Modèle de Plan de gestion de logiciels (PGL) de l'Alliance.

2024-08-06

Digital Research Alliance of Canada | Alliance de recherche numérique du Canada

Introduction

Les logiciels sont devenus une part essentielle de la recherche moderne et sont de plus en plus souvent considérés comme un produit important de la recherche. Malgré la mise en œuvre à grande échelle du plan de gestion des données (PGD) dans les dernières années pour veiller à ce que les bonnes pratiques de gestion des données soient suivies, le Canada ne disposait pas de lignes directrices claires sur lesquelles les diverses parties prenantes pourraient s'appuyer pour planifier adéquatement le développement, la maintenance, le partage et la réutilisation d'un logiciel. Pour remédier à ce problème, l'Alliance a conçu un modèle de Plan de gestion des logiciels (PGL) en français et en anglais.

QU'EST-CE QU'UN PGL?

Un Plan de gestion des logiciels (PGL) est un document structuré qui explique comment gérer un projet logiciel de la conception à l'organisation des données en passant par la maintenance. Il fait en sorte que le logiciel demeure accessible, (ré)utilisable et durable dans le temps, qu'il soit avantageux pour les développeurs, les responsables de la maintenance et les autres personnes participant au cycle de vie du logiciel. Dans l'idéal, un PGL est créé dès le début d'un projet, mais il peut également servir à améliorer un projet existant en résumant les pratiques établies et en promouvant l'amélioration continue.

AVANTAGES D'UTILISER LE MODÈLE DE PGL DE L'ALLIANCE

Le PGL peut aider les subventionnaires à encourager et promouvoir la reconnaissance des logiciels et les bonnes pratiques (p. ex. FAIR4RS), à faciliter le changement de culture et soutenir les communautés, en s'assurant de respecter à la lettre certaines normes et politiques de gestion des logiciels. Il peut également soutenir et outiller les chercheuses et chercheurs pour qu'elles et ils développent des logiciels de meilleure qualité qui constitueront probablement une partie importante de leur portefeuille à l'avenir.

Si vous êtes **subventionnaire**, **responsable des politiques**, **administratrice ou administrateur de collection institutionnelle**, **éditrice ou éditeur**, ou prestataire de services, nous vous encourageons à utiliser ce modèle de PGL :

- comme base ou inspiration pour vos lignes directrices ou politiques;
- pour évaluer les logiciels en matière de besoins financiers et de normes de qualité (p. ex. plateformes de calcul, PaaS, IaaS, SaaS);
- pour apporter un soutien organisationnel (p. ex. infrastructure de soutien, programme de formation, personnel de soutien comme les intendantes, intendants ou professionnels des logiciels) dans le cadre de la mise en œuvre des PGL;
- pour faire en sorte que le logiciel soit géré adéquatement comme décrit dans le PGL;
- pour veiller à ce que les recherches publiées puissent être reproduites (avec les données et documents associés) à l'aide du même logiciel qui a servi à produire les résultats.as a basis or inspiration for guidelines or policies;

Si vous êtes **développeuse ou développeur logiciel**, **utilisatrice ou utilisateur**, **chercheuse ou chercheur, gestionnaire de logiciels** ou **investigatrice ou investigateur principal d'un groupe**, nous vous encourageons à utiliser ce modèle :

- pour orienter le processus de rédaction de la proposition;
- comme guide vers des pratiques exemplaires de développement logiciel;
- pour évaluer la qualité et le niveau technique d'un logiciel afin d'en maximiser les chances d'adoption;

- pour évaluer la compatibilité du logiciel avec le projet en général, les résultats attendus et les délais;
- comme liste de vérification avant la publication;
- pour évaluer la fiabilité du logiciel (p. ex. interopérabilité, flexibilité, extensibilité ou innovation) pour les projets de recherche en cours et à venir.

À l'avenir, le modèle de PGL pourrait – tout comme le PGD – être adapté aux besoins de certains types de logiciels de recherche et rendu exploitable par ordinateur (maSMP), etc.

