Higher Education and Research Forges in France - Definition, uses, limitations encountered and needs analysis

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

Ouvrir la science !

Higher Education and Research Forges in France

Definition, uses, limitations encountered and needs analysis

The Committee for Open Science: Software and Sources Codes College

May 2023

Higher Education and Research Forges in France

Definition, uses, limitations encountered and needs analysis

The Committee for Open Science: Software and Sources Codes College

---

WG 2: Technical and social tools and good practice

Daniel LE BERRE (WG 2 co-chair)
University of Artois/CNRS (French National Scientific Research Centre)

Jean-Yves JEANNAS (WG 2 co-chair)
Lille University/AFUL (French Association of Free Software Users)

Roberto DI COSMO (college co-chair)
Inria (French National Institute for Research in Digital Science and Technology)/Paris Cité University

François PELLEGRINI (college co-chair)
Bordeaux University/CNRS/CNIL (French Data Protection Authority)

May 2023

DOI: 10.52949/37

Conception graphique : opixido

Overview

The first software forge, called SourceForge, was launched in 1999, and was designed to help open-source software developers build their software collaboratively and distribute it to their users. Since then, software forges have become vital tools for all software developers. They feature collaborative development tools (for monitoring code modifications, and managing user tickets, contributions and projects) and they industrialise the process of creating software from their source codes (compilation, automated tests, quality assurance and distribution of deliverables) and communications tools such as forums.

Software forges also act as social networks for developers. Whenever developers want to encourage people to use and make contributions to software, they need to come to a decision about which forge to choose based on the target audience and network. Targeting Higher Education and Research developers in France or abroad is one potential option. There are a number of identity federations such as RENATER or eduGAIN which have been providing long-term support for these collaborations. A number of Higher Education and Research forges provide access to these collaboration networks. Should a developer wish to open and share source codes coming from research with the wider society, there are two alternatives available to them - open-source community or commercial forges. Open-source community forges can be used to distribute open-source software within a community which has co-opted it. The challenge here lies in finding the right community for the software under development. Commercial forges boast many features with very few constraints, and often offer a range of services when the developed software is distributed under an open-source licence. These commercial forges include GitHub (owned by Microsoft), which is the most widely used, followed by BitBucket (owned by Atlassian) and GitLab (owned by GitLab Inc.).

Some forges, be they community-based or commercial, such as GitLab, can be self-hosted by Higher Education and Research Institutions, some of which have their own public forge. This report lists 40 of these types of forge as well as the forges for internal use only. These self-hosted forges are often easy to install, ranging from a simple executable for solutions such as Gogs, Gitea and Forgejo to a preconfigured software package integrated into Linux distribution for GitLab, for example. GitLab is basically a commercial forge (gitlab.com) based on open-source forge software that can be installed on

premise. GitLab Inc.'s financial model is based on selling licences for additional features to be used by online-service users or self-hosted forge administrators.

In reality, installing a self-hosted forge for internal collaborative development requires few human or material resources, and offers a wide range of solutions. However, as soon as developers want take this collaborative development externally, integrate solutions to industrialise software development and implement good development practice, more substantial efforts are needed, and the choice of solution may be led by different criteria such as the platform's popularity, its functionalities and how robust it is.

In Higher Education and Research, developers of supporting software and software based on research work can choose between a number of forges to host their software. The simplest solution is when their institution has its own forge, particularly if no interaction is needed outside the institution.

When wider interaction is required, communities developing research software often look to online commercial forges. This is reflected by the winners of the first french open science open-source research software award laureates, with 9 projects being hosted on GitHub and one project on SourceForge. The social network effect of "people go where most people are" and the international scope of the projects were the reasons for their selection. However, it really should be noted that commercial forges can suddenly disappear, as was the case with the Google forge, Google Code, which ceased operation after nine years presence in just a matter of months. The same thing happened with the Gitorious hosting solution. In addition, these forges have terms and conditions of use which each member must agree to as an individual, rather than on behalf of their institution.

Self-hosted forges are one way of mitigating this kind of problem. However, it may be the case that the solution selected is no longer being maintained, or no longer developed under an open-source licence. This is what happened with the SourceForge code, and it was maintained in a community version under the name "GForge", which has itself changed licences to enable it to be maintained in a community version under the name "FusionForge", to now end up now with an unmaintained software (the latest version of the software dates back to 2018).

