Forges de l'Enseignement supérieur et de la Recherche - Définition, usages, limitations rencontrées et analyse des besoins

Daniel Le Berre, Jean-Yves Jeannas, Roberto Di Cosmo, François Pellegrini

▶ **To cite this version:**

Daniel Le Berre, Jean-Yves Jeannas, Roberto Di Cosmo, François Pellegrini. Forges de l'Enseignement supérieur et de la Recherche - Définition, usages, limitations rencontrées et analyse des besoins. Comité pour la science ouverte. 2023. ⟨hal-04098702v8⟩

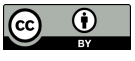
HAL Id: hal-04098702

https://hal-lara.archives-ouvertes.fr/hal-04098702v8

Submitted on 22 Jun 2026

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

Ouvrir la science !

Forges de l'Enseignement supérieur et de la Recherche

Définition, usages, limitations rencontrées et analyse des besoins

Collège Codes sources et logiciels du Comité pour la science ouverte

Mai 2023

Forges de l'Enseignement supérieur et de la Recherche

Définition, usages, limitations rencontrées et analyse des besoins

---

Collège codes sources et logiciels du Comité pour la science ouverte

---

GT 2: Outils et bonnes pratiques techniques et sociales

Daniel LE BERRE (co-pilote GT2)
Université d'Artois/CNRS

Jean-Yves JEANNAS (co-pilote GT2)
Université de Lille/AFUL

Roberto DI COSMO (co-pilote du collège)
Inria/Université Paris Cité

François PELLEGRINI (co-pilote du collège)
Université de Bordeaux/CNRS/CNIL

Mai 2023

DOI : 10.52949/34

Conception graphique : opixido

Except where otherwise noted, this work is licensed under
https://creativecommons.org/licenses/by-nd/4.0/deed.fr

Résumé

Lancée en 1999, la première forge logicielle, SourceForge, a été conçue pour permettre aux développeurs de logiciels libres de construire leurs logiciels de manière collaborative et de les diffuser auprès de leurs utilisateurs. Depuis, les forges logicielles sont devenues un outil incontournable pour tous les développeurs de logiciels. Elles intègrent les outils de développement collaboratif (pour le suivi des modifications du code, la gestion des demandes et des réponses d'utilisateurs - tickets -, la gestion des contributions, la gestion du projet), l'industrialisation du processus de création du logiciel à partir de son code source (compilation, tests automatiques, assurance qualité, diffusion des livrables) et des outils de communication comme des forums.

Une forge logicielle, c'est aussi un réseau social de développeurs. Dès que l'on souhaite favoriser l'utilisation et les contributions autour d'un logiciel, se pose la question de choisir la forge en fonction du public, du réseau visé. On peut viser un public de développeurs de l'Enseignement supérieur et de la Recherche français ou international. Il existe des fédérations d'identité comme RENATER ou eduGAIN qui permettent depuis longtemps ces collaborations. Plusieurs forges de l'Enseignement supérieur et de la Recherche donnent accès à ces réseaux de collaboration. Si l'on souhaite ouvrir et partager les codes sources issus de la recherche avec la société dans son ensemble, deux alternatives existent : les forges communautaires libres ou les forges commerciales. Les forges communautaires libres permettent de diffuser au sein d'une communauté un logiciel libre qui a été coopté par cette communauté. La difficulté est donc de trouver une communauté adaptée au logiciel que l'on développe. Les forges commerciales offrent de nombreuses fonctionnalités, avec très peu de contraintes, et souvent de nombreux services quand les logiciels développés sont diffusés sous une licence libre. Parmi ces forges commerciales GitHub, propriété de Microsoft est la plus utilisée, suivi de BitBucket, propriété d'Atlassian, et de GitLab, propriété de GitLab Inc.

Certaines forges, communautaires ou commerciales comme GitLab, peuvent être auto-hébergées par des établissements de l'Enseignement supérieur et de la Recherche, certains disposant ainsi de leur propre forge publique. Cet état des lieux en comptabilise 81 auxquelles s'ajoutent les forges à usage interne. Ces forges auto-hébergées sont souvent simples

à installer : d'un simple exécutable pour des solutions comme Gogs, Gitea ou Forgejo, à un ensemble de logiciels préconfigurés intégrés à une distribution Linux pour GitLab par exemple. GitLab est en effet une forge commerciale (gitlab.com) basée sur un logiciel libre de forge que l'on peut installer sur ses propres serveurs. Le modèle économique de GitLab Inc. est basé sur la vente de licences pour apporter des fonctionnalités supplémentaires aux utilisateurs du service en ligne ou des administrateurs des forges auto-hébergées.

De fait, l'installation d'une forge auto-hébergée pour du développement collaboratif interne nécessite peu de moyens humains ou matériels, et offre une large palette de solutions. Par contre, dès que l'on souhaite s'ouvrir sur l'extérieur, intégrer des solutions d'industrialisation du développement logiciel, mettre en place des bonnes pratiques de développement, un effort plus conséquent est nécessaire et le choix de la solution peut être guidé par des critères différents : popularité de la plateforme, fonctionnalités offertes, robustesse.

Dans l'enseignement supérieur et la recherche, les développeurs de logiciels soutiens ou issus de travaux de recherche ont le choix entre diverses forges pour héberger leur production logicielle. Si leur établissement dispose d'une forge, c'est la solution la plus simple, surtout si aucune interaction en dehors de l'établissement n'est nécessaire.

Quand un besoin d'interaction plus important est nécessaire, les communautés qui développent des logiciels de recherche se tournent fréquemment vers les forges commerciales en ligne, comme en témoignent les lauréats du premier prix science ouverte du logiciel libre de recherche : 9 projets hébergés sur GitHub et un projet hébergé sur SourceForge. L'effet réseau social « on va ou il y a le plus de monde », et la portée internationale des projets justifient ce choix. Il faut cependant se souvenir qu'une forge commerciale peut disparaître en peu de temps : ce fût le cas pour la forge de google, Google Code, dont l'activité a cessé au bout de neuf années d'existence, en quelques mois. La même mésaventure est arrivée à la solution d'hébergement Gitorious. De plus, ces forges ont des conditions d'utilisation que chaque membre doit accepter à titre individuel et non au titre de son établissement.

Les forges auto-hébergées sont un moyen de minimiser ce genre de problème. Il peut cependant aussi arriver que la solution choisie ne soit plus maintenue, ou développée sous une licence libre : c'est arrivé avec le code de SourceForge, maintenu dans une version communautaire sous le nom de GForge, qui a lui même changé de licence, pour être maintenu dans une version communautaire sous le nom de FusionForge, pour arriver à un logiciel peu développé maintenant (la dernière version du logiciel date de 2018).

Le choix de l'auto-hébergement, et de la forge utilisée, n'est donc pas anodin. Sur les 81 forges répertoriées, 72 sont des instances de GitLab (les autres forges utilisent FusionForge, Gitea, Gogs, Redmine, Trac ou Tuleap). On peut expliquer cette domination de GitLab par sa simplicité d'installation et de maintenance et la richesse des fonctionnalités offertes.

L'intérêt de disposer d'une forge spécifique à l'Enseignement supérieur et de la Recherche, quelle que soit l'échelle (établissement, nationale, européenne, internationale) se pose donc. Les forges d'établissement apportent une réponse quand le développement est interne à l'établissement et qu'une forge d'établissement existe. On maîtrise dans ce cas les fonctionnalités offertes, et l'accès aux données, par contre cela n'autorise que très peu ou pas le développement entre plusieurs établissements. Quand une forge d'établissement n'existe pas ou n'offre pas aux porteurs de projets d'inviter des contributeurs externes, une forge nationale ou européenne permettrait d'offrir une alternative à l'utilisation des forges commerciales.

Au milieu des années 2000, une forge nationale, SourceSup, a été mise en place par RENATER pour dépasser ces restrictions d'interaction. Cependant, cette forge, qui représentait l'état de l'art lors de sa création, propose désormais à ses utilisateurs une collection d'outils qui sort des standards de développement actuels.

Cet état des lieux dresse le panorama des forges et pratiques existantes et met en avant un certain nombre d'observations et de points de vigilance sur la situation actuelle.

()

Sommaire

Résumé ii

Historique 1

1. Introduction 2

2. De l'importance des forges 4
- 2.1. Surveiller l'évolution du code source . . . 4
- 2.2. Piloter tout le cycle de vie du logiciel . . . 5
- 2.3. Permettre et faciliter la collaboration . . . 6
- 2.4. Construire le logiciel, analyser son code source . . . 6
- 2.5. Au delà du code source . . . 7
- 2.6. Évolution du public visé par les forges . . . 8
- 2.7. Cas particulier du logiciel libre de recherche . . . 9
- 2.8. Publics de l'ESR visés par les forges . . . 10

3. Paysage des forges de l'ESR 12
- 3.1. Utilisation importante des forges commerciales . . . 12
- 3.2. Utilisation des forges communautaires libres . . . 14
- 3.3. La forge nationale SourceSup . . . 15
- 3.4. 81 forges publiques auto-hébergées dans l'ESR . . . 17
- 3.5. Analyse de la situation . . . 18
  - 3.5.1. Frise chronologique des forges mentionnées dans ce document . . . 18
  - 3.5.2. Pourquoi tant de forges auto-hébergées ? . . . 19
  - 3.5.3. Difficulté d'interaction avec la société . . . 21
  - 3.5.4. Niveau de support et besoin de confiance . . . 24
  - 3.5.5. Licence libre ou propriétaire ? . . . 25

4. Points d'attention sur les forges 27
- 4.1. Vitrine ou simple outil ? . . . 27
- 4.2. Organisation des projets . . . 28
- 4.3. Gestion du droit d'auteur . . . 29
- 4.4. Gérer un projet sur plusieurs forges . . . 29
- 4.5. Toujours plus de services en intégration continue . . . 30

5. Récapitulatif des solutions 33
- 5.1. Forges externes commerciales . . . 33
- 5.2. Forges externes communautaires . . . 34

5.3. Forges auto-hébergées à l'échelon local 35
5.4. Forge auto-hébergée à l'échelon national 36

6. Conclusion **37**

A. Liste des forges « publiques » auto-hébergées de l'ESR **40**
A.1. Établissements 40
A.2. Laboratoires 45

B. Contributeurs **48**
B.1. Membres du collège codes sources et logiciels 48
B.2. Invités 48
B.3. Communautés 48

C. Questionnaire **49**
C.1. URL de la forge 49
C.2. Par qui est proposée/maintenue cette forge ? 49
C.3. Cette forge est elle publique ? 49
C.4. Comment s'identifie t'on sur cette forge ? 50
C.5. Quels sont les services disponibles ? 50
C.6. Comment sont organisés les projets ? 50
C.7. Commentaire libre 51

D. Glossaire **52**

Bibliographie indicative **54**

Historique

Version	Date	Commentaire
8	17/06/26	Mise à jour de la liste des forges.
7	29/09/24	9 nouvelles forges. Une forge en moins.
6	11/05/24	Politique E4S. V3 en anglais.
5	25/01/24	6 nouvelles forges. Contributions SPOS. Informations ForgeFed.
4	15/11/23	27 nouvelles forges. Modifications mineures.
3	06/09/23	Une nouvelle forge. Mise à jour mineure. Traduction en anglais disponible.
2	04/06/23	Correction de typos et de mise en forme par des lecteurs
1	02/05/23	Version initiale du collège codes sources et logiciels du Comité pour la science ouverte

Ce document est diffusé sous licence creative commons attribution (CC BY 4.0). Vous pouvez contribuer à l'évolution de ce document disponible en version source sur la forge publique de IN2P3.

Nous recommandons l'utilisation de tickets pour nous faire part de vos suggestions et compléments d'information.

1 | Introduction

Ce rapport porte sur les forges logicielles utilisées dans les établissements de l'enseignement supérieur et de la recherche (ESR). En particulier, il s'intéresse aux pratiques et aux besoins, en terme de génie logiciel, pour valoriser au mieux les productions logicielles issues de la science ouverte. L'objectif de ce document est de faire un premier état des lieux des forges logicielles utilisées dans l'Enseignement supérieur et la recherche français et d'identifier les moyens de rendre plus visibles les productions logicielles issues de la science ouverte.

