Software Management Plans

JHU Open Source Programs Office

Explainer

What is a software management plan?

A document written by the developers or maintainers of a software project that describes how the project will be developed, maintained, and curated.

The goal of a software management plan (SMP) is to ensure that the software is usable and maintainable in the long term.

Why create an SMP?

An SMP makes explicit what the software does, who it is for, what the outputs are, who is responsible for the release, and how to ensure the software stays available.

It can be used to explain why new software is needed, and to verify work that went into implementation, i.e. for reporting to funders or university administration.

When should I write an SMP?

Ideally, an SMP should be drafted at the beginning of a project.

However, even for existing projects, it is valuable to create an SMP to summarize existing practices, and stimulate reflection and evaluation.

What are the core elements of an SMP?

- Purpose
- Version control
- Repository
- User documentation
- Software licensing and compatibility
- Deployment documentation
- Citation
- Developer documentation
- Testing
- Software engineering quality
- Packaging
- Maintenance
- Support
- Risk Analysis

Do I have to include all those?

Not all elements are required for all projects. For small projects - where the developer is the primary user, and the software may not be used beyond a defined period or in a different context, you can use a limited set: Purpose, Version Control, User and Deployment Documentation, and License.

The following slides walk through this subset of elements, and an SMP template for small projects is included with the transcript for this Explainer.

Purpose

What is the purpose of the software? What problem does it solve, who is the intended audience, and what are its advantages and limitations?

Example: The JHU project CoGAPS has a useful and succinct statement of purpose: https://github.com/FertigLab/CoGAPS

Further reading: Checklist for a Software Management Plan - https://doi.org/10.5281/zenodo.2159713.

Version control

A version control system is a software tool that helps track and manage changes to files over time, helping developers and users identify specific versions of the software.

Examples: GitHub, GitLab, Bitbucket. JHU affiliates have access to a GitHub Campus enterprise account at no cost through the OSPO.

Further reading: Version Control from The Turing Way - https://the-turing-way.netlify.app/reproducible-research/vcs.html

User documentation

User documentation does not need to be extensive, but should explain clearly what the software does, and how it should be used.

Example: The JHU project Python Microscope has a clean and easy to read getting started guide: https://python-microscope.org/doc/getting-started

Further reading: Tutorial template from the Good Docs project - https://gitlab.com/tgdp/templates/-/blob/main/tutorial/template_tutorial.md

Deployment documentation

Deployment documentation should explain system requirements (e.g. dependencies) for deploying the software, and instructions for installing and testing.

Examples: The JHU project Fortuna is a simple package installation, while the DSpace repository application is more complex.

Further reading: Installation template from the Good Docs project: https://gitlab.com/tgdp/templates/-/blob/main/installation-guide/guide_installation-guide.md

Software licensing and compatibility

Note which license you'll use to specify conditions of use for your software. Software licenses must be compatible with the license of external components (dependencies, libraries) that the software uses.

Example: The MIT License is an open-source license.

Further reading: JHU OSPO Licensing Information - https://ospo.library.jhu.edu/learn-grow/licensing-overview/

The content in this Explainer is adapted from:

Martinez-Ortiz, Carlos, Paula Martinez Lavanchy, Laurents Sesink, Brett G. Olivier, James Meakin, Maaike de Jong, and Maria Cruz. “Practical Guide to Software Management Plans.” Zenodo, October 27, 2022. https://doi.org/10.5281/zenodo.7248877.

Questions? Ask the JHU Open Source Programs Office

ospo@jhu.edu

https://ospo.library.jhu.edu