MODÈLE DE PLAN DE GESTION DE LOGICIELS (PGL) DE L'ALLIACE

NOTE:

Le présent document est la version 1 du modèle de plan de gestion des logiciels (PGL) de l'Alliance. Il comprend 18 questions, dont 13 (dénotées par *) questions obligatoires et 5 questions facultatives.

Glossaire

Logiciel : Dans le cadre du présent modèle de PGL, le terme « logiciel » couvre deux catégories : 1) les outils logiciels (programmes, langages, bibliothèques, scripts, code de calcul, modèles, carnets de notes de laboratoire électroniques, logiciels de dépôt, outils de gestion des flux de travail, etc.); et 2) les plateformes logicielles (aussi appelées infrastructures de recherche, laboratoires scientifiques virtuels, environnements de recherche virtuels [ERV], portails scientifiques, etc.).

Versionnage des logiciels : En développement de logiciel, le versionnage désigne un schéma d'assignation d'un numéro ou d'un nom permettant aux équipes de développement de suivre les modifications apportées au code du projet en identifiant les versions successives. Ces modifications peuvent comprendre l'ajout de fonctions et la correction de bogues. Parfois, les nouvelles fonctions sont ajoutées progressivement sur plusieurs versions. Ainsi, le versionnage permet de créer et de gérer les différentes versions publiées du logiciel. De nos jours, on utilise généralement deux schémas de versionnage : 1) un numéro de version interne, qui peut parfois changer plusieurs fois dans une même journée (p. ex., numéro de contrôle de révision); et 2) une version publiée, qui change beaucoup moins souvent (gestion sémantique de version, nom de code du projet, etc.).

Contrôle des versions : Le contrôle des versions (aussi appelé contrôle de la source ou contrôle des révisions) désigne la surveillance et la gestion des modifications apportées au code au fil du temps. C'est aussi une manière d'optimiser le partage du code et sa modification par divers développeurs, sur différentes versions du logiciel, en tout temps sur le système élargi. Les systèmes de contrôle des versions (SCV), parfois appelés « outils de gestion de code source » (GCS) ou « systèmes de contrôle des révisions » (SCR), sont des outils logiciels qui servent à gérer les modifications apportées au code source au fil du temps.

Cycle de vie des versions : Le cycle de vie des versions est un ensemble de jalons qui décrivent les diverses étapes du parcours d'un logiciel, de sa conception à sa publication. On compte généralement les stades préalpha (stade au cours duquel le logiciel est encore en conception et en développement), alpha (première mise à l'essai avec des ressources internes) et bêta (essai élargi par des utilisateurs externes à l'organisation en vue de détecter et de corriger les bogues et les problèmes), de même que la version d'évaluation (pour le peaufinage et la mise à l'essai) et la version de production (aussi appelée version stable, soit la version complète destinée aux utilisateurs finaux). Une fois la version publiée, selon la méthode de publication, il peut

y avoir d'autres étapes, dont la version RTM (pour *release to manufacturing*, autrement dit la version finale et prête à être commercialisée), la disponibilité générale (étape de marketing où le produit est mis en vente), et la publication sur le Web (ou version Web; il s'agit d'utiliser Internet pour distribuer le logiciel).

But

*** Décrivez brièvement votre logiciel. Indiquez sa raison d'être, son public cible et le problème qu'il résout.**

[Saisir la réponse ici]

Exemple de réponse :

« 3D Slicer est une plateforme en code source libre servant à analyser et à visualiser les données d'imagerie médicale et de jeu de données similaires. Ce type d'environnement logiciel de pointe est utilisé quotidiennement par les chercheuses et chercheurs et par les médecins, de même que dans plusieurs contextes non médicaux.
3D Slicer se distingue parce qu'il convient à la fois aux utilisatrices et utilisateurs cliniques, aux équipes de recherche multidisciplinaires et aux architectes logiciels – le tout au sein d'une seule structure technologique et communauté d'utilisateurs. La visualisation interactive, l'enregistrement d'images, l'analyse basée sur des modèles et d'autres fonctions s'accompagnent de capacités avancées, notamment en intervention et imagerie neurologique. Ces fonctions, autrefois limitées à l'utilisation hors ligne pour des raisons techniques, sont essentielles pour les études à grande échelle qui évoluent rapidement; elles sont d'ailleurs de plus en plus intégrées à la gestion et à la prestation des soins de santé. Cette activité a été dirigée par une communauté de scientifiques et d'ingénieures et ingénieurs travaillant en science fondamentale, en science appliquée, en contexte clinique, en milieu universitaire et au privé. » Source : https://research-software-directory.org/software/3d-slicer

