10 - 14 mars 2025

Love Data Week 2025

Université de Lorraine

“Intégrer les codes et logiciels dans un plan de gestion de données”

Jozefina Sadowska - Inria

Un événement proposé par l’atelier de la donnée ADOC Lorraine
Accompagner aux données les chercheurs et chercheuses en Lorraine
Retrouvez notre offre de services sur notre site Science Ouverte
Contact : donnees-recherche@univ-lorraine.fr

Sommaire :

- ★ 1. Contexte de la présentation
- ★ 2. Qu’est-ce qu’un logiciel ?
- ★ 3. Pourquoi rédiger un plan de gestion de logiciel ?
- ★ 4. Quels thèmes aborder dans un PGL ?
- ★ 5. Comment s’y prendre ?
- ★ 6. A venir dans DMP Opidor

1. Contexte de la présentation

Présentation basée sur les travaux du **GT-Logiciel (Réseau des ateliers de la donnée Recherche Data Gouv)**

Merci aux membres du GT et aux Administrateurs de DMP-Opidor !

source : Freepik.com

Je me présente - Jozefina Sadowska

- Coordinatrice du pôle dédié aux données de la recherche à l'Inria au sein du service national IES - Information et Édition Scientifiques
- Correspondante IES pour le centre de Nancy et membre de l'ADOC Lorraine et du Comité UL consacré à l'ouverture des codes sources et des logiciels de la recherche

  - offre de service et un accompagnement à la gestion et l'ouverture des **données de la recherche** du **code source** et des **publications scientifiques**
  - collaboration avec Software Heritage dans le projet de dépôt de logiciels dans Hal en lien avec l'archive SWH

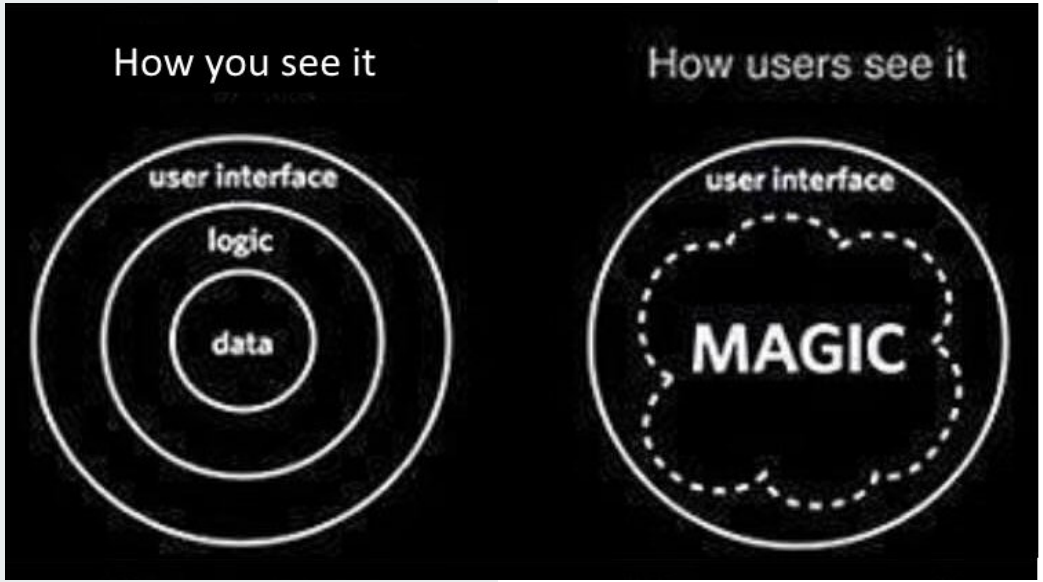
source : Freepik.com

Objectifs de l’atelier

- s’interroger sur l'intérêt de rédiger un plan de gestion de logiciel,
- découvrir sa structure et la manière de l'intégrer dans un plan de gestion de données aux côtés d’autres produits de recherche,
- présenter les premiers résultats du travail commun entre les membres du réseau des ateliers de la donnée de Recherche Data Gouv et les administrateurs de DMP Opidor pour créer un formulaire structuré dédié aux logiciels

source : undraw.co

2. Qu’est-ce qu’un logiciel ?



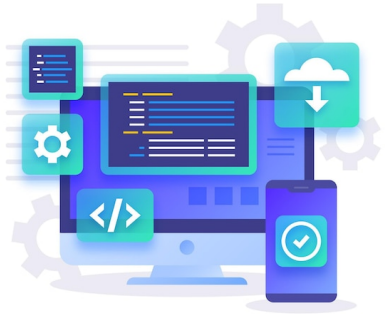
Source: https://www.reddit.com/r/ProgrammerHumor/comments/70fump/programming_is_magic/

2.1 Algorithme ? Code source ? Logiciel ?

- Algorithme : procédure par étapes, ensemble de règles à suivre pour résoudre un problème ou effectuer une tâche.