Therefore, decisions around self-hosting and which forge to use are important. Of the 40 forges listed, 38 are GitLab platforms (the other two forges use Tuleap and Gogs respectively). GitLab's domination can be explained by how easy it is to install and maintain, and the wide range of functionalities which are available.

Hence the interest in having a specific Higher Education and Research forge operating at any level (institutional, national, European or international). In-

stitutional forges are the answer when software is being developed internally within an institution and an institutional forge already exists. In this case, the functionalities available and access to data are managed, but they offer little or no scope for development between multiple institutions. Where an institutional forge does not already exist or does not allow project owners to invite contributions from outside the institution, a national or European forge would provide an alternative to using commercial forges.

Throughout the mid-2000s, a french national forge, SourceSup, was set up by RENATER (which manage the national electronic communications network for technology, education and research) in order to get around these restrictions on interaction. However, this forge, which was a state-of-the-art platform when it was created, now only offers a set of tools that have fallen behind current development standards.

This report provides a comprehensive picture of the existing forges and practices in Higher Education and Research in France, and posits a number of observations and points of concern as regards the current situation.

Summary

Overview ii

Background 1

1. Introduction 2

2. About the importance of forges 3
- 2.1. Monitoring changes to the source code . . . 3
- 2.2. Steering the entire life cycle of the software . . . 4
- 2.3. Enabling and encouraging collaborative work . . . 4
- 2.4. Building software and analyzing its source code . . . 5
- 2.5. Looking beyond the source code . . . 6
- 2.6. Evolution of the target audience for forges . . . 6
- 2.7. Open-source research software . . . 7
- 2.8. Target audiences for forges within Higher Education and Research . . . 8

3. The outlook for Higher Education and Research forges 10
- 3.1. Major use of commercial forges . . . 10
- 3.2. Using open-source community forges . . . 11
- 3.3. The SourceSup national forge . . . 12
- 3.4. 40 self-hosted public Higher Education and Research forges . . . 14
- 3.5. Analysis of the situation . . . 15
  - 3.5.1. Chronological history of the forges referred to in this document . . . 15
  - 3.5.2. Why are there so many self-hosted forges? . . . 16
  - 3.5.3. Difficulties in interacting with society . . . 18
  - 3.5.4. Support levels and the need for trust . . . 20
  - 3.5.5. An open-source or a proprietary license? . . . 21

4. Points to consider when it comes to forges 23
- 4.1. A showcase platform or a simple tool? . . . 23
- 4.2. Project organisation . . . 24
- 4.3. Copyright management . . . 25
- 4.4. Managing a project across a number of forges . . . 25
- 4.5. More and more continuous integration services . . . 26

5. An overview of the solutions 28
- 5.1. Commercial external forges . . . 28

5.2. Community external forges 29
5.3. Local self-hosted forges 30
5.4. National self-hosted forges 31

6. Conclusion **32**

A. List of self-hosted public Higher Education and Research forges **35**
A.1. Institutions 35
A.2. Laboratories 38

B. Contributors **40**
B.1. Members of the Software and Sources Codes College 40
B.2. Guests 40
B.3. Communities 40

C. Questionnaire **41**
C.1. Forge URL 41
C.2. Who is providing / maintaining this forge? 41
C.3. Is this a public forge? 41
C.4. How do you log in to this forge? 42
C.5. Which services are available? 42
C.6. How are the projects structured? 42
C.7. Free comment 43

D. Glossary **44**

Selective bibliography **46**

Background

Version	Date	Comment
3	06/09/23	One more forge. Small updates. English version available.
2	04/06/23	Spelling and formatting fixes by readers
1	02/05/23	Initial version of the Committee for Open Science: Codes Software and Sources Codes College

We would recommend using tickets to send us your suggestions and any further information.

1 | Introduction

This report looks at the software forges used in higher education and research institutions in France (*Enseignement Supérieur et Recherche*, ESR). It focuses in particular on software engineering practices and needs, in order to get the best value from the software created through Open Science. This document aims to provide an initial overview of the software forges used in that context, and to identify the ways of raising the profile of software created via Open Science.

