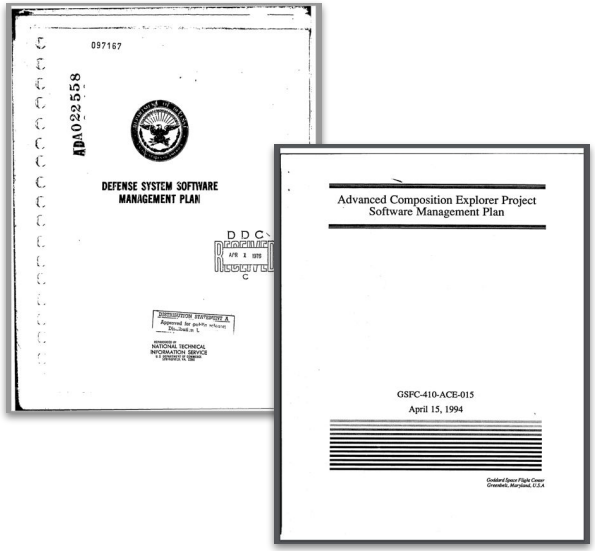
Living, evolving, versioned

Software is often an afterthought in a research project.
SMPs help to focus on software as being part of dissemination.

History of Software Management Plans

www.software.ac.uk

- As long as we've had software, we've had SMPs
- Like Data Management Plans, before the 2000s they were mostly used for large, technical complex projects
- More recently, funders have looked to SMPs and DMPs to drive reuse of research outputs



http://www.srl.caltech.edu/ACE/ASC/DATA/pdf_docs/SOFTWARE_MANAGEMENT_PLAN_ACE.pdf

Output Management Plans

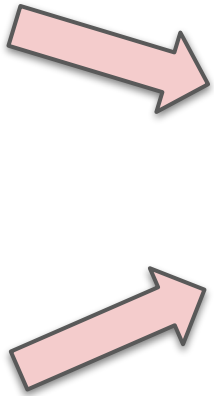
www.software.ac.uk

Data Management Plan

How will you manage the data produced on your project?
- What data created & how?
- Plans for sharing & preservation

Software Management Plan

How will you manage the software produced on your project?
- What software created & how?
- Plans for making it available (including longer term)



Output Management Plan

How will you manage the data, software, and other resources produced on your project?

Who are SMPs for?

- **PIs:** identify planned software outputs and strategy (responsibilities, processes, licenses)
- **Project Managers:** understand responsibilities and manage development and access, sharing and reuse
- **Developers:** understand what is expected to be made available to others, at what points, through what mechanisms; help improve practice and enable on-boarding
- **Funders:** encourage critical thinking about software being produced and how it will be made maximally available

A minimal Software Management Plan

- Outputs
  - What software will you be producing?
  - Who is it for (even if it's just for you)?
  - What does it do?
- Management
  - How will it be developed?
  - Who's responsible for management, quality control and releasing the software?
- Access, sharing and reuse
  - How will it be made available?
  - What license will it be released under?

My minimal Software Management Plan

- Outputs
  - A set of Python notebooks will be produced to analyse survey data collected as part of the project.
- Management
  - The notebooks will be developed in a public code repository on GitHub.
  - All team members will be responsible for checking that the code runs before committing to the repository; that files, functions and variables are appropriately named; and dependencies are recorded.
  - The PI is responsible for designating the formal release that accompanies a paper submission.
- Access, sharing and reuse
  - The code will be freely available under a BSD license from GitHub from the start of the project and formal releases deposited in Zenodo for preservation purposes.

The detail is in the detail

www.software.ac.uk

Gatto (2017): for a BBSRC TRDF grant:

All software infrastructure and statistical routines developed in this project will be submitted to the Bioconductor project (Huber et al. 2015). We will continue to follow the well established standards for packaging, versioning, documentation, updating and installation. Bioconductor will be the official distribution channel for the software, thus benefiting from existing dissemination infrastructure, support channels, user base and the developer community. In addition to biannual official releases in March and October, tested and documented development versions of the software will also be available through the dedicated Bioconductor development branch.

https://doi.org/10.3897/rio.3.e11624

ELIXIR SMP Checklist:

1) Do you provide releases of your software?
2) How do you define language-specific dependencies of your software and their version?
3) How do you capture the environment (operating system, system-level libraries) necessary to run the software (e.g. containers, conda, virtual machine)?
4) Do you provide input and output examples that can be used to reproduce the functioning of your software? If yes, where can they be found?
5) Do you state how to report bugs and/or usability problems by the software user(s)? This could be a simple sentence in the software documentation, stating an email, or an issue tracker, or a direct messaging protocol, or even a statement that bug reporting/issues are not supported.

The detail is in the detail

www.software.ac.uk

Example for a Wellcome Doctoral Studentship:

Any software or programs developed during the project period will be archived online using services such as Zenodo or Github, such that specific versions of the software are issued with Digital Object Identifiers (DOI) at the time of deposit. These will be released for public use after completion of the PhD.

Example for a Wellcome Early Career Researcher grant:

To maximise availability and impact, I will host the software on GitHub. This will allow for the creation of an immediately accessible live link, which will be used for publications, presentations and websites such as institutional and non-institutional profiles and social media. Concurrently, I will use GitHub integration to deposit software to Zenodo. Zenodo is an open-access repository where data is stored in CERN's research data centre. Depositing on Zenodo will provide the software with a unique digital object identifier (doi) and safeguard current and future access. As a further step, to aid in version control during development and pre-release, I will store software versions on the University eData repository (UBIRA) under restricted access until release. These measures will ensure that software created in this project will remain accessible for use and further development beyond the timescales of this project.
By releasing my software open-source, I will freely share the algorithms I develop, including machine learning network designs. To facilitate usability of these developments, I will concurrently provide descriptive material, beyond information that is provided in journal publications including:

- *Diagrams of overall software architecture*
- *Pseudocode*
- *Illustrative diagrams of key algorithmic steps*

These resources will maximise impact of outputs, by explaining key developments without the requirement for detailed understanding of their deployment (i.e. the actual code to implement these approaches).

Breakout 1 - Examining SMPs

www.software.ac.uk

Take a look at these example SMPs. Discuss them in your group.

Goal: to understand...

- What is an SMP?
- Why is an SMP useful?
- When is an SMP is appropriate?
- How detailed should an SMP be for different use cases?

Breakout 1 - Discussion

Take a look at these example SMPs. Discuss them in your group.

Questions

- Which sections of an SMP are clearest to you?
- Which parts are useful for each stakeholder, and why?
- Which statements could help improve practices for the project?
- How do the different example SMPs compare?
- What level of detail seems most appropriate?

Customising SMPs for funders

www.software.ac.uk

- If a funder has explicit guidance / templates, follow them!
  - Wellcome
  - Netherlands eScience Center
  - US National Science Foundation
  - European Commission Horizon Europe (FAIR)
- If there isn't guidance, look at your funders policies around publishing/dissemination, data sharing, reproducibility and impact
  - Often there are hints about what is desired
  - Be clear about what you're producing and how you're sharing it to maximise appropriate reuse

Lessons Learned

- A less detailed SMP that you use and review regularly is better than a perfect one which is created at the start of the project and never read again
  - Start small and get better
- It's important that the whole team understand the purpose of the SMP
  - An SMP is not just for a PI, for a funder
- Living, evolving, versioned
  - Used to improve practice

Reusing this material

www.software.ac.uk