Conseil :

Pensez aux avantages et aux limites du logiciel. À quelle fin a-t-il été créé ou serait-il créé?

Documentation et métadonnées

Décrivez l'architecture du logiciel (ou des plateformes). Y a-t-il des principes directeurs qui ont été utilisés ou qui seront utilisés dans la conception du logiciel (ou des plateformes)?

[Saisir la réponse ici]

Exemple de réponse :

http://ireceptor.irmacs.sfu.ca/architecture

Conseil :

Vous pouvez aussi fournir un diagramme d'architecture logicielle présentant les composants matériels et logiciels du projet proposé. Distinguez clairement les parties préexistantes du système de celles qu'il faudrait ajouter ou modifier dans le cadre du projet; vous pouvez aussi montrer comment les différentes parties interagiront avec les utilisatrices et utilisateurs et les autres ressources, le cas échéant. Le modèle C4 est un exemple d'approche utile pour visualiser l'architecture d'un logiciel. Parmi les principes directeurs en conception, on compte les principes FAIR pour les logiciels de recherche (FAIR4RS), DevOps/MLOps et API RESTful.

*** Décrivez comment votre documentation aide les utilisatrices et utilisateurs et l'équipe de développement de votre logiciel (ou de vos plateformes). Veuillez fournir des liens vers les documents existants (s'il y en a) et vers les pratiques exemplaires de documentation que vous suivez.**

[Saisir la réponse ici]

Exemple de réponse :

La documentation pour compiler/monter/exécuter le code source sera créée et rendue disponible (p. ex., https://slicer.readthedocs.io/en/latest/developer_guide/build_instructions/index.html). De la documentation et des didacticiels seront aussi créés pour les utilisateurs finaux (p. ex., https://discourse.slicer.org, https://www.slicer.org).

Source : https://github.com/Slicer/Slicer?tab=readme-ov-file#readme

Conseil :

Par exemple, comment votre logiciel sera-t-il empaqueté et distribué? Veuillez fournir un lien vers les renseignements d'empaquetage (p. ex., entrée dans un registre d'empaquetage), si possible. Comment allez-vous documenter les exigences d'installation pour votre logiciel? Veuillez énumérer toutes les dépendances et tous les composants logiciels pour votre logiciel ou votre plateforme de logiciels de recherche. Veuillez fournir un lien vers la documentation d'installation, si elle est disponible, et indiquez tous les frais d'utilisation supplémentaires (licences, ressources de calcul ou de stockage, achat d'équipement supplémentaire, etc.).

*** Comment allez-vous veiller à ce que les documents soient créés ou recueillis de façon cohérente tout au long de votre projet?**

[Saisir la réponse ici]

Exemple de réponse :

Une documentation spéciale (pour les utilisatrices et utilisateurs et les développeuses et développeurs) sera créée, partagée et mise à jour, avec un contrôle des versions harmonisé au code source. S'il y a plusieurs documents, un document central sera créé pour résumer, organiser et suivre toute la documentation du projet afin de garantir la facilité d'accès, le contrôle des versions et la visibilité pour l'ensemble de l'équipe.

Conseil :

En discutant des étapes de développement, de la méthodologie et des autres renseignements pertinents pour les logiciels de recherche, vous rendez le processus de développement plus transparent, ce qui contribue à sa reproductibilité. Vous aidez aussi à uniformiser la documentation rédigée tout au long du projet de développement.