Software forges are mainly designed to manage software-engineering artefacts (such as the source code), the binary files and even the documentation over the entire life cycle. Forges can also be used to collaboratively draft scientific papers or documentation, or even write web pages with the help of tools for processing poorly structured text files (for example, markdown and asciidoc). These activities also require continuous integration with tools for processing and deploying source codes and documentation, and making them publicly accessible. This document does not just strictly focus on software itself along with its source codes and binaries, but also touches upon managing any artefacts produced and shared in order to produce high-quality, shareable and reusable software. Software forges are also used for the collaborative sharing of data and models. Consequently their use impacts the three pillars of open science.

It should be noted that this analysis only relates to the needs of higher education and research activity. It does not cover forge needs in education or the needs of information system tools for basic management of establishments. It primarily sets out to provide an overview of current practices and proposed actions to promote good development practices and to get the maximum value out of any software produced. These proposals for action to take are based on an analysis of the current limitations and an initial analysis of common needs.

This document is divided into three chapters and sets out the role of software forges. It provides an initial overview of Higher Education and Research forges in France and their use for people who are well acquainted with the world of software development. A number of initial areas of action are proposed, based on analysis of the limitations encountered by Higher Education and Research institutions in France.

2 | About the importance of forges

If you are developing software, you need to ensure that good practice is implemented. Irrespective of the characteristics of the software being developed, it is always easier to start by implementing good practice rather than trying to bring it in at a later stage. In any case, the quality of the software will be all the better for it. For example, an important aspect, as we will see later, is managing changes to the source code. These need to be automatically tracked with a version management system, such as Git, Mercurial or Subversion, for instance.

Many web platforms, often referred to as software forges, such as BitBucket, Gitea, GitHub, GitLab, Gogs or ForgeJo, simplify the process of implementing this good practice and help with developing better-quality software and with setting up contributor and user communities.

Forums for these platforms can be set up locally by research teams, laboratories and departments, or even by an affiliated institution or a research hub such as Huma-Num for the humanities and social sciences.

The communities on each platform can bring meaningful benefits and assist in finding bugs, adding new features and contributions to the documentation.

In order to harness the contributions of a community, the time and energy required to run it needs to be given over to it, providing documentation about the methods and opportunities for providing contributions, quickly assessing and managing the contributions made, and sharing the software development road map.

See (Pellegrini, Di Cosmo, Romary, Janik, Hodenq, Coutanson, et Géroudet, 2022) for a more detailed discussion of the subject.

2.1. Monitoring changes to the source code

Developing a piece of software and, in particular, writing its source code (known as "coding") is an iterative, incremental activity. The source code for software needs to change throughout its life cycle, not just when it is created. In fact, software changes over time. For example, there is corrective maintenance which makes changes to software by setting out bug fixes and upgrade maintenance, involving taking into account changes in the operat-

ing environment (such as changes in the supporting hardware, the operating system and the libraries used) or introducing new functionalities.

To factor in these changes, this process usually draws on tools which can record any changes made during each stage of the code's development, known as version control systems (VCS). One of the most used VCSs currently is Git.

So VCSs can be used to closely track every commit, to easily compare several branches and formalise specific software releases.

2.2. Steering the entire life cycle of the software

Developing high-quality software does not end at the code-writing stage. Indeed, code is a vital artefact coming from the software development process, but the specifications, design models, tests and developer and user documentation are just as important as artefacts, if developing high-quality, long-lasting codes that can be maintained and even reused is what is the objective. Hence, version control systems are one tool that can be used here, but they are not the only one. In order to manage the entire life cycle of software, it is often also important to manage user needs, record defects identified in the software, produce distributable versions of the software, launch tests regularly, produce documentation, and even more. In order to do this, a set of tools is required that are integrated into the platform and known as a "software forge".

Forges can be used as soon as software is created within a private workspace. When the software is ready to be publicly distributed, there are two solutions. Either the project is made public, (and others can view its development history) or a new public project is created (with a blank development history, as the old history cannot be accessed).