Les forges logicielles sont principalement conçues pour gérer sur tout leur cycle de vie les différents artéfacts associés aux activités de génie logiciel, comme le code source, les fichiers binaires ou encore les documentations. Les forges peuvent aussi être utilisées pour la rédaction collaborative d'articles scientifiques ou de documentation, ou encore de pages web, à l'aide d'outils de transformation de fichiers textes faiblement structurés (par exemple, Markdown et AsciiDoc). Ces activités nécessitent de plus une intégration continue avec les outils de transformation, de déploiement et de publication ouverte du code source, de la documentation. Ce document n'est donc pas strictement limité au logiciel proprement dit, son code source et ses binaires, mais à la gestion de tous les artefacts produits et échangés pour produire un logiciel de qualité, partageable et réutilisable. Les forges logicielles sont aussi utilisées pour partager des données, des modèles, de manière collaborative. Leur usage impacte donc les trois piliers de la science ouverte.

Notons que cette analyse ne concerne que les besoins de l'activité recherche de l'Enseignement supérieur et de la Recherche. Elle ne concerne pas les besoins de forges en enseignement, ni ceux des outils de système d'information pour le fonctionnement des établissements. Il s'agit en premier lieu de faire un état des lieux des pratiques actuelles et des propositions d'actions afin de favoriser de bonnes pratiques de développement et de valorisation de la production logicielle. Cette proposition d'action s'appuie sur une analyse des limitations actuelles, et une première estimation des besoins communs.

Ce document structuré autour de trois chapitres précise pour tous le rôle des forges logicielles et dresse pour les publics avertis en développement logiciel un premier état des lieux des forges de l'Enseignement supérieur et de la Recherche et de leurs usages. Sur la base de l'analyse des limitations rencontrées par les établissements de l'Enseignement supérieur et

de la Recherche, des premières pistes d’action sont suggérées.

2 | De l'importance des forges

Développer un logiciel nécessite toujours la mise en œuvre de bonnes pratiques. En effet, quelles que soient les caractéristiques du logiciel à développer, il est toujours plus facile de démarrer avec des bonnes pratiques, que d'essayer a posteriori d'en instaurer. Quoiqu'il arrive, la qualité du logiciel n'en sera que meilleure. Un point important par exemple, mais pas suffisant comme nous le verrons plus tard, est la gestion des évolutions du code source. Elles doivent être systématiquement tracées avec un système de gestion de versions comme par exemple Git, Mercurial ou Subversion.

De nombreuses plateformes web, souvent appelées forges logicielles, comme par exemple BitBucket, Gitea, GitHub, GitLab, Gogs et ForgeJo, simplifient la mise en oeuvre de ces bonnes pratiques et facilitent le développement de logiciels de meilleure qualité, ainsi que la constitution de communautés de contributeurs et d'utilisateurs.

Des instances de ces plateformes peuvent être mises en place localement par des équipes de recherche, des laboratoires, des départements, voire même l'établissement d'appartenance ou par une infrastructure de recherche, comme Huma-Num pour les sciences humaines et sociales.

Les communautés de chaque plateforme peuvent apporter des bénéfices appréciables et participer à la recherche de bogues, à l'ajout de nouvelles fonctionnalités, à la contribution de documentation, etc.

Pour bénéficier de l'apport d'une communauté, il faut investir le temps et l'énergie nécessaires pour organiser son fonctionnement : fournir de la documentation sur les moyens et les possibilités de contributions, évaluer et gérer rapidement les suggestions et contributions apportées, planifier la feuille de route du développement logiciel, etc.

Voir (Pellegrini, Di Cosmo, Romary, Janik, Hodenq, Coutanson, et Géroudet, 2022) pour une discussion plus détaillée du sujet.

2.1. Surveiller l'évolution du code source

Le développement d'un logiciel, en particulier l'écriture de son code source (qu'on appelle programmation), est une activité itérative et incrémentale. Le code source d'un logiciel est amené à être modifié tout au long de son cycle de vie, et non pas juste à sa création. En effet,

un logiciel évolue dans le temps : on parle de maintenance corrective quand il s'agit d'évolution dont l'objet est de corriger des bogues et de maintenance évolutive quand il s'agit de prendre en compte les évolutions de l'environnement d'exécution – évolution du matériel support, du système d'exploitation, des bibliothèques utilisées, etc. – ou d'introduire de nouvelles fonctionnalités.

Pour prendre en compte ces évolutions, il est d'usage de s'appuyer sur des outils qui peuvent enregistrer toutes les modifications faites à chaque étape de l'évolution du code : on les appelle des *outils de gestion de versions* (en anglais « *version control systems* » ou VCS). L'un des gestionnaires de versions les plus utilisés actuellement est Git.

Le gestionnaire de versions permet donc de suivre finement l'historique d'écriture d'un code, c'est-à-dire chaque contribution (les « *commits* »), de facilement comparer plusieurs versions du code du logiciel (les « *branches* »), et de formaliser des versions précises d'un logiciel (les « *releases* »).

2.2. Piloter tout le cycle de vie du logiciel

Le développement d'un logiciel de qualité ne s'arrête pas à la phase de réalisation du logiciel, c'est-à-dire la phase d'écriture du code. En effet, le code est certes un artefact important issu du processus de développement logiciel, mais les spécifications (aussi appelées « ensemble des exigences » ou encore « cahier des charges »), les modèles de conception, les tests, les documentations concepteur et utilisateur en sont d'autres tout aussi importants si l'on souhaite développer un code de qualité et pérenne dans le temps, c'est-à-dire maintenable, voire réutilisable.

Le gestionnaire de versions est donc l'un des outils à mettre en œuvre, mais pas le seul. Pour gérer tout le cycle de vie d'un logiciel, on a souvent besoin de gérer en plus les besoins des utilisateurs, de consigner les défauts constatés du logiciel, de produire des versions diffusables du logiciel, de lancer des tests périodiquement, de produire de la documentation, etc.

Pour faire tout cela, il est recommandé d'utiliser un ensemble d'outils intégrés au sein d'une plateforme souvent appelée « forge logicielle ».

Les forges peuvent être utilisées dès la création d'un logiciel, au sein d'un espace de travail privé. Quand le logiciel est prêt à être diffusé publiquement, il y a alors deux solutions : soit le projet devient public (et l'on peut consulter l'historique de son développement), soit un nouveau projet public est créé (avec un historique du développement vierge, l'ancien

n'étant pas accessible).

2.3. Permettre et faciliter la collaboration

L'intérêt d'une forge est de favoriser la collaboration autour d'un logiciel, notamment par la collecte des défauts rencontrés dans son utilisation et le recueil des cas d'usage non couverts par le logiciel, mais aussi de permettre la contribution de code. Une personne peut contribuer au code en communicant un correctif ou une amélioration au projet, soit sous forme de « *patch* » (fichier contenant les changements apportés), soit en réalisant une requête de fusion des changements (une « *pull request* », PR, ou « *merge request* », MR, selon les plateformes). C'est aussi un moyen de conserver une copie du logiciel sur une infrastructure mutualisée, qui peut être archivée par Software Heritage quand il s'agit d'un logiciel dont le code source est public.

La forge permet aux auteurs du logiciel de gérer finement ces collaborations en choisissant les personnes autorisées à faire des contributions, via un système de gestion de droits.

Les exemples de logiciels libres de recherche à grand succès confirment que faciliter le plus possible les contributions est un levier très important pour permettre l'émergence d'une communauté de contributeurs. Une telle communauté peut alors décharger les créateurs initiaux du logiciel de diverses tâches de maintenance et de développement, et permet au logiciel de prendre son envol. Les obstacles tels que le parrainage ou la modération pour la création de compte peuvent freiner de manière importante (et invisible !) la venue de nouveaux contributeurs.

On trouvera dans (Raymond, 2001) et (Bangerth et Heister, 2013) par exemple des conseils généraux pour le développement de logiciels libres. Ils soulignent notamment l'importance de la collaboration dans ce contexte.

2.4. Construire le logiciel, analyser son code source

Une fonctionnalité que l'on retrouve souvent dans ces forges est l'« intégration continue » (ou « *continuous integration* », CI), c'est-à-dire la possibilité de construire le logiciel automatiquement à partir de ses sources. Il s'agit en quelque sorte de la garantie que la forge contient toutes les informations nécessaires pour construire le logiciel. Les technologies actuelles permettent de cibler simultanément diverses plateformes (multiples systèmes d'exploitation et architectures de processeurs).

Pour les développeurs, l'intégration continue permet aussi de tester le fonctionnement du logiciel au fur et à mesure des développements, d'éviter les régressions (perte ou mauvais fonctionnement des fonctionnalités existantes) et de disposer de métriques sur la « qualité » du code développé.

On peut classer les solutions de forges libres auto-hébergeables par le niveau de fonctionnalités offertes.

Table 2.1.: Typologie de forges

Niveau	Fonctionnalités	Exemples
0	Hébergement de code source versionné (pas de système de tickets ou de CI)	cgit
1	Niveau 0 + système de tickets et de revue de code	Redmine, Gerrit, Trac
2	Niveau 1 + système de *pull/merge request*	Gitea/Forgejo, Gogs
3	Niveau 2 + CI/CD, voire autres modules (GitLab pages, SourceHut pages)	GitLab, SourceHut, Tuleap

2.5. Au delà du code source

Si les forges ont été conçues au départ pour gérer le cycle de vie de logiciels, elles sont aujourd'hui utilisées dans un cadre plus large :

Publication et maintenance de sites web à l'aide d'outils comme GitHub/GitLab/SourceHut pages ;

Rédaction d'articles scientifiques en utilisant des outils de composition de documents tels que LaTeX, en passant éventuellement par un langage semi-structuré comme Markdown, AsciiDoc ou reStructuredText plus simple à appréhender comme source ;

Partage de données, de modèles La gestion fine des contributions et la mise en ligne de contenu permet la création de communautés autour du partage de données, de modèles (voir par exemple la communauté HTR United).

Toutes les remarques suivantes sur le logiciel sont aussi valables pour ces usages.

2.6. Évolution du public visé par les forges

Il y a vingt ans, les forges étaient le lieu de rencontre entre les utilisateurs et les développeurs d'un logiciel. Basées sur la version libre de la forge sourceforge.net (telle que notamment la forge GForge), on y retrouvait deux fonctionnalités importantes pour les utilisateurs : le téléchargement des différentes versions du logiciel et des forums ou listes de diffusion pour demander de l'aide et interagir avec l'équipe de développement. La gestion du code source du logiciel était une fonctionnalité parmi d'autres. Il existait une séparation claire entre les membres de l'équipe de développement, qui avaient les droits de modification sur le code source, et les autres utilisateurs.

Les logiciels et les pratiques de développement ont évolué. Certains logiciels sont fournis sous forme de services, ce qui fait qu'il n'est plus nécessaire de les télécharger. Les bibliothèques sont diffusées par des entrepôts (maven, pip, npm, etc) qui sont couplés à des outils de gestion des dépendances. Des sites d'entraide comme StackOverflow deviennent des sources d'information alternatives aux forums officiels des logiciels.

Avec l'arrivée de GitHub en 2007, le positionnement des forges a changé : le code source est devenu central, c'est la première chose que l'on voit quand on arrive sur un projet. Toutes les discussions sont contextuelles, liées à des tickets ou des contributions. Avec la possibilité de créer très facilement une divergence (ou « *fork* «) d'un projet, le code source devient un bien commun, modifiable et diffusable par tous, ce qui favorise les contributions. On constate en parallèle l'émergence des environnements de développement en ligne, et la généralisation de l'intégration et du déploiement continus. Le public de la forge devient plus spécialisé : il faut déjà connaître les pratiques de développement pour s'y retrouver, alors que disparaissent les forums servant à poser des questions et permettre à la communauté d'y répondre.

À ce titre, il convient de noter que GitHub permet de nouveau depuis fin 2020 les échanges entre utilisateurs sans ouvrir de tickets à l'aide de la fonctionnalité « discussion ».

Observation 1

La forge devient un réseau social de développeurs autour d'un bien commun : le code source.