Stratégie de mise à l'essai et de déploiement

Décrivez ou résumez comment vous prévoyez de mettre à l'essai et déployer votre logiciel (ou plateforme) de manière à garantir qu'il : (1) respecte les exigences qui ont guidé sa conception et son développement; (2) est utilisable et remplit la ou les fonctions escomptées; et (3) est compatible avec l'environnement prévu.

[Saisir la réponse ici]

Exemple de réponse :

https://github.com/Slicer/Slicer/tree/main/Testing – Documentation de mise à l'essai sous Windows

(https://slicer.readthedocs.io/en/latest/developer_guide/build_instructions/windows.html#test-slicer), MacOS (https://slicer.readthedocs.io/en/latest/developer_guide/build_instructions/macos.html#test-slicer) et Linux (https://slicer.readthedocs.io/en/latest/developer_guide/build_instructions/linux.html#test-slicer)

Conseil :

Vous pouvez, par exemple, mentionner les environnements de développement et de production, les dépendances aux fonctionnalités exclusives à certains fournisseurs (p. ex., OpenShift ou Kubernetes standard). La stratégie de mise à l'essai doit décrire les essais d'unité, de fonction, d'intégration, de régression ainsi que les outils de couverture, entre autres éléments.

Contrôle et gestion des versions

*** Comment prévoyez-vous gérer le versionnage de votre logiciel? Quelles fonctions précises sont fournies par le système ou la plateforme de contrôle des versions?**

[Saisir la réponse ici]

Exemple de réponse :

Un numéro interne *de version ou de révision sera utilisé tout au long de la gestion du contrôle des versions. Nous utiliserons Git pour le contrôle des versions. Cette plateforme offre des contrôles distribués permettant de faire un suivi et de gérer les modifications au code source du logiciel, conformément au flux de travail Gitflow, et de créer des branches pour les fonctions, les versions et les correctifs. Des étiquettes (p. ex., https://github.com/aces/cbrain/tags) seront utilisées pour désigner les versions. La branche principale (maîtresse) sera protégée : avant d'y être intégré, le code devra avoir été révisé et passer des tests automatisés.*

Les versions importantes du logiciel suivront les règles de gestion sémantique de version.

Conseil :

De nos jours, on utilise généralement deux schémas de versionnage : 1) un numéro de version interne, qui peut parfois être changé plusieurs fois dans une même journée (p. ex., numéro de contrôle de révision), et 2) une version publiée, qui change beaucoup moins fréquemment (p. ex., gestion sémantique de version ou nom de code du projet). Pour gérer le versionnage interne, utilisez un système ou une plateforme de contrôle des versions ou de suivi des problèmes, comme Git, Mercurial, Apache Subversion (SVN), Beanstalk, BitBucket ou GitHub.

Quels principes directeurs/stratégies/mécanismes/pratiques/outils prévoyez-vous utiliser pour votre cycle de vie des versions?

[Saisir la réponse ici]

Exemple de réponse :

Durant la phase préalpha, notre code source sera hébergé sur GitHub pour permettre la collaboration au sein de l'équipe de développement. GitHub offre des fonctions d'intégration continue/livraison continue pour automatiser le développement, la mise à l'essai et le déploiement. Les versions importantes seront publiées sur une base trimestrielle, et des mises à jour mineures seront publiées au besoin. GitHub servira aussi à gérer le cycle de publication, notamment grâce à un environnement de préproduction pour les mises à l'essai d'avant publication (versions alpha/bêta/d'évaluation) et de stabilité (version de production). Le dépôt de code source sur GitHub sera privé en phase préalpha et rendu public à la phase bêta. Il y aura possibilité de migrer vers une instance GitLab publique ou institutionnelle à l'étape de publication.

Conseil :

Utilisez les outils comme GitHub, GitLab, BitBucket et Jenkins pour automatiser le processus de publication du logiciel. Définissez clairement le cycle de vie des versions, en proposant des mises à jour majeures planifiées et des mises à jour mineures fréquentes. Assurez-vous que les essais en environnement de préproduction

seront exhaustifs. Rédigez une documentation détaillée pour faciliter les déploiements. L'intégration continue et la livraison continue constituent une stratégie puissante offrant une assurance qualité robuste puisqu'elles automatisent le processus de développement de la conception jusqu'au déploiement. Les essais automatisés sont une facette essentielle de la livraison continue parce qu'ils garantissent que le nouveau code respecte les normes établies.