2.3. Enabling and encouraging collaborative work

Forges aim at enhancing collaboration around software, particularly by collecting the dysfunctions and use cases not covered by the software, but also to contribute to codes. Anyone can contribute to codes by sending a patch, a file containing changes to correct or improve the project, or by making a pull request (PR) or a merge request (MR), depending on the platform. It is also a way of saving a copy of the software on shared infrastructure, which can be archived by Software Heritage when the software has a public source code.

Forges enable software developers to carefully manage these collaborations by choosing which collaborators are authorized to make contributions through a rights management system.

Examples of highly successful open-source research software underline the view that making it as easy as possible to make contributions is a major factor in helping communities of contributors to emerge. These communities can then relieve the initial software creators of a range of maintenance and development tasks, and help the software to gain momentum. Barriers like referrals and moderation around account creation can significantly (and unnoticeably) prevent new contributors from getting involved.

(Raymond, 2001) and (Bangerth et Heister, 2013), for example, set out general guidelines for developing open-source software. They particularly stress the importance of collaboration in this process.

2.4. Building software and analyzing its source code

A common feature in these forges is continuous integration (CI), which enables developers to automatically build the software from its sources. In a way, it guarantees that the forge contains all of the information needed to build the software. Current technologies mean that a number of platforms (multiple operating systems and processor architectures) can be targeted at the same time.

For developers, continuous integration also makes it possible to test how well the software is functioning while it is being developed, prevent regressions (loss of functioning and malfunctioning in existing features) and provide metrics on the "quality" of the code developed.

Solutions for open-source forges that can be self-hosted are classified by the level of features offered.

Table 2.1.: Typology of forges

Level	Features	Examples
0	hosting of the source code versioned with Git (no issues system or CI)	cgit
1	Level 0 + issues system and code review	Redmine, Gerrit, Trac
2	Level 1 + *pull/merge request* system	Gitea/Forgejo, Gogs
3	Level 2 + CI/CD, even other modules (GitLab pages, SourceHut pages)	GitLab, SourceHut, Tuleap

2.5. Looking beyond the source code

Although forges were originally designed to manage the life cycle of software, they are now used more broadly:

For publishing and maintaining websites Using tools such as GitHub/GitLab/-SourceHut pages;

For drafting scientific papers By using document composition tools, such as LaTeX, or possibly using a semi-structured language such as markdown, asciidoc or reStructuredText, which are easier to read as a source;

Sharing data and models Closely managing contributions and publishing content online helps to create communities around data and model sharing (see the HTR United community, for example).

All of the comments on the software are also valuable for these uses.

2.6. Evolution of the target audience for forges

Twenty years ago, forges were platforms where software users and developers interacted. Taking the open-source version of the sourceforge.net forge (such as the GForge forge, in particular) as a basis, they featured two important features for users - downloads of the different versions of the software and forums or mailing lists for requesting assistance and interacting with the development team. Managing the software's source code was just one of many features. There was a clear separation between the members of the development team, who had editing rights over the source code, and the other users.

Software and development approaches have evolved. Some software are provided as services, which means they no longer need to be downloaded. Libraries are distributed through package managers (such as maven, pip and npm), which work in tandem with dependency management tools. Mutual assistance sites like StackOverflow have become alternative sources of information for official software forums.

The arrival of GitHub in 2007 meant the profile of forges rose, and source codes took centre stage, as they became the first thing that people wanted to see when starting a project. All discussions were context-specific, based on issues raised and contributions made. Source codes provide scope to create a project fork, so they have become a common good which can be modified and distributed by everyone, thereby encouraging people to make contributions. At the same time, we have seen the emergence of online development environments, with continuous integration and deployment be-

coming mainstream. The people using forges are becoming more specialist, as people already need to be well versed in developing in order to use them, while forums that are used to ask questions which can be answered by the community are disappearing.

With this in mind, it should be noted that, from late 2020, GitHub started to allow discussions between users without opening issues via the "discussion" function.

Observation 1

Forges have become social networks for developers which are built around common good: source codes.

2.7. Open-source research software

Open-source research software is often created from a proof of concept. It was not initially designed to be distributed outside the academic world and so is not designed do that. One of the key pillars of open science is about raising the profile of this software by helping laboratories to integrate good practice that enable them to raise the profile of their software with little effort.