2.7. Cas particulier du logiciel libre de recherche

Le logiciel libre de recherche est souvent issu d'une preuve de concept. Il n'a pas vocation initialement à être diffusé en dehors du monde académique, et n'a donc pas été conçu pour cela. L'un des enjeux de la science ouverte est de rendre visible cette production logicielle, en permettant aux laboratoires d'intégrer les bonnes pratiques qui leur permettront de rendre visible leur production logicielle à moindre effort.

Observation 2

Le logiciel libre de recherche est conçu dans des laboratoires qui peuvent avoir des tutelles diverses. Les interactions doivent donc pouvoir se faire entre membres de ces différentes tutelles, ce qui signifie qu'idéalement, les personnels de ces différentes tutelles puissent avoir accès au même outil.

Des logiciels sont parfois développés lors de projets collaboratifs qui peuvent inclure des entreprises. Il est important de pouvoir accueillir ce type de projets. Dans ce cadre, l'outil en question ne doit pas être limité au monde académique.

Plus généralement, il est aussi important, pour certains projets, de pouvoir interagir avec des utilisateurs hors Enseignement supérieur et Recherche (collecte de retours via des « tickets », contribution de code, de documentation). C'est un facteur décisif pour qu'une communauté d'utilisateurs et de contributeurs puisse se développer autour de ces logiciels.

La modularité et la construction de logiciels par l'intégration de composants (c'est-à-dire d'autres briques logicielles), rendues disponibles grâce au cadre de la science ouverte et une diffusion libre, sont aussi des vecteurs facilitateurs de collaborations, disciplinaires comme interdisciplinaires.

Les logiciels libres de la recherche peuvent aussi servir de sujets d'étude pour les chercheurs en génie logiciel. Les rendre accessibles sur une forge bien identifiée favoriserait leur réutilisation dans ce cadre. De plus, en intégrant l'état de l'art en matière de génie logiciel (analyse statique de code, tests, assurance qualité, documentation, etc.), une forge académique permettrait à l'ensemble de la communauté de profiter des nouvelles avancées dans ce domaine.

La facilité de « reproductibilité » offerte par l'intégration continue (via des images d'environnements d'exécution maîtrisés) est aussi un aspect très important en science ouverte.

Favoriser l'utilisation d'outils standards faciliterait la formation et l'acculturation au sein des laboratoires. L'automatisation de tâches à l'aide de l'outil informatique est devenue une compétence importante dans toutes les disciplines académiques (Wilson, Aruliah, Brown, Chue Hong, Davis, Guy, Haddock, Huff, Mitchell, Plumbley, Waugh, White, et Wilson, 2014). La forge logicielle doit devenir un outil comme un autre dans la boite à outils des chercheurs et des ingénieurs.

2.8. Publics de l'ESR visés par les forges

Une forge peut être utilisée à différents moments d'un projet de recherche.

Lors du déroulement d'un projet, elle peut être utilisée pour le développement d'une preuve de concept. Ce travail peut être réalisé par des personnels ingénieurs ou chercheurs, qui peuvent être temporaires (stagiaires de recherche ou contractuels: ingénieurs et chercheurs doctorants ou docteurs) ou permanents. On créera dans ce cas un projet privé, pour lequel l'historique du gestionnaire de versions constituera un moyen commode de connaître la contribution de chacun au logiciel produit. C'est une pratique à encourager.

À un moment donné, pour tout projet se pose la question de diffuser les résultats obtenus, dont les logiciels font partie. Quand il s'agit de diffuser ces résultats sous forme de logiciels libres, plusieurs stratégies peuvent être mises en œuvre.

Une stratégie minimaliste est de diffuser le logiciel en l'état, sans favoriser outre mesure l'interaction avec d'autres groupes de recherche ou la société. On peut simplement diffuser le code exécutable du logiciel, par exemple en le rendant disponible sur une page web sous forme de fichier archive (.tar ou .zip). Cependant, un tel mode de diffusion limite l'impact de la mise à disposition, en rendant difficile les contributions.

Une stratégie plus volontariste est de diffuser le logiciel sur une forge, pour permettre des retours d'autres utilisateurs (notamment via un système de tickets). Il est alors préférable de réaliser en amont un effort de documentation sur le fonctionnement et la construction du logiciel, pour éviter des sollicitations fréquentes sur ces sujets. Il s'agit ici d'une véritable approche de valorisation du logiciel. De plus en plus de projets de recherche intègrent cet aspect durant le déroulement du projet.

Une stratégie accomplie est d'ouvrir le développement même du logiciel, pour faciliter les contributions d'autres développeurs (au travers des MR/PR). Il est alors pertinent de documenter le fonctionnement interne du logiciel pour faciliter la plongée dans le code, par exemple en rédigeant un « manuel de maintenance ». Il est aussi utile de documenter le processus de contribution : style de code attendu, processus de soumission de MR/PR, de relecture de code, fonctionnement de la CI, accord de cession de droits d'auteur, etc. Les forges permettent généralement de faciliter ce processus, pour favoriser les contributions. Cela peut être décisif dans la prise d'envol d'un projet logiciel, dont la maintenance et le développement peuvent alors être supportés par une communauté internationale (bénéficiant donc d'une puissance de travail importante), plutôt que la seule équipe de recherche à l'origine du logiciel (avec ses moyens limités).

Les forges utilisées pour le déroulement du projet et pour la valorisation du logiciel peuvent être différentes, afin de s'adapter aux besoins respectifs de ces deux versants.

Observation 3

Il existe actuellement de nombreux logiciels libres issus de la recherche française qui sont développés depuis des années grâce à une communauté active et hébergés sur des forges commerciales.

3 | Paysage des forges de l'ESR

Nous faisons le point dans ce chapitre sur le type de forges utilisées dans l'enseignement supérieur dans le cadre de la recherche. Si ce panorama n'est pas exhaustif, il permet de noter des tendances. Une analyse des raisons qui ont contribué a aboutir à cette situation est aussi proposée.

3.1. Utilisation importante des forges commerciales

La plupart des logiciels libres issus de l'Enseignement supérieur et de la Recherche et désireux d'interaction avec la société utilisent une solution d'hébergement commerciale ouverte à tous, comme `github.com`, `gitlab.com` ou `sourceforge.net` (Escamilla, Klein, Cooper, Rampin, Weigle, et Nelson, 2022). Il suffit de regarder les dépôts où sont hébergés les logiciels récompensés lors du prix science ouverte du logiciel libre pour s'en convaincre :

Première édition (2022) :

- Coq : https://github.com/coq/coq
- Coriolis : https://gitlab.lip6.fr/vlsi-eda/coriolis
- Scikit-learn : https://github.com/scikit-learn/scikit-learn
- Vidjil : https://github.com/vidjil/vidjil
- WebObs : https://github.com/IPGP/webobs
- Faust : https://github.com/grame-cncm/faust
- OpenVibe : https://gitlab.inria.fr/openvibe/meta.git
- Gammapy : https://github.com/gammapy/gammapy
- SPPAS : https://sourceforge.net/projects/sppas/
- Gama : https://github.com/gama-platform/gama

Seconde édition (2023) :

- Brian simulator : https://github.com/brian-team/brian2
- Fink : https://github.com/astrolabsoftware
- Hyphe : https://github.com/medialab/hyphe
- KeOps : https://github.com/getkeops/keops
- NoiseCapture : https://github.com/Universite-Gustave-Eiffel/NoiseCapture/
- OCaml : https://github.com/ocaml/ocaml
- PPanGGOLIN : https://github.com/labgem/PPanGGOLiN/
- Smilei : https://github.com/SmileiPIC/Smilei

La plateforme code.gouv.fr référence, au 11 février 2023, 9818 dépôts relevant de l'Enseignement supérieur et de la Recherche. **29 %** de ces dépôts (2627) sont hébergés sur GitHub. Seulement 30 dépôts sont hébergés sur gitlab.com. Les autres solutions commerciales ne sont pas encore comptabilisées.

La plateforme Software Heritage permet d'avoir une idée du nombre de projets publics globaux trouvés sur les forges commerciales : 156 M pour GitHub, 4 M pour GitLab, 2 M pour Bitbucket, pour les plus utilisées début février 2023. On note que GitHub contient bien plus de projets publics que les autres forges.

Le principal intérêt de se tourner vers ces solutions pour un projet libre de l'Enseignement supérieur et de la Recherche est de faciliter les contributions de code ou de documentation en adoptant un modèle de « réseau social de développeurs ». En effet, pour les nombreux développeurs ayant un compte sur ces plateformes, contribuer à un projet public de ces forges se fait de manière très naturelle. C'est la raison pour laquelle des structures communautaires comme Eclipse par exemple ont décidé d'intégrer ces forges commerciales, alors qu'elles maintiennent aussi leur propre forge.

Cependant, les forges commerciales ne sont pas des solutions pérennes pour héberger des logiciels de la recherche. Ainsi, en août dernier, GitLab prévoyait d'archiver automatiquement les projets non actifs depuis un an (Sharwood, 2022), alors que certains logiciels matures peuvent avoir un cycle de développement plus lent et des périodes d'inactivité prolongées, sans que cela remette en cause leur utilité. Par le passé, on a pu voir la fermeture de forges commerciales comme Google Code (Staff, 2015) et même de services communautaires comme Gitorious (Degeler, 2015) ou la suppression de centaine de milliers de projets suite à la décision de BitBucket de ne plus offrir d'hébergement pour les projets utilisant le système de contrôle de version Mercurial (Chan, 2020). Ces fermetures ont conduit à la disparition d'un patrimoine important, concernant les historiques de modifications et les échanges de forums.

Les conditions d'utilisation des forges commerciales peuvent donc poser des problèmes dans ce contexte.

Enfin, les forges commerciales ne répondent pas forcément aux besoins de l'ensemble du cycle de vie d'un projet. L'utilisation de forges commerciales pour des projets privés doit se faire dans le respect de la réglementation européenne sur la protection des données, qui préconise que les serveurs soient hébergés sur le sol européen, et ne soient pas soumis à des législations extra-européennes (telles que : CLOUD Act, FISA 702, etc.). Plus d'informations à ce sujet sont disponibles sur le site de la CNIL.

Observation 4

Les conditions d'utilisation des forges commerciales doivent être acceptées de manière individuelle par les personnels de l'Enseignement supérieur et de la Recherche pour leur permettre d'accéder à ces plateformes. Cela peut poser des problèmes si un personnel n'accepte pas ces conditions.

3.2. Utilisation des forges communautaires libres

Il existe de nombreuses communautés libres, comme la Free Software Foundation (FSF), et son projet GNU, Apache, Eclipse, OW2 et bien d'autres qui, en plus de leurs projets propres, accueillent des projets libres issus du monde de la recherche. Elles permettent de faire émerger des projets, en les accompagnant au delà du simple démarrage. Quand elles sont parties prenantes de ces projets, elles peuvent pérenniser les productions logicielles. Ces communautés peuvent aussi avoir leur propre positionnement concernant la science ouverte ; voir par exemple celui d'OW2.

Cependant, ces forges hébergent des projets mûrs, professionnels, qui ont vocation à être utilisés de façon large. Ces communautés décident des projets qu'elles hébergent et soutiennent, selon leurs critères propres. Il peut s'agir de la thématique du projet, de son niveau de maturité, de ses processus de développement ou de gouvernance, de son positionnement dans le monde économique ou sociétal.

Observation 5

Les forges communautaires libres sont donc une solution pour des projets académiques qui ont obtenu une certaine visibilité, ou qui ont une stratégie de valorisation bien définie. Elles ne sont pas une solution pour le développement de preuves de concepts dans les laboratoires.

> La forge d'OW2 peut cependant être utilisée comme forge d'expérimentation, via les « *community user* » c'est à dire les membres individuels, qui peuvent disposer de deux projets en leur nom propre. Ces projets individuels n'entrent pas dans le processus classique de soumission d'un projet communautaire.