Partage et réutilisation

*** Décrivez comment vous vous publierez votre logiciel pendant et après le développement/les mises à niveau au moyen, par exemple, de dépôts comme GitHub, Zenodo et Figshare. Si ce n'est pas possible de le publier en accès libre, expliquez pourquoi.**

[Saisir la réponse ici]

Exemple de réponse :

Pendant la phase de développement, le code source sera créé, stocké, géré et partagé sur GitHub, et les versions seront contrôlées. À la fin du projet, le logiciel (ou la plateforme logicielle) sera publié et partagé sur Zenodo ou Figshare et recevra un DOI pour l'accès libre.

Conseil :

Pour rendre votre logiciel public, vous devez le stocker (soit pendant qu'il est en développement, soit une fois terminé) dans un dépôt approprié qui, idéalement, est publiquement accessible dépôt et fournit un identifiant globalement unique, pérenne et résolvable à chaque version.

*** Précisez le type de licence associé à votre logiciel – de préférence, une licence ouverte. Si la licence n'est pas ouverte, expliquez pourquoi. Envisagez une licence hybride à plusieurs composants.**

[Saisir la réponse ici]

Exemple de réponse :

Licence LGPL-3 : https://github.com/aces/cbrain#GPL-3.0-1-ov-file

Licences BSD : https://github.com/Slicer/Slicer?tab=License-1-ov-file

Conseil :

Documentez la licence et les modalités d'utilisation du logiciel, le cas échéant. Il est fortement recommandé d'envisager une licence de code source libre. Pour en savoir plus : https://opensource.org/licenses/, ou Choose an open source license *: https://choosealicense.com/. Vous pouvez aussi demander l'avis un avocat, au besoin. Si les*

exigences ou les contraintes du projet ne permettent pas d'adopter une licence ouverte, justifiez clairement pourquoi vous optez pour une licence plus restreinte. Pour les logiciels (ou plateformes) hybrides qui intègrent plusieurs composants, on peut avoir recours à plusieurs licences à la fois; dans ce cas, on adopte une structure de licence hybride au cas par cas. Veuillez préciser quelle licence sera utilisée pour le logiciel (ou la plateforme) intégré. Décrivez votre approche en matière de licence en présentant les modalités spécifiques pour l'utilisation, la modification et la redistribution du logiciel. Au besoin, vous pouvez aussi fournir votre propre fichier licence.txt.

*** Quelles mesures seront prises pour permettre à la communauté de recherche d'être au fait de l'existence de votre logiciel?**

[Saisir la réponse ici]

Exemple de réponse :

Un identifiant pérenne (p. ex., DOI) sera créé pour le logiciel (ou la plateforme), qui pourra alors être cité dans les données et les articles connexes.

Conseil :

Pour donner de la visibilité à votre logiciel au sein de la communauté scientifique, vous pouvez notamment tirer parti des réseaux universitaires, des congrès et des plateformes en ligne et publier votre documentation, des didacticiels et des articles scientifiques démontrant son utilité. Envisagez d'assigner un identifiant pérenne à votre logiciel pour faciliter la traçabilité et les citations. Si votre logiciel n'a pas d'identifiant pérenne, il est recommandé de l'inscrire auprès d'un service pour lui donner plus de visibilité et de notoriété au sein de la communauté scientifique. Pour en savoir plus, consultez l'article Introducing the FAIR Principles for research software.