Observation 2

Open-source research software is created in laboratories which may have a number of supervisory bodies. Therefore, it must be possible for there to be interactions between the members of these different supervisory bodies, meaning that, ideally, individuals from these different supervisory bodies should be able to access the same tool.

Software is sometimes developed as part of collaborative projects which may include businesses. It is important for these types of projects to also be accommodated. Hence the tool in question must not be restricted to the academic world.

More broadly, for some projects, it is also important to be able to interact with users from outside higher education and research (such as gathering feedback via issues and contributing code and documents). This plays a decisive role in whether a user and contributor community can grow and develop around this software.

The modularity and construction of software by integrating components

(i.e. other software modules) made available thanks to the open science environment and unrestricted distribution, are also factors that can help to foster both disciplinary and interdisciplinary collaborations.

Open-source research software can also act as research topics for software-engineering researchers. Making them accessible on a clearly identified forge would make it easier to reuse them for this purpose. In addition, by bringing state-of-the-art software engineering onto these academic forges (such as code static analysis, tests, quality assurance and documents), the entire community can benefit from the latest advances in this area.

The easy "reproducibility" offered by continuous integration (via images of managed execution environments) is also a very important aspect of open science.

Promoting the use of standard tools would make training and acculturation with laboratories easier. Automating tasks using computer tools has become a key skill in all academic disciplines (Wilson, Aruliah, Brown, Chue Hong, Davis, Guy, Haddock, Huff, Mitchell, Plumbley, Waugh, White, et Wilson, 2014). Software forges need to become tools like any other found in the toolbox of researchers and engineers.

2.8. Target audiences for forges within Higher Education and Research

Forges may be used at different times in a research project.

When a project is being launched, they can be used for developing a proof of concept. This work can be done by students (as part of a research internship), doctoral or post-doctoral candidates, engineers and/or researchers. In this instance, a private project will be created where the VCS will be an easy way of finding out the contribution made on the software produced. This practice should be encouraged.

At some point during any project, there is the issue of sharing what has been produced, including the software. When these project outputs are shared in the form of open-source software, a number of strategies may be put into action.

A pared-back strategy would involve distributing the software as it is, without overly promoting interaction with other research groups or general society. The software's executable code can be distributed easily by making it available on a webpage as an archive file (.tar or .zip), for example. However, distributing it using this approach will have less of an impact, as it will make it difficult to make contributions.

A more proactive approach would involve distributing the software on a forge, so that other users can provide feedback (via an issues system, in particular). Therefore, it would be better to put in the work beforehand to document how the software operates and is constructed, in order to avoid frequent requests about these topics. This approach would ensure that the maximum value can be obtained from the software. More and more research projects are incorporating this aspect during the development phase of projects.

One successful strategy is to open up the development of the software itself in order to make it easier for other developers to make contributions (through MRs or PRs). So it is important to document the internal functioning of the software in order to make it easier to dive into the code, for instance by drafting a "maintenance manual". It is also worth documenting the contribution process, such as the code style expected, the MR/PR submission and code review process, how CI is functioning and the agreement around copyright assignment. Forges generally make this process easier in order to encourage contributions. This could be decisive in whether a software project takes off. Its maintenance and development could then be taken on by an international community (enjoying significant working capacity, as a result), rather than the single research team behind the software (with limited resources).

The forges used for developing the project and for getting the maximum value out of the software may be different, in order to meet the respective needs to these two aspects.

Observation 3

There are currently a lot of software created as a result of French research which have been developed over a number of years due to an active community, and these have been hosted on commercial forges.

3 | The outlook for Higher Education and Research forges

In this chapter, we examine the kinds of forges used in France in higher education as part of research. Although this is not a comprehensive picture, we have been able to note some trends. An analysis of the reasons why the current situation exists is also put forward.