Voici quelques exemples de logiciels des fondations OW2, Eclipse et du Projet GNU issus de la recherche publique :

- ASM (licence BSD) : https://asm.ow2.io/
- MPFR (licence GNU LGPL) : https://www.mpfr.org/
- OM2M (licence EPL) : https://www.eclipse.org/om2m/
- Papyrus (licence EPL) : https://www.eclipse.org/papyrus/
- Sat4j (licence GNU LGPL/EPL) : https://www.sat4j.org/
- SensiNact (licence EPL) : https://projects.eclipse.org/projects/technology.sensinact
- Spoon (licence CeCILL-C/MIT) : https://spoon.gforge.inria.fr

3.3. La forge nationale SourceSup

La forge SourceSup (https://sourcesup.renater.fr) a été mise en service en 2004 par RENATER. Il s'agissait à l'époque d'une instance de GForge. Ce service a basculé sous FusionForge en 2009, quand la solution GForge a cessé d'être libre. Cependant, la solution FusionForge n'est plus développée depuis quelques années (la dernière version majeure date de 2018). Des briques fonctionnelles supplémentaires ont été rajoutées en 2015 : MantisBT pour la gestion de tickets, Testlink pour la gestion des campagnes de tests, Sonarqube pour l'assurance qualité, Nexus pour la gestion d'artefacts, Jenkins pour l'intégration continue, et Nuxeo pour la gestion documentaire. L'infrastructure est virtualisée, selon un principe de séparation des applications gourmandes en ressources. Une personne assure la maintenance et le support de « niveau 2 » des briques logicielles. Le support de « niveau 1 » est mutualisé avec les autres applications RENATER.

En décembre 2022, la forge SourceSup contenait 762 projets publics sur plus de 5200 projets au total, et comptabilise plus de 13 000 utilisateurs.

Quelques exemples de projets hébergés sur SourceSup :

AGATTE : Progiciel de gestion des congés, utilisé dans beaucoup d'établissements de l'Enseignement supérieur et de la Recherche

WIMS : Logiciel d'enseignement interactif

OTAWA : Cadriciel C++ pour déterminer le temps d'exécution au pire de cas de programmes.

La forge accepte des projets publics ou privés. Tout personnel de l'Enseignement supérieur et de la Recherche peut demander la création d'un projet. Cependant, les projets étudiants ne sont pas acceptés.

Observation 6

L'évolution technique et fonctionnelle de SourceSup est décidée par le comité de direction de RENATER. Aucune évolution majeure de SourceSup n'est prévue à ce jour.

La solution SourceSup est actuellement assez différente des autres forges que l'on peut trouver dans l'Enseignement supérieur et la Recherche. Tout d'abord, elle a gardé son positionnement orienté utilisateur, avec ses listes de diffusion et ses forums (à la SourceForge). De plus, elle maintient un support de Subversion. En effet, de nombreux projets ont commencé avec ce gestionnaire de versions centralisé, qui était le plus utilisé avant l'apparition de Git. Si SourceSup permet d'utiliser Git comme gestionnaire de versions, comme les autres forges, il propose aussi d'utiliser Subversion.

SourceSup a fait le choix d'utiliser des outils de différentes provenances pour fournir tous les services nécessaires au développement de logiciels. Cette démarche détone par rapport à l'évolution des forges commerciales comme GitHub ou GitLab.com, mais aussi des forges auto-hébergées qui offrent désormais la plupart de ces outils au sein d'une seule et unique solution, afin de faciliter leur appropriation par les utilisateurs et leur maintenance par les établissements.

Le choix d'outils distincts permet une certaine forme de souveraineté, en maîtrisant finement les logiciels utilisés pour chaque besoin. Cependant, c'est aussi un frein à l'adoption de la forge par de nouveaux utilisateurs, qui ne retrouvent pas les outils qu'ils ont désormais l'habitude d'utiliser.

À noter aussi : la forge nationale éducative

Récemment, le ministère de l'éducation nationale et de la jeunesse a annoncé la création d'une nouvelle forge nationale, dite « forge des communs numériques éducatifs » :

Par ailleurs, les communautés d'enseignants (et d'autres acteurs de l'éducation) peuvent également être des lieux de construction de nouveaux outils. Des professeurs, notamment de NSI ou de SNT, sont en attente d'une « forge » qui leur permettrait de collaborer entre pairs et de partager du code informatique. Le ministère répond dorénavant à ce besoin avec la mise à disposition d'une forge technologiquement souveraine et mutualisée à l'échelle nationale.

Cette forge est depuis l'automne 2023 déployée à l'adresse https://forge.apps.education.fr/.

Source : Stratégie du numérique pour l'éducation 2023-2027 (MENJ), page 25 (MENJ, 2023)

3.4. 81 forges publiques auto-hébergées dans l'ESR

En 2023, on compte au moins 81 forges auto-hébergées publiques dans les établissements de l'Enseignement supérieur et de la Recherche, sans compter les forges à usage strictement interne des établissements.

La liste des forges répertoriées, disponible en annexe A, ne se veut pas exhaustive. Elle est issue, d'une part, des déclarations qui alimentent la plateforme `code.gouv.fr` et, d'autre part, du réseau de connaissance des membres du groupe de travail à l'origine de ce rapport à qui a été envoyé le questionnaire en annexe C. Il faut la considérer comme un recensement préliminaire qui, bien qu'incomplet, fournit d'ores et déjà des informations intéressantes pour le présent rapport. Bien qu'il puisse être intéressant de la compléter, il faut garder en tête les difficultés sous-jacentes à un tel recensement exhaustif. Il est par nature difficile de répertorier toutes les installations de forges, car celles-ci peuvent être opérées à divers niveaux : groupe de recherche, laboratoire, établissement, etc. Certaines de ces instances sont exclusivement dédiées aux usages internes d'un établissement de l'Enseignement supérieur et de la Recherche (celle de l'université d'Artois, par exemple), et ne sont donc pas répertoriées ici.

On peut noter, à la lecture de cette liste, que :

- les grands organismes de recherche nationaux disposent de leur propre forge ou sont sur le point d'en disposer ;
- des universités ou écoles proposent leurs forges, en séparant éventuellement l'enseignement et la recherche (comme pour CentraleSupélec par exemple) ou sur une granularité plus fine encore (l'université de Lille propose sa propre instance GitLab, le laboratoire CRIStAL la sienne et le département informatique de la faculté des sciences et technologies une autre).

Observation 7

Beaucoup de forges auto-hébergées sont des instances de GitLab. Bien que GitLab soit un logiciel libre auquel chaque personne peut contribuer, il convient de s'interroger sur la dépendance que cela implique par rapport à la société GitLab Inc. qui, en pratique, en dirige le développement.

3.5. Analyse de la situation

3.5.1. Frise chronologique des forges mentionnées dans ce document

En avant propos, pour comprendre la situation actuelle, il est important d'avoir en tête les quelques dates clés liées aux forges logicielles que nous considérons :

- 1999 : apparition de SourceForge, la première forge de logiciels libres
- 2002 : apparition de GForge, l'alternative libre à SourceForge pour l'auto-hébergement
- 2005 : apparition de Git (logiciel libre de gestion de versions)
- 2007 : apparition de GitHub (forge commerciale basée sur git)
- 2011 : apparition de GitLab (logiciel libre fonctionnellement similaire à GitHub pour l'auto-hébergement de projets privés)
- 2012 : apparition de GitLab.com (forge commerciale basée sur le logiciel GitLab, projets privés)
- 2014 : GitLab.com auto-héberge son développement, devenant ainsi une alternative à GitHub

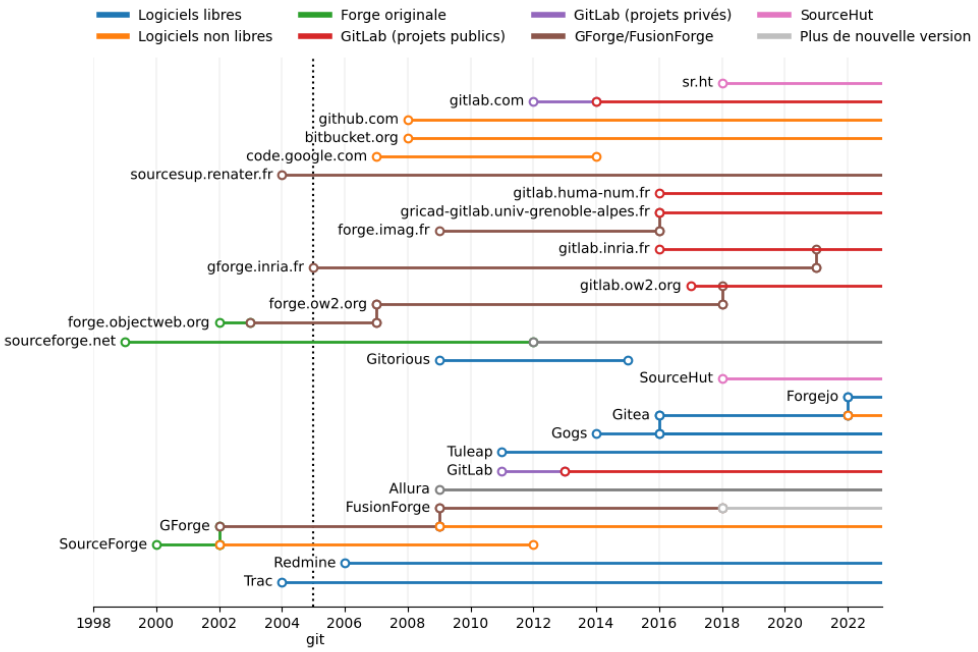


Figure 3.1.: Dates d'apparition de différentes forges

L'apparition d'une version libre du logiciel de SourceForge a permis la création des premières forges auto-hébergées. Quand la licence de diffusion en est devenue propriétaire, des *forks* issus de la communauté ont vu le jour (SourceForge/GForge/FusionForge par exemple). Des changements de technologie apparaissent régulièrement. La forge `sourceforge.net` utilise désormais le logiciel libre Apache Allura.

On peut noter, à la fin des années 2010, le passage de la technologie GForge/FusionForge vers GitLab pour l'auto-hébergement au sein de plusieurs structures. Les premières versions de GitLab ne permettaient que l'hébergement de projets privés. C'est en octobre 2013 (GitLab 6.2) que GitLab a introduit la notion de projet public, et qu'il est devenu une véritable alternative à GForge/FusionForge. En janvier 2014, le développement collaboratif de GitLab se fait à l'aide de GitLab.

3.5.2. Pourquoi tant de forges auto-hébergées ?

Il est difficile de répondre à cette question car il existe sans doute de nombreuses raisons à cet état de fait. Il est néanmoins possible d'avancer quelques éléments susceptibles d'expliquer cette situation.

Tout d'abord, on peut noter que la quasi-totalité des forges répertoriées sont des instances de GitLab. Ce logiciel est très simple à installer et à maintenir, et n'est pas gourmand en ressources informatiques. Il peut

donc être facilement déployé dans un établissement à moindre coût, humain et financier.

Notamment, dans les universités ou les écoles d'ingénieurs possédant une spécialité informatique, ces forges sont nécessaires pour permettre l'apprentissage de leur utilisation dans un cadre pédagogique. Si dans certains établissements il existe une distinction entre forges d'enseignement et de recherche, ce n'est pas le cas dans d'autres.

Des forges sont également souvent déployées dans les laboratoires d'informatique, ces derniers ayant à la fois les compétences techniques pour mettre en place ces solutions et la connaissance des limites des solutions commerciales.

Le principal intérêt d'utiliser une forge auto-hébergée est de maîtriser la localisation des serveurs, et donc de respecter la législation européenne sur le traitement des données à caractère personnel (le RGPD) et de prendre en compte si nécessaire les enjeux de souveraineté (pour les projets privés ou pour les partenariats avec des industriels par exemple).

Un autre intérêt de l'auto-hébergement est de pouvoir maîtriser l'accès à ce service, son usage, les règles de diffusion des logiciels, etc. Cela permet à l'établissement de mettre en œuvre une politique propre concernant les logiciels de la recherche.

Enfin, d'un point de vue pratique, disposer de sa propre forge permet de centraliser soit le développement des logiciels, soit d'exposer l'ensemble de sa production logicielle en y installant systématiquement un miroir des projets hébergés sur d'autres forges. L'établissement peut ainsi avoir une vision globale de sa production logicielle.