*** Comment les utilisateurs pourront-ils citer votre logiciel? Si vous avez un fichier de citation de logiciel (CFF), veuillez fournir un lien pour y accéder. (https://citation-file-format.github.io/)**

[Saisir la réponse ici]

Exemple de réponse :

Un identifiant pérenne (p. ex., DOI) sera créé pour le logiciel (ou la plateforme), qui pourra alors être cité dans les données et les articles connexes. Fichier CFF pour 3 D Slicer : https://github.com/Slicer/Slicer/blob/main/CITATION.cff

Conseil :

Veuillez fournir des lignes directrices claires quant à la manière de citer votre logiciel dans les publications scientifiques, en donnant le format de citation à privilégier et tous les renseignements requis pour l'attribution. Si possible, fournissez un fichier de citation de logiciel (CFF) pour les systèmes de gestion des citations. Si vous

n'avez pas encore de fichier CFF, créez-en un rapidement pour faciliter le processus de citation. Pour en savoir plus, consultez le https://cite.research-software.org/.

Conservation et maintenance à long terme

*** Une fois créé, où et comment votre logiciel sera-t-il archivé?**

[Saisir la réponse ici]

Exemple de réponse :

Nous archiverons notre code source dans l'archive Software Heritage (https://www.softwareheritage.org/) pour la rendre accessible et la conserver à long terme. iReceptor Gateway est archivé à l'adresse suivante : https://archive.softwareheritage.org/browse/origin/directory/?origin_url=https://github.com/sfu-ireceptor/gateway.

Conseil :

L'archive Software Heritage (https://docs.softwareheritage.org/#landing-preserve) propose un guide étape par étape pour l'archivage du code source. Pour parcourir l'archive, allez à https://archive.softwareheritage.org/.

*** Comment prévoyez-vous assurer la maintenance à long terme de votre logiciel?**

[Saisir la réponse ici]

Exemple de réponse :

La maintenance à long terme de notre logiciel s'effectuera au moyen de mises à jour régulières et de pratiques d'archive. Nous utiliserons un dépôt dédié sur des plateformes comme GitHub et GitLab pour stocker les différentes versions et les documents connexes. Le code source d'iReceptor est conservé sur GitHub : https://github.com/sfu-ireceptor/gateway; les versions importantes se trouvent à l'adresse suivante : https://github.com/sfu-ireceptor/gateway/releases.

Conseil :

Assurez la pérennité de votre logiciel en préparant des stratégies de maintenance à long terme. Trouvez des solutions de versionnage, de documentation et d'archivage pour préserver l'intégrité et l'accessibilité de votre logiciel pour l'avenir.

Si vous utilisez un composant, une plateforme ou un progiciel tiers, comment allez-vous gérer les mises à niveau et les correctifs?

[Saisir la réponse ici]

Exemple de réponse :

Pour les mises à niveau et les correctifs des progiciels tiers, nous allons surveiller les mises à jour des fournisseurs et prioriser les correctifs essentiels (qui éliminent les bogues et contribuent à la gestion des dépendances) ainsi que les nouvelles fonctionnalités. Les mises à jour seront testées pour la compatibilité et les éventuels problèmes de sécurité, et les procédures de déploiement seront consignées, au besoin. Les mises à niveau et les correctifs seront annoncés à la communauté, sur le site Web du code source et dans les documents connexes.

Conseil :

Les logiciels évoluent sans cesse. Que ferez-vous si l'un des composants (surtout un progiciel tiers) utilisés par votre logiciel (ou votre plateforme de logiciels de recherche) est mis à jour pour vous assurer que ce dernier demeure fonctionnel?

Soutien aux utilisatrices et utilisateurs

*** Veuillez fournir vos plans d'activités de soutien (exploitation de la plateforme, soutien aux utilisatrices et utilisateurs, extension/ajout de fonctionnalités, facilitation de l'adoption par de nouvelles équipes de recherche, etc.) une fois le logiciel (ou la plateforme de logiciels de recherche) développé. Décrivez la formation, le personnel de soutien et les canaux de communication prévus.**

[Saisir la réponse ici]

Exemple de réponse :

Page Web du logiciel	*https://www.slicer.org/*
Télécharger 3D Slicer	*https://download.slicer.org/*
Documents	*https://slicer.readthedocs.io/en/latest/*
Dépôt GitHub	*https://github.com/Slicer/Slicer*
Instructions	*https://slicer.readthedocs.io/en/latest/developer_guide/build_instructions/index.html*