- Code source : mise en œuvre et formalisation de l’algorithme dans un langage informatique (par exemple python, C++, java...). C’est le code source qui permet d’accéder aux informations techniques et scientifiques. Il prend forme d’un ou des fichiers texte.

- Logiciel ou programme d’ordinateur : est la description, dans un ou plusieurs langages informatiques, d’un processus de traitement de données que l’on souhaite faire réaliser par un ordinateur.



On parle de logiciel lorsqu’un code a suffisamment d’utilité par lui-même pour qu’il soit considéré comme ayant une existence propre en termes de préservation, d’évolution et de diffusion.

Source : collège Code source et logiciels du COSO

2.2 Définition d’un logiciel de recherche

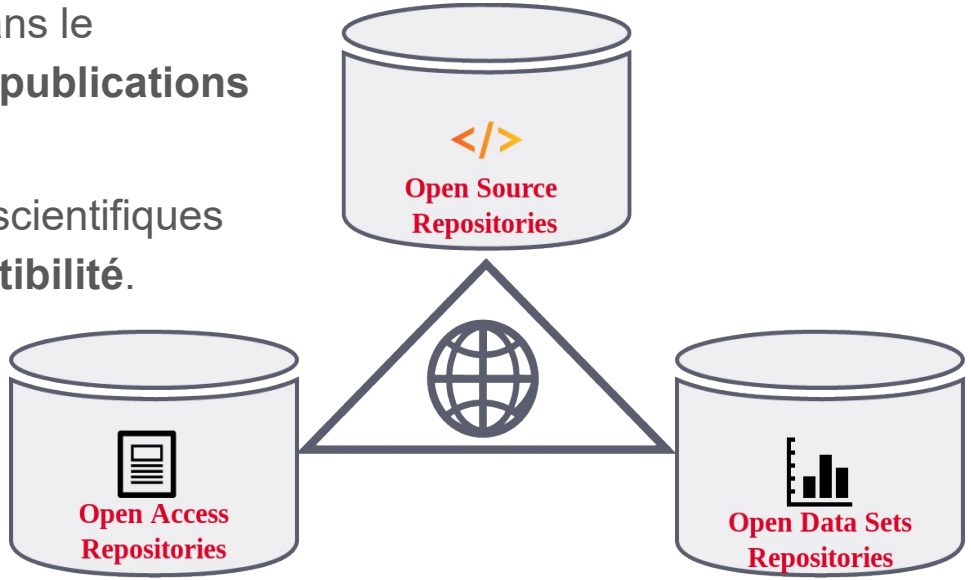
Les logiciels de recherche sont développés pour répondre à des besoins spécifiques de la science. Ils sont **conçus, maintenus, et utilisés par des scientifiques** (chercheurs et ingénieurs) et **institutions de recherche**, éventuellement dans une dimension internationale.

Ils peuvent **découler de travaux de recherche** comme ils peuvent les **favoriser**, notamment par des **publications avant/sur/autour/avec le logiciel**.

Source : collège Code source et logiciels du COSO

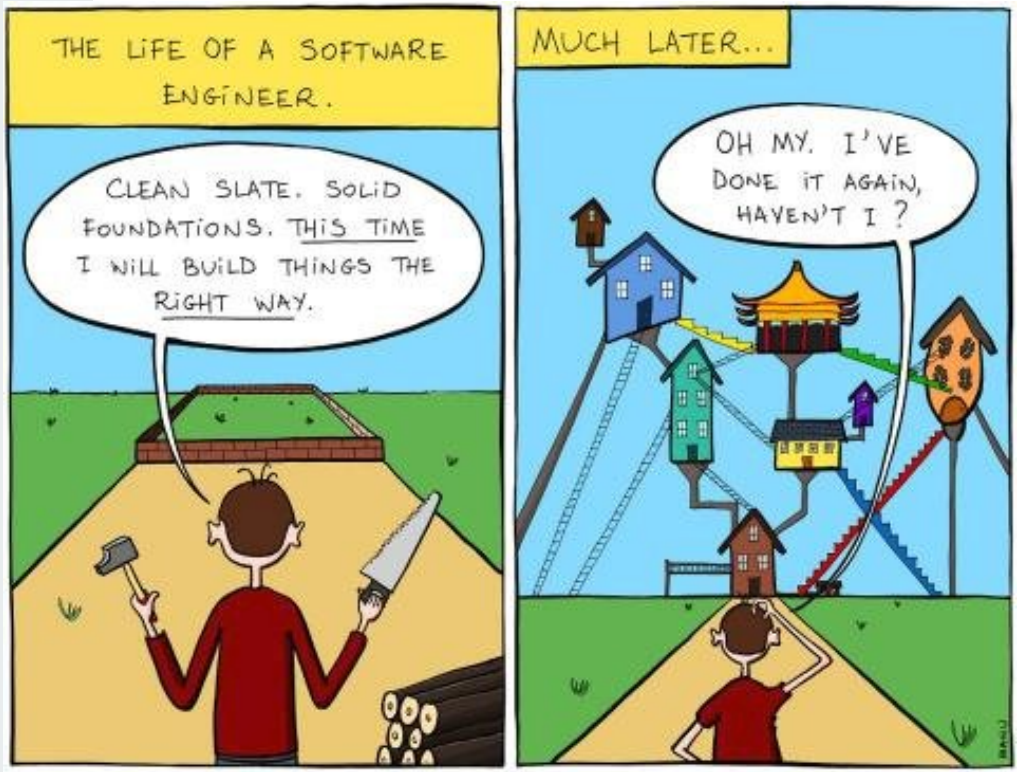
2.3 Logiciel - l’un des pilliers de la science ouverte