On peut se demander pourquoi tant d'établissements du supérieur ont installé leur propre forge alors qu'une forge nationale, SourceSup, existe. Parmi les arguments qui peuvent être avancés, on peut citer son évolution technique hétérogène, l'absence de voix des utilisateurs dans le comite de pilotage, ou encore l'absence d'une stratégie de communication dans les deux sens (de SourceSup vers les utilisateurs, et des parties prenantes vers SourceSup). Cette trajectoire peut être mise en regard de l'histoire des autres forges, comme Inria dans l'Enseignement supérieur et la Recherche, ou OW2 et Eclipse dans les communautés libres. À Inria, après qu'une nouvelle forge basée sur GitLab a été mise en place, la forge Inria basée sur GForge a été arrêtée en décembre 2020, après une période de « tuilage » visant à assurer la migration des projets. Depuis janvier 2023, Inria dispose de deux instances de GitLab : une instance pour ses projets collaboratifs publics et une autre pour ses projets internes. Il en est de même pour OW2 : sa forge historique basée sur GForge a été remplacée par une instance de GitLab en 2018. Plus récemment, Eclipse a décidé de migrer

vers GitLab sa plateforme technique éprouvée combinant Git, Gerrit et BugZilla. Denis Roy, de la fondation Eclipse (Roy, 2022), a justifié cette migration par les arguments suivants :

> *Kids born in 2005 are turning 17 years old this year. They've likely been using Eclipse at school. They've been using Eclipse at home to learn to code. They've been using Eclipse to hack Minecraft mods. And they use GitHub to pull code from. Some may even know how to fork a repo and submit a Pull Request.*

> *If Eclipse projects want any hope of drawing those fresh young minds into their open source world, and turning casual explorers into productive contributors, it needs to be as simple as pulling a Minecraft mod. It needs to be on GitHub, or on a modern stack that works just like it, such as GitLab.*

Pour résumer, les pratiques de développement ont évolué de façon significative, avec le passage rapide en quelques années des forges orientées utilisateurs aux forges orientées développeurs qui ont mis au centre de l'attention le code source et facilité l'adoption massive de l'approche de contribution via les *forks* et les *pull requests* ou *merge requests*.

Observation 8

Un défi majeur pour une forge nationale est donc de surveiller activement les évolutions des usages et des solutions techniques, et d'engager les transformations nécessaires pour permettre d'offrir à tous les utilisateurs un environnement familier et cohérent avec les outils qu'ils utilisent lorsqu'ils collaborent sur d'autres projets logiciels.

C'est une tâche difficile, parce que, comme il a été dit, le développement logiciel est une activité qui dépasse largement la sphère de l'Enseignement supérieur et de la Recherche : les évolutions peuvent survenir à tout moment et les technologies évoluent vite.

En l'absence d'une stratégie ambitieuse dans ce sens, et au vu de la facilité d'installer des forges auto-hébergées récentes, on aboutit à la situation de fragmentation actuelle, caractérisée par une multitude de forges éparpillées dans les différents établissements.

3.5.3. Difficulté d'interaction avec la société

La principale limitation des forges disponibles actuellement dans l'Enseignement supérieur et de la Recherche est le fait que le public de

ces forges (c'est-à-dire les personnes pouvant créer un compte) est limité : la plupart des instances de forges disponibles ne permettent pas à une personne extérieure à l'Enseignement supérieur et de la Recherche de créer elle-même un compte sur ces plateformes. Il existe donc un frein à l'interaction avec la société.

Si certaines de ces forges permettent la création de comptes externes, elles sont souvent difficiles d'accès (par exemple, pour `gitlab.inria.fr` , un compte externe doit être « parrainé » par une personne membre d'une équipe-projet Inria) et limitées (le compte externe GitLab ne permet pas de créer ses propres projets). Il est donc souvent impossible, ou à tout le moins très difficile, de proposer des changements avec ce genre de compte, car cela nécessite d'effectuer un *fork* du projet original, et donc de créer un projet propre sur la forge. Rapporter un bogue nécessite d'avoir obtenu un compte au préalable, ce qui peut être rédhibitoire.

Cela peut aussi avoir des conséquences fâcheuses avec d'autres communautés du logiciel libre de recherche. Par exemple, le Journal of Open Source Software, un journal qui évalue et publie environ 400 papiers courts et les logiciels correspondants par an nécessite que le logiciel soumis "soit hébergé à un endroit où les utilisateurs peuvent créer des tickets et proposer des changements de code sans validation manuelle ou paiement pour la création de compte". Cette politique a été mise en place car JOSS est intéressé par la mise en avant de logiciels qui peuvent être utilisés et maintenus par une large communauté. Un de ses critères d'évaluation concerne les lignes directrices de la communauté, et cela inclus "il doit y avoir des règles claires pour des tiers qui souhaiteraient : contribuer au logiciel ; remonter des dysfonctionnements ; demander de l'aide". Ces activités ne peuvent se faire si la forge ne permet pas une inscription automatique gratuite et ouverte à tous.

Par ailleurs, des projets tels que l'« Extreme-scale Scientific Software Stack » (E4S) intègrent dans leur politique la nécessité de pouvoir soumettre des contributions: « Public Repository: Each E4S member package will have a public repository, for example at GitHub or Bitbucket, where the development version of the package is available and **pull requests can be submitted** ». Le PEPR NumPEx envisage de s'inspirer de cette exigence pour faciliter l'intégration de sa pile logicielle, et considère notamment que `gitlab.inria.fr` n'est pour le moment pas assez ouvert pour ses besoins.

> L'approche d'OW2 est pragmatique : elle permet à chacun de créer un compte sur sa plateforme, mais limite par défaut à deux le nombre de créations de nouveaux projets. Cela permet à un utilisateur d'un logiciel OW2 d'interagir facilement avec le système de tickets et de

> pouvoir proposer une contribution de code. Les gestionnaires de la forge ont pu constater que cette ouverture par défaut n'est pas une source de création de comptes fantômes. OW2 délègue cependant la création de comptes à son annuaire, et pas directement à sa forge.

> RENATER propose la création d'un compte réseau universel (CRU) pour permettre aux personnes extérieures à l'Enseignement supérieur et de la Recherche d'accéder à ces services en acceptant ce fournisseur virtuel d'identité. Certaines forges permettent l'authentification à l'aide du compte CRU.

Observation 9

En général, les logiciels de forge tels que GitLab ne permettent pas de limiter la création de projets à de simples contributions (c'est-à-dire à des *forks* de projets de la plateforme). Il n'y a pas non plus de moyen d'empêcher un utilisateur de déposer des photos ou vidéos sur son espace de projet.
Or, permettre la création de projets à des personnes extérieures à l'établissement peut poser des problèmes juridiques, tels que de respect de la loi du 24 juin 2020 relative à la lutte contre les contenus haineux sur Internet.

Une approche parfois mise en œuvre est l'authentification distribuée EduGAIN / Shibboleth, qui automatise la création de comptes et évite de multiplier les étapes d'authentification. Il faut cependant que l'institution qui gère l'authentification fournisse les informations nécessaires à la création du compte (identifiant utilisateur, adresse courriel). Certaines institutions utilisent une liste blanche de tiers autorisés : leurs utilisateurs doivent alors explicitement demander l'ajout à cette liste pour pouvoir utiliser cette authentification. Sans cela, un message d'erreur plus ou moins cryptique sera retourné (« *Empty uid* », « *Email can't be blank* », « *Email is invalid* », voire « *500 Whoops, something went wrong on our end* »), déroutant complètement l'utilisateur. Il vaut mieux que l'institution utilise une page de consentement permettant à l'utilisateur de confirmer la demande immédiatement plutôt que d'avoir à deviner qu'il doit demander à sa DSI locale de l'ajouter à la liste blanche.

> En février 2023, le support de Shibboleth a été supprimé de GitLab 15.9 car le composant Ruby fournissant cette fonctionnalité n'était plus maintenu. Cela empêche en pratique les montées de versions de nombreuses forges de l'Enseignement supérieur et de la Recherche

> à moins de modifier le code de l'application. Un appel à la communauté a été lancé pour mettre à jour ce composant. Une solution de contournement est de déléguer la gestion de Shibboleth à un système d'authentification unique. Le support de Shibboleth a été rétabli en juin 2023, dans GitLab 16.1, grâce à la communauté.

La limitation d'accès à une communauté complique aussi la gestion de comptes utilisateurs vis-à-vis de l'évolution de leur statut (départ de ou arrivée dans l'Enseignement supérieur et de la Recherche, l'institution ou le laboratoire).

> La forge GitLab portée par l'infrastructure HumaNum étend cette approche « EduGAIN / Shibboleth » en permettant, grâce à son système d'HumanID, de s'authentifier, en plus d'eduGAIN, par son compte ORCID, HAL, Twitter, LinkedIN ou Google.

Un autre point compliqué à gérer, du fait de la possibilité de s'inscrire sur la plateforme sans validation, est celui des *spams*, qui sont mis en ligne via les fonctionnalités d'exemples de code et de tickets pour les projets publics.

> En pratique, OW2 indique ne pas souffrir de *spam* sur son instance, car cette plateforme recourt à une création de compte indépendante de GitLab. Pour la fondation Eclipse, la situation est plus compliquée, notamment concernant leur service Wiki.

Observation 10

Pour résumer, sur les multiples forges existantes dans l'Enseignement supérieur et de la Recherche, tant pour l'interaction entre personnels qu'avec la société, un problème essentiel à résoudre est la définition d'une politique d'accès cohérente qui maximise les interactions sans mettre en danger les infrastructures.

3.5.4. Niveau de support et besoin de confiance

Il peut exister une différence entre le niveau de support attendu d'une forge et le niveau de support fourni, notamment parce que ce support sera forcément comparé à celui des forges commerciales. Si dans la majorité des cas, le support fourni correspond au besoin, il peut arriver que des évolutions techniques doivent être mises en place pour rétablir des fonctionnalités face à l'évolution permanente de l'environnement technique.

> Par exemple, sur les forges GitLab, l'intégration continue basée sur des images Docker est impactée par les changements de politique des registres d'images Docker (Docker Hub, registre d'images GitLab) qui limitent le nombre d'accès par machine et par jour. Par conséquent, chaque jour, au bout d'un certain nombre d'appels au registre Docker par l'intégration continue, les connexions peuvent être refusées et les intégrations continues échouer. Pour contourner ce problème, les mainteneurs de chaque instance doivent installer un registre local, qui sert de cache et réduit les appels aux registres publics à accès limités. Cependant, ces solutions ne sont pas mises en place de manière systématique. Pour l'utilisateur, pas forcément au courant de ces subtilités techniques, il en résulte une impossibilité de travailler sereinement.

Observation 11

Plus généralement, il existe un besoin de confiance dans une plateforme qui soit robuste et pérenne, sur laquelle les usagers puissent compter dans le temps pour diffuser leurs travaux de recherche, quelle qu'en soit la forme.

3.5.5. Licence libre ou propriétaire ?

Les solutions disponibles pour l'auto-hébergement peuvent être diffusées sous licence libre ou sous licence propriétaire, dans ce dernier cas généralement avec plus de fonctionnalités ou de support. On peut citer par exemple la version « Ultimate » de GitLab, diffusée sous licence propriétaire, qui offre des fonctionnalités supplémentaires par rapport à la version diffusée sous licence libre.

La question du choix de la licence peut alors se poser : est-il avantageux d'utiliser la solution sous une licence non-libre pour bénéficier de plus de fonctionnalités ?

Observation 12

Choisir un logiciel de forge sous licence propriétaire est un frein à l'ouverture de la forge. En effet, les licences d'utilisation non-libres sont généralement porteuses de certaines restrictions, telles qu'un nombre de comptes maximal et un périmètre d'utilisation limité.

Le choix d'une version payante peut cependant résulter de la volonté assumée de fournir aux entités éditrices de ces outils les moyens de pour-

suivre leur activité, gage de pérennité des solutions qu'elles proposent et donc des forges qui les utilisent. La fourniture sous licences fermées de certains de leurs codes et les modèles économiques « freemium » mis en place sont en effet pour ces entités le moyen de disposer de revenus récurrents à même de garantir les emplois de leurs développeurs.

La suite de ce chapitre va s'attacher à analyser l'usage courant de ces forges en mettant en avant trois grandes difficultés rencontrées dans leur mise en œuvre, à savoir l'organisation de la forge et de son cycle de vie, son périmètre et la gestion du droit d'auteur et des licences.

4 | Points d'attention sur les forges

Lors de notre étude, nous avons noté un certain nombre de points d'attention lorsque l'on met en place une forge. Il n'y a pas de bonne ou de mauvaise façon d'utiliser une forge. Il s'agit plutôt de mettre en cohérence le fonctionnement des forges et leurs objectifs.