3.1. Major use of commercial forges

The majority of open-source software created by Higher Education and Research and looking to interact with wider society use a commercial hosting solution open to everyone such as `github.com`, `gitlab.com` or `sourceforge.net` (Escamilla, Klein, Cooper, Rampin, Weigle, et Nelson, 2022). This becomes clear in France simply by taking a glance at the repositories where software that has won awards during the first french open science software award ceremony are hosted:

- 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

On 11 February 2023, the code.gouv.fr platform listed 9818 depositories linked to Higher Education and Research. 29% of these depositories (2627) are hosted on GitHub. Only 30 depositories are hosted on gitlab.com. The other commercial solutions have not yet been included.

The Software Heritage enables us to take a view of the number of public projects across the world that are found on commercial forges: 156 million for GitHub, 4 million for GitLab and 2 million for Bitbucket, which were the most frequently used in early February 2023. It should be noted that GitHub contains far more public projects than the other forges.

The main benefit of turning to these solutions for an open-source Higher Education and Research project is making code or documentation contributions easier by adopting a "developer social network" model. In fact, for many developers who have an account on these platforms, contributing to a public project on these forges is very straightforward. This is why community structures such as Eclipse have decided to introduce these commercial forges, while also maintaining their own forges.

However, commercial forges are not sustainable solutions for hosting research software. For instance, in August 2022, GitLab made plans to automatically archive projects that have been inactive for a year (Sharwood, 2022), although some mature software may have a slower development cycle and extended periods of inactivity, without having their usability called into question. In the past, commercial forges like Google Code (Staff, 2015) and even community services like Gitorious (Degeler, 2015) have been closed, and hundreds of thousands of projects were deleted from BitBucket following its decisions to stop hosting projects using the Mercurial version control system (Chan, 2020). These closures have led to the disappearance of modification histories and forum exchanges, which are important assets.

Therefore the terms and conditions of use for commercial forges may cause problems here.

Finally, commercial forges are not necessarily able to meet the needs of the entire project life cycle. Commercial forges for private projects must be used in compliance with European Union data protection regulations which call for servers to be hosted in European Union territory and not to be subject to non-European-Union legislation (such as CLOUD Act and FISA 702). More information about this is available on the CNIL web site.

Observation 4

In order to gain access to these platforms, the terms and conditions of use for commercial forges must be accepted by Higher Education and Research staff on an individual basis. If a member of staff does not accept these conditions, it may cause problems.

3.2. Using open-source community forges

There are many open-source community forges such as the Free Software Foundation (FSF) and its GNU project, Apache, Eclipse, OW2 and many others that, in addition to their own projects, host open-source projects from the research world. They enable projects to be launched by giving them

support beyond just getting them started. When they are stakeholders for these projects, they can ensure the long term viability of software production. These communities may also position themselves individually towards open science; see OW2's position, for example.

However, these forges host mature professional projects which are intended for wide usage. These communities determine which projects they would like to host and support, based on their own criteria. It may be the subject area for the project, its level of maturity, its development or governance processes or its positioning in the financial world or society.

Observation 5

This makes open-source community forges a good solution for academic projects which have gained somewhat of a profile or with a clearly defined strategy for getting the maximum value out of these projects. They are not a good solution for developing proofs of concept in laboratories.

> However, the OW2 forge can be used as a forge for experimentation through individual community users, who can have two projects in their own name. These individual projects do not go through the standard process for submitting a community project.

These are some examples of software created from public research hosted by the OW2 and Eclipse foundations, as well as by the GNU Project:

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

3.3. The SourceSup national forge

The SourceSup forge (https://sourcesup.renater.fr) went into operation in 2004 operated by RENATER. At the time, it was a GForge platform. This service switched to using FusionForge in 2009, when the GForge ceased to be an open-source platform. However, the FusionForge solution has not been developed for a number of years (the last major release was in 2018). Additional functional bricks were added in 2015: MantisBT for managing

issues, Testlink for managing testing campaigns, Sonarqube for quality assurance, Nexus for managing artefacts, Jenkins for continuous integration and Nuxeo for documentation management. The infrastructure is virtualized, based on a principle of separating resource-intensive applications. One individual oversees "level 2" maintenance and support on software bricks. "Level 1" support is shared with other RENATER applications.

In December 2022, the SourceSup forge contained 762 public projects out of more than 5,200 projects in total, and boasted more than 13,000 users.