Didacticiels	*https://www.slicer.org/wiki/Documentation/Nightly/Training*
Forum (annonces communautaires et soutien)	*https://discourse.slicer.org* *https://twitter.com/3DSlicerApp*

Conseil :

Il est essentiel d'avoir un plan pour les activités de soutien aux utilisatrices et utilisateurs afin de garantir leur efficacité. On peut notamment planifier des procédures d'exploitation de la plateforme (surveiller la performance du système, veiller à l'intégrité des données, mettre en œuvre des processus pour la maintenance continue du logiciel…). En outre, il peut être envisagé de créer des canaux de communication pour que les utilisatrices et utilisateurs puissent demander de l'aide et signaler des problèmes, et le personnel peut se rendre disponible pour répondre aux questions et résoudre les problèmes. Il faut aussi envisager des programmes de formation sur le logiciel.

Responsabilités et ressources

Quelles ressources seront nécessaires à la mise en œuvre de votre plan de gestion du logiciel? À combien estimez-vous le coût global de la gestion du logiciel?

[Saisir la réponse ici]

Exemple de réponse :

Le développement du logiciel nécessitera xx ETP pour la programmation en Python, avec un salaire annuel minimal de xxx $ CA pendant x ans. Le projet nécessite l'utilisation du logiciel commercial MATLAB, dont la licence coûte yyy $ CA par année, pendant y années. Le total est estimé à yyyy $ CA pendant y années.

Conseil :

Les ressources peuvent comprendre le personnel (membres de l'équipe proposée pour le logiciel, etc.), les coûts associés aux documents, à la formation, au soutien et aux outils de contrôle des versions, de gestion des documents et de communication. Les coûts estimés peuvent refléter le salaire du personnel, les licences des logiciels et les autres dépenses, comme le matériel de formation et les services-conseils externes. En outre, prévoyez les coûts de maintenance continue et de soutien au-delà du cycle de financement.

*** Qu'adviendra-t-il des responsabilités relatives aux activités de gestion du logiciel si le personnel en assurant la supervision, notamment la chercheuse principale ou le chercheur principal, vient à changer? Pensez au pendant et au après.**

[Saisir la réponse ici]

Exemple de réponse :

Habituellement, la chercheuse principale ou le chercheur principal est la personne-ressource qui voit à la gestion du logiciel pendant et après le projet. Si cette personne change en cours de route, le bailleur de fonds en sera informé et les documents seront mis à jour. Si cette personne change après le projet, les documents seront mis à jour et la communauté sera informée.

Conseil :

Comme la personne responsable de superviser le logiciel du projet peut changer, il est important d'avoir des protocoles clairs pour assurer la continuité des activités de gestion. Durant le projet, il faut désigner une personne remplaçante et consigner les processus aux fins de transition. Une fois le projet terminé, il faut disposer d'une documentation complète et envisager des ateliers de formation pour les nouveaux membres de l'équipe.

Autres enjeux

*** Décrivez les principaux facteurs externes dont l'équipe de développement et les utilisateurs doivent tenir compte (renseignements ou enjeux de sécurité, etc.).**

[Saisir la réponse ici]

Exemple de réponse :

Au moment de développer un dépôt de partage des données, l'équipe logicielle mettra en œuvre assez de mesures de cybersécurité pour prévenir les vulnérabilités de données de la plateforme.

Conseil :

Assurez-vous de mener une évaluation du risque de sécurité exhaustive pour vérifier que le logiciel incorpore des mesures de sécurité robustes et se protège des menaces externes. Pensez aussi aux autres facteurs qui peuvent influer sur votre logiciel, comme les biais potentiels des algorithmes, la conformité aux politiques de respect de la vie privée, les questions de sécurité, les exigences de fiabilité, la transférabilité ou la dépendance à un fournisseur.

Date and Signature

Les autrices et auteurs du présent document veilleront à ce que ce plan de gestion des logiciels soit appliqué conformément à ce qui est décrit ci-dessus.

Nom : [Saisir ici]

Affiliation : [Saisir ici]

Date : [Saisir ici]

Signature : [Saisir ici]