- Le logiciel est un élément essentiel dans le processus de recherche, au côté **des publications** et **des données**.
- **Les liens** entre ces trois productions scientifiques sont très importants pour **la reproductibilité**.
- L’enjeu :
  - Archiver
  - Décrire
  - Référencer les logiciels



Trois piliers de la Science Ouverte, Software Heritage CC-By

3. Pourquoi rédiger un plan de gestion de logiciels ?



http://www.globalgeeknews.com/wp-content/uploads/2011/12/The-Life-of-a-Software-Engineer-768x584.png

Lien : https://www.canal-u.tv/chaines/callisto/la-perte-des-donnees-de-recherche-n-arrive-t-elle-qu-aux-autres

Callisto. (2023, 5 mai). La perte des données de recherche n'arrive-t-elle qu'aux autres ?. [Vidéo]. Canal-U. https://doi.org/10.60527/455c-6b77 . (Consultée le 8 mars 2025)

Avant tout pour se poser les bonnes questions au bon moment !

- Tout comme le PGD, le plan de gestion de logiciel est un outil de gestion de projet qui permet de planifier les actions et de se prémunir contre les accidents. C'est le chercheur qui en est le premier bénéficiaire !

- Les logiciels font partie intégrante des travaux de recherche : leur bonne gestion est essentielle pour garantir **la pérennité**, **la fiabilité** et **la reproductibilité** des recherches.

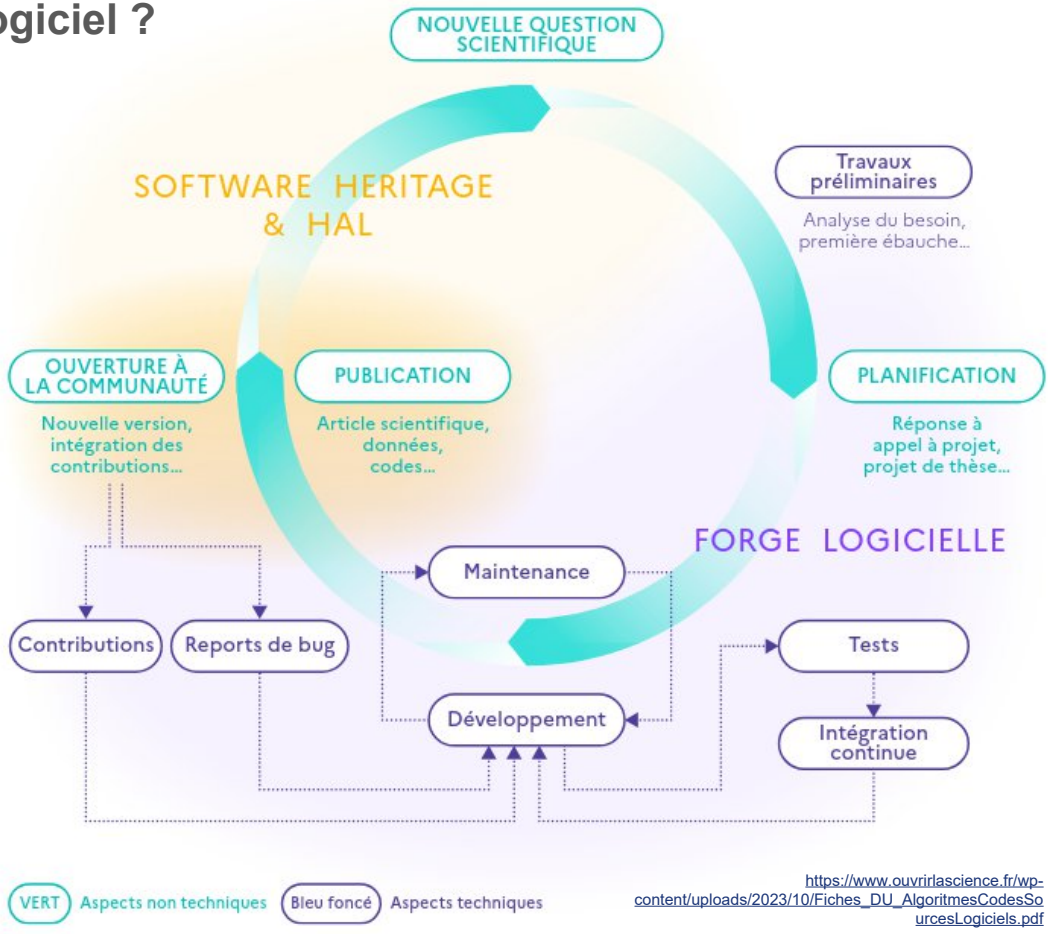
- Un PGL peut faciliter la collaboration au sein d'un projet de recherche : il assure une vision claire et partagée des logiciels utilisés.