4.1. Vitrine ou simple outil ?

Il y a deux façons orthogonales de considérer le fonctionnement de ces forges.

La première est de considérer qu'il s'agit principalement d'outils, qui sont mis à disposition de la communauté. Il s'agit en l'espèce de favoriser les bonnes pratiques en donnant accès à des outils de qualité pour la gestion des projets. Cette vision semble correspondre à celle de la majorité des forges de l'Enseignement supérieur et de la Recherche.

La seconde, non exclusive, est de considérer que les projets publics sont la vitrine de la communauté, et que cette dernière a donc le droit et le pouvoir de décider ce qui est rendu public. Cette vision se retrouve essentiellement au sein des communautés libres (Apache, Eclipse, OW2[1]).

Cependant, au sein de nombreuses forges, la création de projets publics n'est pas « validée ». On y retrouve donc des projets publics allant de « *hello word* » à une grosse application maintenue depuis des années. Cette hétérogénéité constitue un problème tant vis-à-vis de l'objectif de proposer une vitrine mettant en avant des projets d'une certaine « qualité », que de celui de lister la production logicielle d'une communauté. En revanche, ce n'en est pas forcément un si l'objectif principal est de favoriser l'usage d'un outil et des méthodes associées.

Observation 13

L'absence de contrôle (ou le faible contrôle) des créations de projets peut rendre leur gestion compliquée. Il est difficile, voire impossible, pour les administrateurs de déterminer quels projets doivent être conservés ou pas.

[1] OW2 permet cependant la création de deux projets personnels publics pour chaque utilisateur.

> Pour gérer plus finement les contenus d'une forge, il semble nécessaire de définir un processus et un cycle de vie des projets, par exemple en définissant un niveau de maturité (comme le Market Readiness Level proposé par OW2 par exemple). Chez Eclipse ou Apache, il existe la notion de « projet en incubation » pour les logiciels qui évoluent beaucoup. Eclipse propose également la notion de « train » qui inclut des briques a priori matures. Cette forge met également en œuvre la notion de « mentor », dont le rôle est de valider la création d'un projet.

Par ailleurs, et sans trancher entre la notion de simple outil ou de vitrine, il convient de noter que les forges, en regroupant les développements d'une communauté, d'une unité de recherche ou d'une tutelle, aident à suivre et à pérenniser l'activité de production logicielle de celles-ci. Les constructions de sites de référencement comme code.gouv.fr ou de l'archive universelle des logiciels Software Heritage se sont appuyées sur la fonctionnalité d'indexation offerte par les forges.

> L'université de Strasbourg propose une approche intermédiaire sur son instance GitLab en permettant à tous les usagers de créer des projets publics mais en localisant dans un groupe particulier (community) les développements « officiels » de l'université.

4.2. Organisation des projets

Observation 14

L'une des principales difficultés dans la gestion d'une forge est de savoir comment organiser les projets. Une solution peut être de créer un espace dédié pour chaque sous-structure (projet de recherche, équipe ou laboratoire), qui aurait la charge de la création de ses projets dans cet espace. Cette solution induit deux autres problèmes : où placer les projets communs aux sous-structures ? Comment gérer le cas des sous-structures qui n'ont pas de compétences ou de ressources internes pour assurer cette mission ?

> Par exemple, au niveau de l'IRD, la gestion s'effectue en premier lieu par UMR, celles-ci déclarent ensuite des sous-projets. Sur demande argumentée, des racines peuvent être déclarées pour des projets de longue durée inter-UMR. Les projets sont accessibles par défaut aux utilisateurs connectés. L'IRD tend vers un principe d'ouverture aux

> personnes connectées dès le démarrage des projets afin de susciter des collaborations le plus en amont possible. Cette politique est en cours de réalisation mais intégrera probablement ces aspects.

4.3. Gestion du droit d'auteur

Si les développeurs du logiciel ont tous le même employeur, ou appartiennent tous à la même structure multi-tutelles, il n'existe pas de problème particulier pour la gestion des droits d'auteur.[2]

À l'inverse, dès que l'on intègre du code qui vient d'un développeur extérieur, il convient de se préoccuper de la gestion de ses droits d'auteur. Cela s'effectue généralement de deux manières : de façon légère, au moyen des Developer Certificate of Origin (DCO), ou de façon plus encadrée, au moyen des Contributor License Agreement (CLA). Indépendamment de la solution choisie, se pose la question du moment auquel le faire. Il faut que la situation soit claire au moment de la contribution du code, car cette information constitue un élément de décision pour l'intégration ou pas du code dans le logiciel.

Le demander à l'inscription sur la forge risque de décourager des personnes qui souhaitent simplement signaler un défaut. De plus, il faut vérifier régulièrement que le document choisi est toujours valable (notamment, parce que l'employeur du contributeur peut changer). Chez Eclipse, les contributeurs doivent revalider régulièrement leur Eclipse CLA.

Le demander avant une première contribution, par exemple pour pouvoir créer une MR/PR, peut faire peur à un développeur qui n'est pas habitué à ce procédé.

Le demander avant d'intégrer le code dans le projet est une pratique courante : le développeur sait à ce moment-là que son code est prêt à être intégré, et qu'il manque seulement pour cela le versant « administratif » du processus. Lorsque la contribution est extrêmement petite, ce processus peut être contourné, et la correction effectuée par les développeurs principaux, en mentionnant le contributeur dans le texte du commit.

4.4. Gérer un projet sur plusieurs forges

Il est possible de synchroniser des forges auto-hébergées (de l'Enseignement supérieur et de la Recherche) avec des forges commerciales (gitlab.com

[2] On suppose ici que l'employeur bénéficie du transfert de la titularité des droits des différents contributeurs, et que les employés sont autorisés à publier en *open source* le code qu'ils produisent.

ou github.com). Cependant, on perd alors la centralisation des informations. Il est alors pertinent de désactiver le système de tickets publics, afin de conserver les tickets sur la forge locale, si la forge locale permet l'inscription libres de contributeurs externes (qui peuvent créer des tickets, mais pas proposer de contributions de code). Cette solution n'est cependant pas pratique pour les contributeurs, qui doivent utiliser deux comptes : un pour les alertes (tickets, rapports de bogues, et suggestions) et l'autre pour les contributions (de code, de documentations). Il est d'autant plus difficile de s'astreindre à cette discipline lorsqu'il existe un renouvellement important des contributeurs dans les projets de recherche.

> La fondation Eclipse utilise plusieurs forges (l'une auto-hébergée, d'autres commerciales). Même si un miroir est mis en place entre les forges, chaque projet est géré sur une seule forge, pour éviter les problèmes évoqués.

Pour faire face à cette limitation, il faut noter des initiatives en cours visant la fédération des forges. Par exemple, on peut citer les travaux réalisés dans le cadre du projet open-source Forgejo, une alternative communautaire à GitLab et GitHub, visant à spécialiser le protocole ActivityPub issu des travaux plus génériques autour de la fédération des univers Fediverse avec ForgeFed. Une implémentation de ce standard dans Forgejo devrait voir le jour en 2024. Le support de ce protocole est aussi en cours d'implémentation dans GitLab. L'initiative ForgeFriends va également dans ce sens.

Nous ne pouvons pas conclure ce chapitre sans souligner un besoin montant qui devient de plus en plus présent et important, l'intégration continue.

4.5. Toujours plus de services en intégration continue

La gestion d'un projet logiciel n'est pas la seule fonctionnalité attendue d'une forge. Offrir une documentation à jour est important, et disposer de moyens de publication de sites web à partir d'une forge est un atout supplémentaire (par exemple, le service « *GitHub/GitLab/SourceHut Pages* »). De plus, de nombreux outils basés sur l'analyse du contenu des dépôts peuvent s'avérer importants pour la maintenance du logiciel :

- analyse de la compatibilité juridique des licences du logiciel et de ses composants ;
- détection de composants possédant des vulnérabilités connues ;
- détection de vulnérabilités dans le code produit ;

- détection de mauvaises pratiques de développement dans le projet ;
- etc.

La plupart de ces fonctionnalités sont basées sur la possibilité de faire de l'intégration continue, c'est-à-dire de déclencher l'exécution de programmes en fonction de certains événements ou sous certaines conditions, comme par exemple à chaque mise à jour du code.

> Le besoin de sobriété énergétique du numérique implique que cette intégration continue soit configurée de manière à limiter les déclenchements inutiles. Des tests rapides sont généralement exécutés à chaque changement, et des traitements par lots peuvent être exécutés à des temporalités différentes selon les outils utilisés : chaque nuit ou chaque semaine par exemple, voire chaque mois. Un filtrage des tests lancés peut également être opéré selon les fichiers effectivement modifiés.

Il faut noter que les machines utilisées pour gérer l'intégration continue partagée d'une instance GitLab de l'Enseignement supérieur et de la Recherche sont généralement plus puissantes que celles utilisées par l'instance GitLab elle-même. En effet, si une modification de code va mobiliser le serveur GitLab quelques secondes au maximum, son intégration continue peut nécessiter plusieurs minutes, voire plusieurs heures, d'exécution. C'est donc une fonctionnalité qui passe difficilement à l'échelle sans une architecture adéquate, permettant d'adapter les ressources disponibles à la demande, et donc demande des moyens conséquents.

De plus, mettre en œuvre du déploiement continu, c'est à dire la mise en production automatique du logiciel dans un environnement donné, nécessite une architecture plus sophistiquée, qui permet d'accéder de manière sécurisée à ces différents environnements dynamiques.

La solution préconisée à l'heure actuelle se base généralement sur l'utilisation de conteneurs qui permettent de déployer des ressources d'intégration continue à la demande, et d'en faciliter la mise à niveau.

Des questions techniques supplémentaires doivent être traitées, telles que la signature des logiciels pour permettre leur installation sur des systèmes d'exploitation récents sans avoir à changer leur niveau de sécurité. Mutualiser cet effort permettrait de diffuser plus facilement dans la société des logiciels issus de la recherche.

Observation 15

Être capable de maintenir une telle architecture sur la durée nécessite des ressources et des compétences particulières, qu'il est pertinent de mutualiser.

Proposer des ressources d'exécution, souvent appelées *runners*, pour l'intégration continue pose des questions de sécurité et de risque de mésusage. Les forges commerciales ont réduit ou simplement désactivé les comptes qui permettent d'utiliser gratuitement du temps CPU sur des *runners*.

5 | Récapitulatif des solutions

Les matrices SWOT suivantes résument les forces et faiblesses des différents types de forges actuellement disponibles pour diffuser du logiciel libre ou tout autre artefact issu de la recherche, du point de vue de l'usager de ces forges.

5.1. Forges externes commerciales

On considère ici les forges telles que github.com, gitlab.com, bitbucket.org, sr.ht, etc.

Forces

- Intégration des fonctionnalités « tout en un »
- Gratuité d'un ensemble de fonctionnalités de base
- Disponibilité acceptable pour la plupart des projets
- Pour GitHub : une ergonomie devenue familière
- Pour sr.ht : un support humain très réactif
- Visibilité internationale
- Communauté internationale

Faiblesses

- Dépendance à l'égard de la politique commerciale de la société
- Pas de contrôle sur les ressources allouées aux fonctionnalités
- Pas de possibilité d'être associé aux décisions stratégiques
- Non souveraineté
- Pas de garantie de pérennité

Opportunités

- Nombreuses ressources disponibles pour la formation à ces outils

Menaces

- Dépendance à la réglementation du pays ou siègent ces sociétés

5.2. Forges externes communautaires

On considère ici les forges de fondations comme Apache, Eclipse, OW2 ou la FSF.

