JOHNS HOPKINS SHERIDAN LIBRARIES | Open Source Programs Office

OSPO Explainer Transcript: Software Management Plans

URL: https://ospo.library.jhu.edu/ospo-explainers/

Slide 1: Hello, and welcome to the Johns Hopkins University Open Source Programs Office Explainer Series, a collection of bite-sized videos tackling a variety of open-source essentials. New topics are added regularly, check out ospo.library.jhu.edu to see what's coming up, or drop us a line at ospo@jhu.edu to make a request. Today's explainer? Software Management Plans.

Slide 2: What is a software management plan? A software management plan, or SMP, is 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 is to ensure that the software is usable and maintainable in the long term.

Slide 3: 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.

Slide 4: 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.

Slide 5: What are the core elements of an SMP? The core elements include purpose, version control, repository, user documentation, software licensing and compatibility, deployment documentation, citation, developer documentation, testing, software engineering quality, packaging, maintenance, support, and risk analysis.

Slide 6: Do I have to include all those? Not all elements are required for all projects. For small project, 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 including Purpose, Version Control, User and Deployment Documentation, and License.

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

Slide 7: The first element is 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?

The JHU Project CoGAPS linked here has a useful and succinct statement of purpose: https://github.com/FertigLab/CoGAPS.

For further reading, follow the link to the Checklist for a Software Management Plan: https://doi.org/10.5281/zenodo.2159713.

Slide 8: The next element is 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 of version control systems include Git, GitHub, GitLab, and Bitbucket. JHU affiliates have access to a GitHub Campus enterprise account at no cost through the OSPO.

For further reading, follow the link to the chapter on Version control from the Turing Way: https://the-turing-way.netlify.app/reproducible-research/vcs.html.

Slide 9: The next element is User documentation. User docs do not need to be extensive, but should explain clearly what the software does and how it should be used.

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

Follow the link to the Good Docs project for a great template for an end user getting started tutorial: https://gitlab.com/tgdp/templates/-/blob/main/tutorial/template_tutorial.md.

Slide 10: The next element is Deployment documentation. Deployment docs should explain system requirements for deploying the software, and instructions for installing and testing.

The JHU project Fortuna, linked here, is a simple package installation, while the DSpace repository application, also linked, is more complex.

Follow the link to the Good Docs projects for an installation documentation template: https://gitlab.com/tgdp/templates/-/blob/main/installation-guide/guide_installation-guide.md.

Slide 11: The final element is our subset is Software licensing and compatibility. In your repository, note which license you'll use to specify conditions of use for your software. Software licenses must be compatible with the licenses of external components such as dependencies and libraries that the software uses.

The MIT License is an example of an open source license.

Follow the link to the JHU OSPO to learn more about open source licensing: https://ospo.library.jhu.edu/learn-grow/licensing-overview/.

Slide 12: The content in this Explainer was adapted from the publication a Practical Guide to Software Management Plans, the full text of which is available at the link: https://doi.org/10.5281/zenodo.7248877.

Slide 13: Have any questions about what you've just learned? Ask the JHU Open Source Programs Office for more information. You can reach out to us online at ospo.library.jhu.edu or send an email to ospo@jhu.edu.