- On peut s'attendre à ce que les agences de financement en France et en Europe demandent la réalisation des plans de gestion des logiciels au même titre que des plans de gestion de données. Ce document est déjà demandé dans certains appels d'offres au niveau international.

Qu'est-ce qu'un plan de gestion de logiciel ?

Un Plan de Gestion de Logiciel est un document évolutif décrivant les informations concernant un logiciel de la recherche, la façon dont il est conçu et développé, ses objectifs, à qui il s'adresse, les résultats attendus et obtenus, son éventuelle diffusion, les informations de propriété intellectuelle...

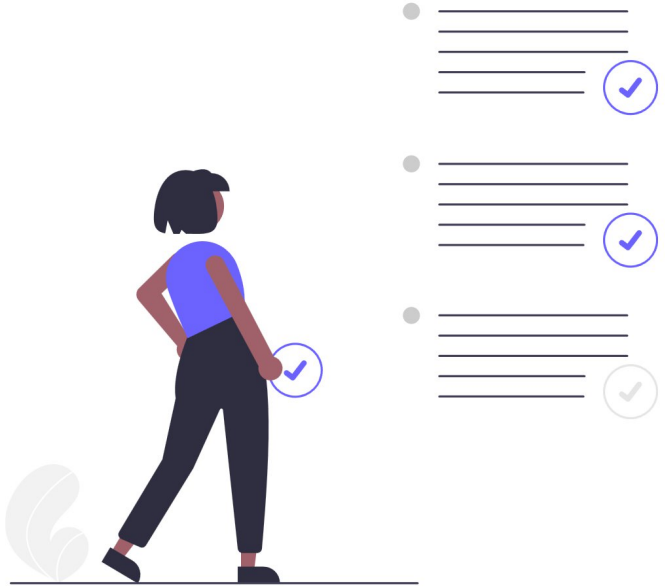
tout au long du cycle de vie de ce logiciel.



Teresa Gomez-Diaz, Genevieve Romier. Research Software Management Plan template, V3.2. 2018. ⟨hal-01802565⟩

https://www.ouvrirlascience.fr/wp-content/uploads/2023/10/Fiches_DU_AlgoritmesCodesSourcesLogiciels.pdf

4. Quels thèmes aborder dans un Plan de gestion de logiciels ?



source : undraw.co

Contenus à aborder dans un plan de gestion de logiciel de recherche

Devraient prendre en compte :

- le contexte du développement,
- le type du logiciel,
- les besoins de l'équipe de recherche, ...

Pour que le résultat reste avant tout un outil de réflexion.

Plan de gestion de logiciel
n'est pas un
Plan de développement du logiciel

Grands thèmes à aborder
Description / présentation du logiciel
Outils et environnement d'exécution
Préservation du logiciel
Questions juridiques
Valorisation scientifique
Liens avec les autres produits de recherche
+ Coûts et ressources
+ Questions environnementales

Description / présentation du logiciel

- Nom complet du logiciel
- Nom abrégé du logiciel
- Description de votre logiciel
- URL du site de présentation du logiciel ou du projet
- Date de création
  - Année de début du développement
  - Année de la première version mise en production
- Domaine d'application (principal et connexe)
- Mots-clés
- Numéro de version
- Catégorie

Conseils et bonnes pratiques :

Veiller à vérifier la disponibilité du nom, en particulier au regard de droits antérieurs de tiers en consultant la page de l'INPI.

Préciser quel est l'objectif du logiciel et indiquer à quel public potentiel (cible) le logiciel s'adresse.

Utiliser les mots-clés liés au projet, à une méthode scientifique, au programme, aux données manipulées ou produites. Utiliser de préférence des mots-clés issus de thésaurus.

Outils et environnement d'exécution (1)

- L'accès au code source est-il
  - Privé (fermé ou propriétaire)
  - Public (ouvert)
- L'utilisation d'un gestionnaire de version (Git, SVN, ...) – si oui indiquer le nom
- L'utilisation d'une forge logicielle (forge académique, Github.com, ...) si oui indiquer le nom et le lien vers le dépôt du code ou URL d'accès.
- Mise en place d'un système d'intégration continue, si oui indiquer le système choisi (Gitlab CI, Github actions, ...)

Conseils et bonnes pratiques :

L'usage d'un gestionnaire de version est particulièrement utile pour tracer tous les développements et pouvoir revenir en arrière le cas échéant. C'est la bonne pratique lorsqu'on développe du code.
L'utilisation d'une forge logicielle permet également d'assurer la sauvegarde du code sur un serveur et est un outil indispensable dès qu'on développe à plusieurs. Consulter le rapport du COSO pour avoir une liste complète des forges logicielles.
Penser à faire au minimum des tests fonctionnels pour vous assurer que les évolutions du code n'impactent pas les résultats attendus.
L'intégration continue permet de lancer ces tests automatiquement à chaque modification du code enregistrée par le gestionnaire de version.