Forces

- Fonctionnalités
- Disponibilité
- Support
- Pérennité
- Visibilité au niveau de la communauté
- La communauté elle même

Faiblesses

- Nécessité d'acceptation du projet par la communauté
- Non couverture de toutes les thématiques par les communautés de logiciels libres

Opportunités

- Promotion de son projet dans un écosystème
- Bénéfice d'un accompagnement méthodologique
- Bénéfice de la confiance accordée à la communauté pour son propre projet

Menaces

- Dépendance à la réglementation du pays ou siègent ces fondations/associations

5.3. Forges auto-hébergées à l'échelon local

Forces

- Fonctionnalités sur mesure
- Disponibilité sur mesure
- Souveraineté
- Résilience
- Support (dépend des instances)
- Pérennité (dépend des instances)
- (Proximité)
- Visibilité au niveau de l'institution
- Communauté au niveau de l'institution

Faiblesses

- Disponibilités (dépend des instances)
- Support (dépend des instances)
- Multiplicité des solutions disponibles
- Difficulté d'accès en dehors de l'institution porteuse

Opportunités

- De facto, catalogue institutionnel de logiciels
- Mise en œuvre d'une politique institutionnelle de développement des logiciels de recherche
- Expertise maintenue au niveau de l'institution
- Diffusion de bonnes pratiques

Menaces

- Evolution de la solution choisie (périmètre des fonctionnalités sous licence libre vs fonctionnalités devenant payantes)
- Expertise difficile à trouver nécessaire pour maintenir la solution au niveau de l'institution

5.4. Forge auto-hébergée à l'échelon national

Forces

- Fonctionnalités
- Disponibilité
- Support
- Souveraineté (projets publics)
- Pérennité
- Sobriété
- Visibilité nationale
- Communauté nationale

Faiblesses

- Souveraineté (projets privés)
- Ouverture en dehors du cadre national

Opportunités

- De facto, catalogue national de logiciels
- Mise en œuvre de préconisations nationales pour le développement de logiciels de recherche
- Mutualisation de l'effort de maintenance
- Soutien par une politique publique globale

Menaces

- Evolution de la solution choisie (périmètre des fonctionnalités sous licence libre vs fonctionnalités devenant payantes)

6 | Conclusion

Les pratiques de développement logiciel ont profondément évolué, avec la généralisation de l'usage d'outils qui permettent de *fluidifier grandement la collaboration* sur des projets logiciels. Cela a progressivement conduit à transformer les forges en *réseaux sociaux de développeurs*, et la facilité d'intégrer la contribution de collaborateurs, même occasionnels, est devenue un *facteur clé pour accompagner l'essor* des logiciels libres et la création de communautés autour d'eux. Au delà du logiciel, les forges sont aussi utilisées pour la rédaction d'articles ou la gestion communautaires de données ou de modèles.

Dans ce contexte, le monde de la recherche a besoin d'une forge *a minima* ouverte sur l'ensemble de l'Enseignement supérieur et de la Recherche, et idéalement ouverte sur la société toute entière.

S'il existe bien une forge nationale pour l'Enseignement supérieur et la Recherche, SourceSup, son évolution fonctionnelle a divergé des pratiques d'une bonne partie des développeurs de l'Enseignement supérieur et de la Recherche, et aucune évolution majeure n'est prévue à ce jour ; sous sa forme actuelle, elle ne répond donc plus aux besoins exprimés.

La disponibilité de solutions sous licences libres pour installer des forges qui intègrent les fonctionnalités attendues, et leur facilité d'installation et de maintenance, ont fait qu'il existe actuellement au sein de l'Enseignement supérieur et de la Recherche de multiples forges maintenues par des individus, des équipes, des laboratoires ou des établissements. Sur ces forges, tant pour l'interaction entre personnel de l'Enseignement supérieur et de la Recherche qu'avec la société, un des problèmes essentiels à résoudre est la définition d'une politique d'accès cohérente qui maximise les interactions sans mettre en danger les infrastructures.

Pour résoudre ces difficultés, de nombreux logiciels libres issus de la recherche française ont dû migrer ailleurs. Pour des projets académiques qui ont obtenu une certaine visibilité, ou qui ont une stratégie de valorisation bien définie, une alternative est offerte par les forges maintenues par des communautés de logiciels libres, mais cette piste n'est pas exploitable pour des logiciels qui sont à l'état de preuves de concepts dans les laboratoires.

Beaucoup de projets se retrouvent donc sur des forges commerciales, qui permettent la création de projets sans restriction, mais leur usage néces-

site que des personnels de l'Enseignement supérieur et de la Recherche acceptent individuellement leurs conditions d'utilisation, pratique qui peut poser problème, d'autant que les forges commerciales n'offrent aucune garantie de pérennité.

Tous ces enjeux doivent être pris en compte de façon globale.

D'un coté, s'expriment les besoins des personnels de l'Enseignement supérieur et de la Recherche qui développent des logiciels de recherche :

- une forge dans laquelle ils puissent avoir confiance : robuste et pérenne, sur laquelle il doit être possible de compter dans le temps pour diffuser ses travaux de recherche, quelle qu'en soit la forme ;
- une forge qui évolue avec l'état de l'art : il doit être possible d'y retrouver des outils d'un niveau comparable à ceux mis à disposition des développeurs sur d'autres forges ;
- une forge qui permette de construire et faire croître facilement la communauté de contributeurs : l'inscription et la création de projets doit être facile, et son usage simple et ergonomique ;
- une gestion simplifiée des démarches liées à la propriété intellectuelle.

De l'autre, se posent les défis à relever par qui doit fournir ces services :

- l'absence de contrôle (ou le faible contrôle) des créations de comptes et de projets sur une forge est importante pour la création de communautés, mais peut rendre leur gestion compliquée, et induit des risques techniques et juridiques pour les infrastructures. Il devient difficile, voire impossible, pour les administrateurs de déterminer quels projets doivent être conservés ou pas, et faire le tri entre usages normaux et abusifs peut demander des ressources conséquentes ;
- le suivi de l'évolution technologique, et des usages, nécessite une activité de veille, et la capacité opérationnelle de déployer des nouvelles solutions, y compris à titre expérimental, tout en maintenant les anciennes en production, y compris dans le cadre de « tuilages » ;
- fournir un service à une communauté très large nécessite de réfléchir à l'organisation des projets sur cette forge, en particulier pour des projets multi-tutelles, et à la mise en œuvre pratique d'une telle politique dans des laboratoires ayant des compétences hétérogènes ;
- faciliter les collaborations entre établissements de l'Enseignement supérieur et de la Recherche pourrait nécessiter la mise en place d'une gestion des droits mutualisée au sein d'une fédération d'identités.

Enfin, ce sujet possède des enjeux stratégiques majeurs : les forges

logicielles commerciales n'apportent pas de garantie de pérennité, et les plus populaires d'entre elles sont soumises à des juridictions extra-européennes. Chaque base de code qui est déposée sur une forge commerciale offre un aperçu très en amont sur les activités de recherche et industrielles des personnels des pays déposants (dans le cadre des projets privés), vient enrichir les données sur lesquelles la plateforme s'appuie pour l'apprentissage des outils d'aide à l'écriture de code (OpenAI Codex/GitHub copilot, GitLab suggestions, etc.), et donc contribue à la domination des GAFAM sur le numérique. La maîtrise de l'usage du code issu de nos laboratoires constitue en cela un enjeu majeur de souveraineté.

Il est urgent d'apporter une réponse réfléchie et coordonnée à l'ensemble des besoins et défis exprimés. Déployer une forge auto-hébergée est relativement facile, mais la maintenir à l'état de l'art sur la durée en termes d'infrastructure et de fonctionnalités, mettre en place des politiques d'accès et d'organisation des projets, suivre et accompagner les usages, et éviter les abus nécessite des ressources et des compétences particulières, qu'il est nécessaire de mutualiser.

A | Liste des forges « publiques » auto-hébergées de l'ESR

La très grande majorité des forges auto-hébergées étant des instances de GitLab, nous ne précisons que le nom des autres logiciels utilisés.

A.1. Établissements

Forge	Identification	Hors ESR	Intégration continue	Autres services
CEA codev-tuleap.cea.fr	LDAP CEA			Forge basée sur Tuleap
CentralSupelec gitlab-research.centralesup	LDAP Centrale SupElec (+ invités)			
CIRAD gitlab.cirad.fr	Renater	Compte externe pour les personnels de l'ESR	GitLab CI	GitLab Pages, SonarQube
CNRS src.koda.cnrs.fr	CNRS (Janus)	Non	Oui, mais pas de *runner* partagé	Non
forge.in2p3.fr	EduGAIN	utilisateurs externes		Forge basée sur Redmine
gitlab.in2p3.fr	EduGAIN	utilisateurs externes	GitLab CI	GitLab Pages
plmlab.math.cnrs.fr	Renater mais compte PLM nécessaire	Invitation	CI/CD avec *runners* partagés	Pages avec domaines personnalisés, gestionnaire d'artefact, registre d'images Docker

Forge	Identification	Hors ESR	Intégration continue	Autres services
gitlab.mbb.cnrs.fr	LDAP MBB + invitations	Non	Oui, mais pas de *runner* partagé	registre d'images de conteneurs
gitlab.huma-num.fr	EduGAIN, ORCID, HAL	LinkedIn, Twitter et Google	GitLab CI	GitLab pages
Centre de Calcul de l'Ecole Centrale de Lyon gitlab.pmcs2i.ec-lyon.fr	Annuaire Interne + Invités	Non	Gitlab CI/CD	
Département biologie ENS Lyon gitbio.ens-lyon.fr				
EURECOM gitlab.eurecom.fr				
IFREMER forge.ifremer.fr				Forge basée sur FusionForge
IMT gitlabev.imtbs-tsp.eu	Annuaire Interne + Shibboleth IMT	Non	CI	
INRAE forge.inrae.fr	Renater	CRU	GitLab CI/CD (4 runners partagés)	GitLab Pages, Gestionnaire d'artefacts, Registre d'images Docker, Mattermost
INRIA gitlab.inria.fr	Inria	invité·es externes parrainé·es	GitLab CI	GitLab Pages, registre d'images Docker
INSA Rennes gitlab.insa-rennes.fr				
IRD forge.ird.fr	Renater	CRU - autres en cours	En cours	GitLab pages

Forge	Identification	Hors ESR	Intégration continue	Autres services
IRSTEA gitlab.irstea.fr	LDAP	utilisateurs externes	Gitlab CI/CD	
ISAE-SUPAERO gitlab.isae-supaero.fr	LDAP	utilisateurs externes	GitLab CI	GitLab Pages à l'étude
Institut Pasteur gitlab.pasteur.fr				
OSU Phythéas gitlab.osupytheas.fr				
TelecomParis gitlab.telecom-paris.fr				
U. Bordeaux gitub.u-bordeaux.fr	Personnels	Après validation, limités	Non	
gitlab.emi.u-bordeaux.fr	Personnels, étudiants	Non	Oui	CI, GitLab pages
U. Bretagne Sud gitcdr.univ-ubs.fr/				Forge basée sur Forgejo
U. Caen git.unicaen.fr	Renater		GitLab CI	
U. Gustave Eiffel gitlab.univ-eiffel.fr	Adresse email Gustave Eiffel	Non	CI/CD	Pages
U. Grenoble gricad-gitlab.univ-grenoble-alpes.fr	UGA	Oui : sur simple inscription (sans validation) mais avec des droits limités.	GitLab-CI (avec *runners* partagés)	gitlab pages (pages privées et publiques), registre de container
U. La Rochelle gitlab.univ-lr.fr	Annuaire	Invitation	Non	
U. Lille gitlab.univ-lille.fr	RENATER			
U. Limoge git.unilim.fr	Annuaire interne	Non	CI	

Forge	Identification	Hors ESR	Intégration continue	Autres services
U. Littoral gogs.univ-littoral.fr	LDAP ULCO	invités	Non	Forge basée sur Gogs
U. Lorraine gitlab.univ-lorraine.fr	LDAP établissement	Compte invité numérique	CI/CD avec runners partagés	Registre de container
U. Lyon1 forge.univ-lyon1.fr	CAS univ Lyon 1			
U. Montpellier 2 gitlab.mbb.univ-montp2.fr	LDAP + invités			
U. Nantes gitlab.univ-nantes.fr	Annuaire Interne	Invitation	CI/CD	Pages, gestionnaire d'artefact, registre d'images Docker
UFR Math Paris Cité gitlab.math.univ-paris-diderot.fr				
U. Paris Saclay gitlab.dsi.universite-paris-saclay.fr	Annuaire interne	utilisateurs externes	GitLab CI	GitLab pages, gestionnaire d'artefacts
U. Pau git.univ-pau.fr	LDAP Université	GitLab CI (runners privés)		
U. Reims gitlab-mi.univ-reims.fr	LDAP			Départements mathématiques et informatique, orienté enseignement
U. Reims romeogit.univ-reims.fr	LDAP		GitLab CI runners privés	