Outils et environnement d'exécution (2)

- Documentation :
  - Indiquer le lien vers la documentation utilisateur si elle est disponible
  - Indiquer le lien vers la documentation développeur si elle existe et est disponible
- Les langages de programmation qui sont utilisés dans le logiciel
- Les systèmes d'exploitation sur lesquels le code s'exécute
- Les principales dépendances requises pour l'utilisation du logiciel

Conseils et bonnes pratiques :

Penser à rédiger au minimum un fichier readme avec les éléments permettant l'installation du code (dépendances, …) et la façon de l'utiliser
Si l'objectif est d'avoir des contributions externes, il est nécessaire de prévoir une documentation développeur.

Si c'est pertinent et applicable, penser à préciser quelle version du ou des langages de programmation vous avez utilisé (Python 2, Python 3, etc.)

Si vous souhaitez que votre logiciel soit utilisé à une échelle assez large, assurez-vous qu'il soit suffisamment portable sur les principaux
systèmes d'exploitation.

Si c'est pertinent, préciser le compilateur, l'outil de construction de programme, les bibliothèques utilisées, l'environnement d'exécution (notebook par exemple)

Préservation du logiciel

- Référencement :
  - Le signalement dans Hal ou dans un autre catalogue - l'URL du dépôt
  - L'identifiant pérenne (DOI, HalId, ...)
  - La citation du logiciel

- Archivage pérenne
  - L'archivage dans l'archive universelle Software Heritage ou dans une autre archive

Conseils et bonnes pratiques :

Les métadonnées associées au signalement du logiciel dans une archive ouverte de confiance permettent une meilleure visibilité et la possibilité
de faire le lien avec les publications et les données.
Pour faciliter le dépôt, il est possible d'utiliser l'outil Code Meta Generator qui génère un fichier codemeta.json interprété automatiquement par les entrepôts. Éviter les archives privées.
Il est conseillé de faire le lien entre la notice dans HAL et l'archivage dans Software Heritage.
La citation du logiciel est importante pour assurer le lien avec les autres productions scientifiques et créditer ses auteurs.

Software Heritage assure le moissonnage régulier des projets publics présents dans les principaux systèmes de forges et de référencement de paquets (GitHub, GitLab etc.).
Consulter la page de Software Heritage pour archiver votre logiciel grâce en particulier à la fonctionnalité Save Code Now.

Questions juridiques

- Les auteurs et éventuellement les principaux contributeurs du logiciel
  - Nom et prénom
  - Affiliation
  - ORCID
  - Rôle

- Licence(s) sous quelle(s) sera diffusé le logiciel

Conseils et bonnes pratiques :

Auteurs : liste des personnes qui ont le droit d'auteur sur le logiciel
Contributeurs : les autres collaborateurs.
Penser à créer un fichier « Auteurs ».
Il est recommandé d'avoir un identifiant auteur ORCID.

La licence doit idéalement être mise en place avant le développement du logiciel, dans le respect des éventuels contrats et conventions et avec l'accord de la tutelle concernée et de l'ensemble des co-auteurs, et de toute façon impérativement avant sa diffusion.
Elle doit aussi tenir compte des conditions de diffusion des briques logicielles préexistantes intégrées dans le code.
Avant de choisir la licence **contacter le service de valorisation de votre établissement**.
Dans le cadre de la science ouverte, préférer des licences libres. Penser à créer un fichier licence, et intégrer l'entête associé dans chaque fichier du code source. Pour plus d'informations sur les licences, consulter le site SPDX.

Valorisation scientifique et liens avec d'autres produits de recherche

- Les publications scientifiques dont le logiciel a fait l'objet
  - Identifiant pérenne p.ex. : DOI
- Les publications auxquelles le logiciel est associé
  - Identifiant pérenne p.ex. : DOI
- Liens avec d'autres produits de recherche
  - Données traitées par le logiciel
  - Workflows
  - Modèles ...

Conseils et bonnes pratiques :

Indiquer la publication qui décrit votre logiciel. Consulter la ressource Doranum qui définit ce qu'est un software paper et décrit quelques règles de rédaction.
Vous pouvez également accéder à la liste des journaux qui permettent ce type de publications.

Indiquer les publications dans lesquelles votre logiciel a été cité.

Penser à faire le lien entre votre code et les autres produits de recherche du projet.

Coûts, ressources et enjeux environnementaux

- Estimation des coûts de développement et de maintenance
- Ressources humaines (développeurs, ingénieurs, chercheurs)
- Sources de financement et stratégies pour assurer la durabilité du logiciel

- Eco-conception des logiciels
  - Utilisation d'algorithmes optimisés pour limiter la consommation de CPU et de mémoire.
  - Choix de langages et frameworks économes en énergie
  - Adoption de standards ouverts et d'une documentation complète pour favoriser la réutilisation.

5. Comment s’y prendre ?

https://pixabay.com/fr/illustrations/id%c3%a9e-d-entreprise-planification-831053 /

5.1 Utiliser un modèle dédié au Plans de gestion de logiciels

Modèles de PGD

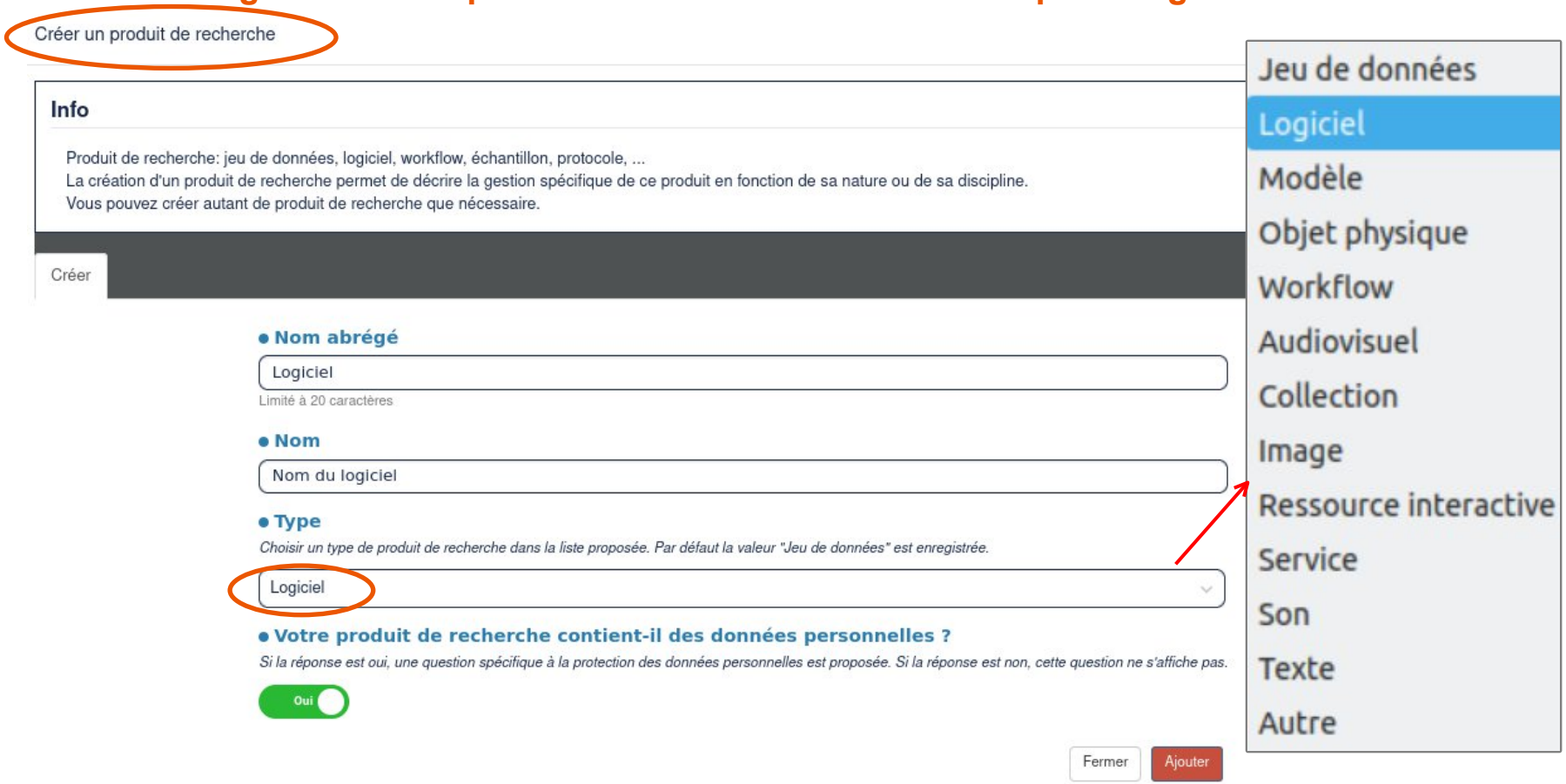
Modèles de PGD proposés par les financeurs ou par les organismes de recherche, disponibles dans DMP OPIDoR. Vous pouvez télécharger ces modèles et les recommandations associées.