Forge	Identification	Hors ESR	Intégration continue	Autres services
U. Strasbourg gitlab.unistra.fr	Auto-inscription	Non	CI/CD	GitLab Pages, Security Testing, Analytics, Error Tracking (Sentry)
U. Tours scm.univ-tours.fr				
UTC gitlab.utc.fr				
ESRF gitlab.esrf.fr	Annuaire interne	Invitation, création de compte ESRF	CI/CD	Pages, gestionnaire d'artefact, registre d'images Docker
PhyMath@X gitlab.labos.polytechnique.fr				

A.2. Laboratoires

Forge	Identification	Hors ESR	Intégration continue	Autres services
CCSD gitlab.ccsd.cnrs.fr				
CNRM opensource.umr-cnrm.fr				Forge basée sur Redmine
CRIStAL gitlab.cristal.univ-lille.fr	LDAP CRIStAL		GitLab CI avec runners privés	
DI ENS gitlab.di.ens.fr				
IAS git.ias.u-psud.fr	LDAP interne du labo + invités	Non	CI	Gitlab Pages, registre d'images Docker
iCube forge.icube.unistra.f	??? + invités	Non	CI avec runners privés	Gitlab Pages, Mattermost
iLM cameleon.univ-lyon1.fr				
ILL code.ill.fr				
IP gitlab.ip.uca.fr				
IPSL forge.ipsl.jussieu.fr				Forge basée sur Trac
IRMA gitlab.math.unistra.fr	LDAP IRMA et comptes locaux	Sur invitation	GitLab-Ci avec runners partagés	GitLab Pages, Mattermost, Registre de containers, Gestionnaire d'artefacts
IRCAD git.ircad.fr				
IRISA gitlab.irisa.fr	Annuaire LDAP			Expérimental.
IRIT gitlab.irit.fr	Annuaire	Invités LDAP		

Forge	Identification	Hors ESR	Intégration continue	Autres services
ISIR git.isir.upmc.fr				
FRESNEL gitlab.fresnel.fr	Institut Fresnel + invités			
LAAS gitlab.laas.fr	Annuaire LDAP			
LAM gitlab.lam.fr				
LAMSADE git.lamsade.dauphine.fr				
LACL git.lacl.fr				
LEGI servforge.legi.grenoble-inp.fr				Forge basée sur Trac
LIMOS gitlab.limos.fr	Annuaire LIMOS	Invité dans l'annuaire	CI/CD	
LIP6 gitlab.lip6.fr	LDAP LIP6	Non	GitLab CI (sans runner partagé)	
LIPN depot.lipn.univ-paris13.fr	Annuaire LIPN + invités			
LIRIS gitlab.liris.cnrs.fr	LDAP LIRIS	Invitation	GitLab CI (partagés et spéci-fiques)	Pages, mattermost
LIRMM gite.lirmm.fr	Annuaire LIRMM + invités	GitHub et Bitbucket	GitLab CI, GitLab Pages	
LIS gitlab.lis-lab.fr				
LISN gitlab.lisn.upsaclay.f	Annuaire LISN + invités	Non	GitLab CI pour pages	GitLab Pages
LMGC git-xen.lmgc.univ-montp2.fr				

Forge	Identification	Hors ESR	Intégration continue	Autres services
MDLS gitlab.maisondelasir				
OBSPM gitlab.obspm.fr	LDAP	Non	CI/CD avec runners partagés	Pages, registre d'images Docker, mattermost
OCA forge.oca.eu				Forge basée sur Trac
OCA gitlab.oca.eu	eduGAIN (seuls les membres de l'OCA peuvent créer des projets)	Invitation	CI	Pages, gestionnaire d'artefact
Ortolang git.ortolang.fr				
SOLEIL gitlab.synchrotron-soleil.fr	SOLEIL SSO	GitHub, GitLab		
XLIM gitlab.xlim.fr	Annuaire XLIM	Invitation	CI	Gitlab Pages

B | Contributeurs

B.1. Membres du collège codes sources et logiciels

- Ludovic Courtes, Inria
- Roberto Di Cosmo, Inria/Université Paris Cité
- Sébastien Gérard, Université Paris-Saclay/CEA List
- Timothée Giraud, CNRS
- Jean-Yves Jeannas, Université de Lille/AFUL
- Nicolas Julien, IMT Atlantique
- Daniel Le Berre, Université d'Artois/CNRS
- Violaine Louvet, CNRS/GRICAD/Université Grenoble Alpes
- François Pellegrini, Université de Bordeaux/CNIL
- Nicolas Rougier, Inria/Université de Bordeaux/CNRS
- Francois Sabot, IRD
- Samuel Thibault, Université de Bordeaux

B.2. Invités

- Denis Arrivault, David Margery Inria
- Céline Blitz CIRAD
- Gérald Dherbomez CNRS INS2I
- Alban Espie-Guillon, Pierre-Yves Gibello, Antoine Mottier OW2
- Bastien Guerry DINUM
- Alexis Kauffmann DNE MENJ
- Philippe Krief Fondation Eclipse
- Christian Poirier INRAE
- Jean-Christophe Souplet OpenEdition

B.3. Communautés

- GDR GPL
- Réseau Calcul
- Réseau DevLog

C | Questionnaire

Le questionnaire suivant a été envoyé au GDR GPL et réseaux Calcul et DevLog en septembre 2022 avec une version préliminaire de l'état des lieux.

C.1. URL de la forge

Cette information permettra d'identifier de manière unique la forge dont il s'agit.

C.2. Par qui est proposée/maintenue cette forge ?

Veuillez sélectionner une réponse ci-dessous

- Une personne
- Une structure interne (équipe)
- Un laboratoire
- Une structure institutionnelle : UFR, école, université
- Autre :
- Sans réponse

C.3. Cette forge est elle publique ?

Une forge est considérée comme publique si la totalité ou même une partie des codes sources qu'elle héberge est accessible à tous sans être identifié.

Une forge privée (ou interne) nécessite d'être identifié pour pouvoir accéder aux projets.

Si vous donnez des informations sur une forge privée via ce questionnaire, elles ne seront pas publiées dans l'état des lieux, mais elles nous permettront d'avoir une idée du nombre et des fonctionnalités des forges privées.

- Oui
- Non
- Sans réponse

C.4. Comment s'identifie t'on sur cette forge ?

L'identification se fait généralement à l'aide d'un annuaire.

Certaines forges permettent l'utilisation de fédérations d'identité pour ce connecter (renater, eduGAIN).

Certaines permettent la création de compte, d'autres pas.

Cochez la ou les réponses

- Annuaire interne
- RENATER
- EDUGAIN
- Invitation
- Auto-inscription (création de compte libre)
- Autre :

C.5. Quels sont les services disponibles ?

Il existe de nombreux services annexes possibles autour de la forge. Le plus souvent, on y retrouve de l'intégration continue, de la gestion d'artefacts, d'images docker, des outils d'analyse statique de code comme Sonarqube, etc.

Cochez la ou les réponses

- Intégration continue
- "Gitlab pages"
- Déploiement continu
- Gestionnaire d'artefact
- Gestionnaire d'images docker
- Assurance qualité (sonarqube)
- Autre :

C.6. Comment sont organisés les projets ?

Une des difficultés des forges publiques est de gérer la façon dont sont organisés les projets : on ne souhaite pas forcément avoir un projet lié à une personne en particulier, mais plutôt à une équipe de recherche, un projet scientifique, voire un laboratoire de recherche.

Il est possible dans le cas là de créer une structure arborescente de projets.

Se pose alors le problème des projets transverses à plusieurs équipes, projets, laboratoires.

Vous avez l'occasion de décrire ici librement la façon dont sont organisés les projets logiciels sur cette forge.

C.7. Commentaire libre

Commentaire libre sur le document de travail.

D | Glossaire

C

commit unité de modification

F

forge outil de développement logiciel collaboratif

***fork*/divergence** évolution alternative du code source

G

génie logiciel champ de l'informatique s'intéressant à la gestion et au cycle de vie des projets logiciels

git outil de gestion de versions

I

intégration continue capacité pour une forge de permettre la construction automatique du logiciel depuis l'ensemble de ses sources et en fonction de certains paramètres

M

***merge request* (MR)** proposition de révision

P

plateforme site internet qui fait fonctionner un logiciel permettant d'avoir accès à des fonctionnalités spécifiques au travers d'un navigateur Web (exemple : plateforme de formation à distance)

***pull request* (PR)** synonyme de *merge request* (l'appellation peut être différente selon les plateformes)

S

Software Heritage initiative internationale visant à conserver pour l'Histoire les codes source des logiciels dont le code source est public

souveraineté Capacité d'un individu, d'un groupe ou d'un état à préserver ses données d'un accès et d'un usage sans contrôle par des personnes ou entités extérieures

T

ticket Déclaration en ligne d'un incident ou d'un dysfonctionnement, ou proposition d'amélioration du logiciel

Bibliographie indicative

Bangerth Wolfgang, Heister Timo. « What makes computational open source software libraries successful? » *Computational Science & Discovery* [En ligne]. novembre 2013. Vol. 6, n°1, p. 015010. Disponible sur : https://doi.org/10.1088/1749-4699/6/1/015010

Chan Denise. « Sunsetting Mercurial support in Bitbucket ». *BitBucket blog* [En ligne]. 2020. Disponible sur : https://bitbucket.org/blog/sunsetting-mercurial-support-in-bitbucket

Degeler Andrii. « Code collaboration platform GitLab acquires rival Gitorious, will shut it down on June 1 ». *The Next Web* [En ligne]. 2015. Disponible sur : https://thenextweb.com/news/gitlab-acquires-rival-gitorious-will-shut-june-1

Escamilla Emily, Klein Martin, Cooper Talya, Rampin Vicky, Weigle Michele C., Nelson Michael L. « The Rise of GitHub in Scholarly Publications ». In : Silvello G, Corcho O, Manghi P, Di Nunzio GM, Golub K, Ferro N, Poggi A, Éd. *Linking Theory and Practice of Digital Libraries*. Cham : Springer International Publishing, 2022. p. 187‒200.ISBN : 978-3-031-16802-4.

MENJ. *Stratégie du numérique pour l'éducation 2023-2027* [En ligne]. 2023. Disponible sur : https://www.education.gouv.fr/strategie-du-numerique-pour-l-education-2023-2027-344263

Pellegrini François, Di Cosmo Roberto, Romary Laurent, Janik Joanna, Hodenq Sacha, Coutanson Romane, Géroudet Madeleine. *Science ouverte – codes et logiciels* [En ligne]. 2022. Disponible sur : https://www.ouvrirlascience.fr/science-ouverte-codes-et-logiciels/

Raymond Eric S. *The cathedral and the bazaar - musings on Linux and open source by an accidental revolutionary (rev. ed.)*. [s.l.] : O'Reilly, 2001. ISBN : 978-0-596-00108-7.

Roy Denis. « Moving Eclipse projects to GitHub and GitLab ». *Denis Roy's blog* [En ligne]. 2022. Disponible sur : https://blogs.eclipse.org/post/denis-roy/moving-eclipse-projects-github-and-gitlab

Sharwood Simon. « GitLab plans to delete dormant projects in free accounts ». *The Register* [En ligne]. 2022. Disponible sur : https://resana.numerique.gouv.fr/public/document/consulter/3231705?slug=160133

Staff ARS. « Google to close Google Code open source project hosting ». *ARS Technica* [En ligne]. 2015. Disponible sur : https://arstechnica.com/information-technology/2015/03/google-to-close-google-code-open-source-project-hosting/

Wilson Greg, Aruliah D. A., Brown C. Titus, Chue Hong Neil P., Davis Matt, Guy Richard T., Haddock Steven H. D., Huff Kathryn D., Mitchell Ian M., Plumbley Mark D., Waugh Ben, White Ethan P., Wilson Paul. « Best Practices for Scientific Computing ». *PLOS Biology* [En ligne]. janvier 2014. Vol. 12, n°1, p. 1?7. Disponible sur : https://doi.org/10.1371/journal.pbio.1001745