software Rechercher

Nom du modèle	Nom de l'organisme	Type d'organisme	Description	Dernière mise à jour	Télécharger
Research Software Management Plan template (PRESOFT project)	PRESOFT projet	Institution	This Software Management Plan (SMP) template is provided by the PRESOFT project: Preservation for REsearch SOFTware. PRESOFT is a CNRS - IN2P3 funded project with the participation of three CNRS units: CC-IN2P3, IdGC, LIGM (2017-2018). The PRESOFT template provides a document with all the necessary elements to elaborate research software SMPs and improve thus research software preservation. PRESOFT advise you to update your SMP at each important step of your software's evolution (new version, new developers, new funding…) and to keep copies of the successive versions. Your feedback in the creation of SMPs using the PRESOFT template would be much helpful. Please send your questions or comments to presoft@univ-eiffel.fr. More information on the PRESOFT project and versions of the template in odt or pdf formats are available at http://www.france-grilles.fr/presoft-en/.	22/02/2023	
SSI Software Management Plan	Software Sustainability Institut	Institution	This Software Management Plan Template is provided by the Software Sustainability Institute based on the "*Checklist for a Software Management Plan (2016, v0.1)*". You can write a Minimal Software Management Plan or a Full Software Management Plan. Software management plans set down goals and processes that ensure software is accessible and reusable throughout a project and beyond. Research software can include both scripts and programs and can be written in languages as diverse as bash shell, R, MATLAB, Python, Java, C, C++, or Fortran; and vary in scale from 100 lines to 10,000 lines of code. Warnings: Advice and guidance provided on writing Software Management Plans, particularly that relating to intellectual property, copyright, licensing and patents, is for informational purposes only. It is not, and nor is it intended to be, legal advice. You should not act in any way on the basis of the information without seeking, where necessary, appropriate professional advice concerning your own individual circumstances. You are solely responsible for determining the appropriateness of any advice and guidance provided, and assume any risks associated with your use of this advice and guidance.	16/05/2022	

- Modèles de Software Management Plans disponibles sur DMP Opidor :
  - PRESOFT project - également sur Hal
  - Software Sustainability Institut - basé sur la checklist (version minimale ou complète)

5.2 Décrire le logiciel dans le produit de recherche au sein d'un plan de gestion de données



5.2 Décrire le logiciel dans le produit de recherche au sein d’un plan de gestion de données

5.2 Décrire le logiciel dans le produit de recherche au sein d'un plan de gestion de données

Research outputs :

1. Experimental data produced by service platforms (Dataset)
2. Databank data extracted from public data banks (Dataset)
3. Partner data shared with the project but not otherwise available in a public data bank (Dataset)
4. Calculated data produced by computer programs from other data (Dataset)
5. Source code for computer software (Software)
6. Computational workflows that automate analyses (Workflow)

Source code for computer software

1a. How will new data be collected or produced and/or how will existing data be re-used?

Software is developed by MISTIC participants of all profiles, including researchers, engineers, and students. Software development entails specification, programming, and testing of computer programs. MISTIC will encourage the adoption of institutional project-specific best practices for software quality assurance. MISTIC will use Inria's GitLab instance https://gitlab.inria.fr to host specifications, source code, and test results.

Software specifications may be informal, or may adhere to different software engineering formalisms. MISTIC will encourage software developers to include specifications in source code repositories; for example, for software projects using behavior-driven development, the feature files will be included in the source code.

Software programming involves writing source code for computer programs. MISTIC will encourage the use of GitLab tools for assisting program writing and modification for evolutive or corrective maintenance, including *issues*, *labels*, *milestones*, and *merge requests*.

Software testing improves the quality of software and confidence in its correctness by comparing the results of program executions to expected results. MISTIC will encourage the adoption of *continuous integration* practices, using GitLab CI in particular.

Exemple réalisé avec un modèle ANR sur DMP Opidor : « Microbial communities and TIC » project DMP par D.Sherman (Inria) :

– https://dmp.opidor.fr/plans/19965/export.pdf?export%5Bquestion_headings%5D=true

6. A venir dans DMP Opidor

Décrire le logiciel dans le produit de recherche dédié et adapté au sein d'un plan de gestion de données

LOGICIEL

⊕ Créer

1. Description générale du logiciel

Tout développer | Tout réduire

1.1 **Dénomination et description du logiciel**

Ceci est une aide à la saisie

2. Outils et environnement d'exécution

Tout développer | Tout réduire

2.1 **Environnement de développement**

2.2 **Documentation**

2.3 **Environnement d'exécution**

3. Préservation du logiciel

Tout développer | Tout réduire

3.1 **Archivage pérenne**

3.2 **Référencement**

4. Questions juridiques

Tout développer | Tout réduire

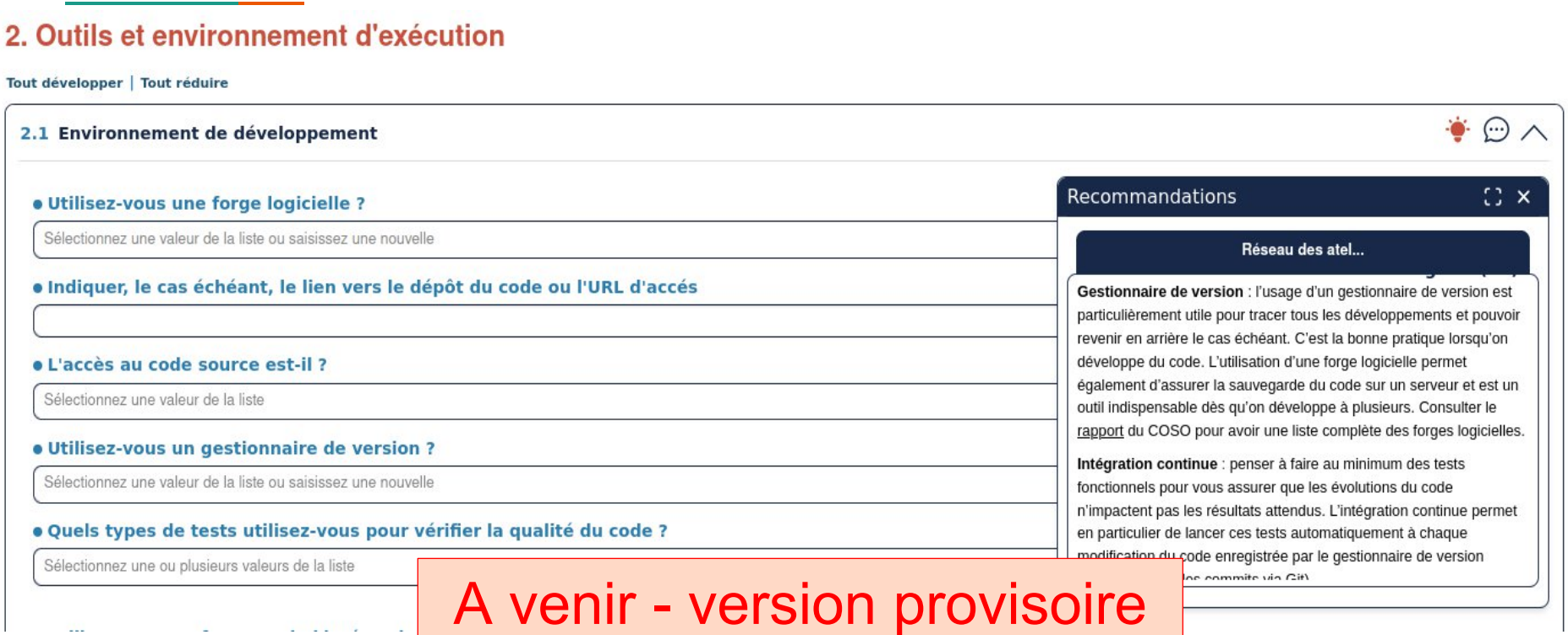
4.1 **Quels sont les aspects juridiques en lien avec la gestion du logiciel ?**

5. Valorisation scientifique

Tout développer | Tout réduire

A venir - version provisoire

Décrire le logiciel dans le produit de recherche dédié et adapté au sein d'un plan de gestion de données



Références

- Grossmann, Y.V., Lanza, G., Biernacka, K., Hasler, T. and Helbig, K. (2024) 'Software Management Plans – Current Concepts, Tools, and Application', Data Science Journal, 23(1), p. 43. Available at: https://doi.org/10.5334/dsj-2024-043

- R. Di Cosmo, M. Gruenpeter, B. Marmol, A. Monteil, L. Romary, J. Sadowska. *Curated Archiving of Research Software Artifacts: lessons learned from the French open archive*. IJDC. 2020 (10.2218/ijdc.v15i1.698). (hal-02475835)

- R. Di Cosmo, M. Gruenpeter, S. Zacchiroli. *Referencing Source Code Artifacts: a Separate Concern in Software Citation*, CiSE, IEEE, pp.1-9. 2020. (10.1109/MCSE.2019.2963148) (hal-02446202)

- R. Di Cosmo. Vers un pilier Logiciel de la Science Ouverte, Open Science Days@UGA, décembre 2022, https://osd-uga-2022.sciencesconf.org/data/pages/RobertoDiCosmo_2022_12_13_UGA_OpenDays.pdf

Références

- V. Louvet. Les codes, des données pas comme les autres ? - Plénière “Ateliers de la donnée” Grenoble, décembre 2022
- F. Pellegrini, R. Di Cosmo, L. Romary, S. Granger, S. Hodencq, J. Janik, R. Coutanson, M. Géroudet Passeport pour la science ouverte : Code et logiciels, Coso (collège Compétences et collège Code source et logiciels), 2022, 16 p. https://www.ouvrirlascience.fr/science-ouverte-codes-et-logiciels/
- Mélanie Clément-Fontaine, Roberto Di Cosmo, Bastien Guerry, Patrick Moreau, François Pellegrini. Note d’opportunité sur la valorisation des logiciels issus de la recherche. [Rapport de recherche] Comité pour la science ouverte. 2019, 6 p. ⟨hal-03606374⟩
- Teresa Gomez-Diaz, Genevieve Romier. Research Software Management Plan template, V3.2. 2018. ⟨hal-01802565⟩
- The Software Sustainability Institute. (2018). Checklist for a Software Management Plan (1.0). Zenodo. https://doi.org/10.5281/zenodo.2159713

Merci de votre attention !

Pour toute question :
donnees-recherche@univ-lorraine.fr
ou
logiciels-recherche@univ-lorraine.fr

Pour en savoir plus : https://scienceouverte.univ-lorraine.fr/