Here are some examples of projects hosted on SourceSup:

AGATTE : Leave management software package used by many Higher Education and Research institutions

WIMS : Interactive teaching software

OTAWA : C++ framework for determining the worst-case programme run time.

The forge accepts public or private projects. Any Higher Education and Research staff member can ask for a project to be created, although student projects are not accepted.

Observation 6

RENATER's management committee decides how SourceSup's technical and operational capabilities will develop and move forward. There are no major SourceSup developments currently in the pipeline.

The SourceSup solution is currently quite different from the other forges in Higher Education and Research. First, it has maintained its user-oriented focus with its distribution lists and its forums (like SourceForge). It also continues to support Subversion. Many projects have started with this centralized version manager, which was the most used system before Git appeared. While SourceSup can use Git as a version manager, like other forges, it also offers Subversion as an option.

SourceSup has chosen to use tools coming from different sources to provide all of the services required for software development. This approach sharply contrasts with the way in which commercial forges, such as GitHub and GitLab.com, have evolved, as well as with self-hosted forges which now offer the majority of these tools in a single solution to make it easier for users to adopt them and for institutions to maintain them.

The choice of distinct tools enables there to be a certain level of autonomy by closely managing the software used for each need. However, it is also a barrier to the forge being adopted by new users who are unable to find the tools that they are now getting accustomed to using.

Educational National Forge

Recently, the French Ministry of National Education and Youth announced the creation of a new national forge known as the "forge des communs numériques éducatifs" (forge of shared educational digital technologies):

Furthermore, communities of teachers (and other individuals involved in education) can also be platforms for building new tools. Teachers, particularly those in the fields of digital technology and computer sciences (NSI) and digital sciences and technology (SNT), are waiting for a "forge" which would help them to collaborate with their peers and to share computer code. The Ministry is now responding to this need by providing a technologically autonomous forge that is shared at a national level.

Source : Digital strategy for education 2023-2027 (MENJ - French Ministry of National Education and Youth), page 25 (MENJ, 2023)

3.4. 40 self-hosted public Higher Education and Research forges

In 2023, there were at least 40 self-hosted public forges in Higher Education and Research Institutions, not including the forges used on a strictly internal basis by the institutions.

The list of these forges, available in appendix A, is non-exhaustive. It has been put together based on, on one hand, reports submitted to the `code.gouv.fr` platform and, on the other, from the knowledge network of members of the working group behind this report who were sent the questionnaire in appendix C. It must be treated as a preliminary survey which, although incomplete, is already providing interesting information for this report. Although it may prove useful to complete it, it is worth bearing in mind the underlying difficulties associated with listing every single one. It is inherently difficult to list every forge system because they may be operating on different levels, for example, in research groups, laboratories and institutions. Some of these platforms are for internal use only by a Higher Education and Research institution (such as the University of Artois' platform, for example) and are therefore not listed here.

When reading this list, it is worth noting that:

- major national research bodies have their own forges or are about to get them;

- universities and higher education institutions offer their own forges, while possibly separating teaching and research (as is the case with CentraleSupélec, for example) or providing even more refined granularity (Lille University offers its own GitLab platform, with a forge for the Research Centre in Computer Science, Signal and Automatic Control (CRIStAL) laboratory and another for the Computer Science Department of the Faculty of Sciences and Technologies).

Observation 7

Many self-hosted forges are GitLab platforms. Even though GitLab is a piece of open-source software which is open to contributions from all individuals, the resulting dependence on GitLab Inc. needs to be questioned as, in practice, it drives its development.

3.5. Analysis of the situation

3.5.1. Chronological history of the forges referred to in this document

As an initial point, and in order to help you understand the current situation, it is important to be aware of some key dates for the software forges that we are looking at:

- 1999: SourceForge, the first open-source software forge, is launched
- 2002: GForge, the open-source alternative to SourceForge for self-hosting, is launched
- 2004: Git (open-source version management software) is launched
- 2007: GitHub (a Git-based commercial forge) is launched
- 2011: GitLab (a functionally similar piece of open-source software to GitHub for self-hosting of private projects) is launched
- 2012: GitLab.com (a GitLab-based commercial forge for private projects) is launched
- 2014: GitLab.com self-hosts its own development, thereby becoming an alternative to GitHub

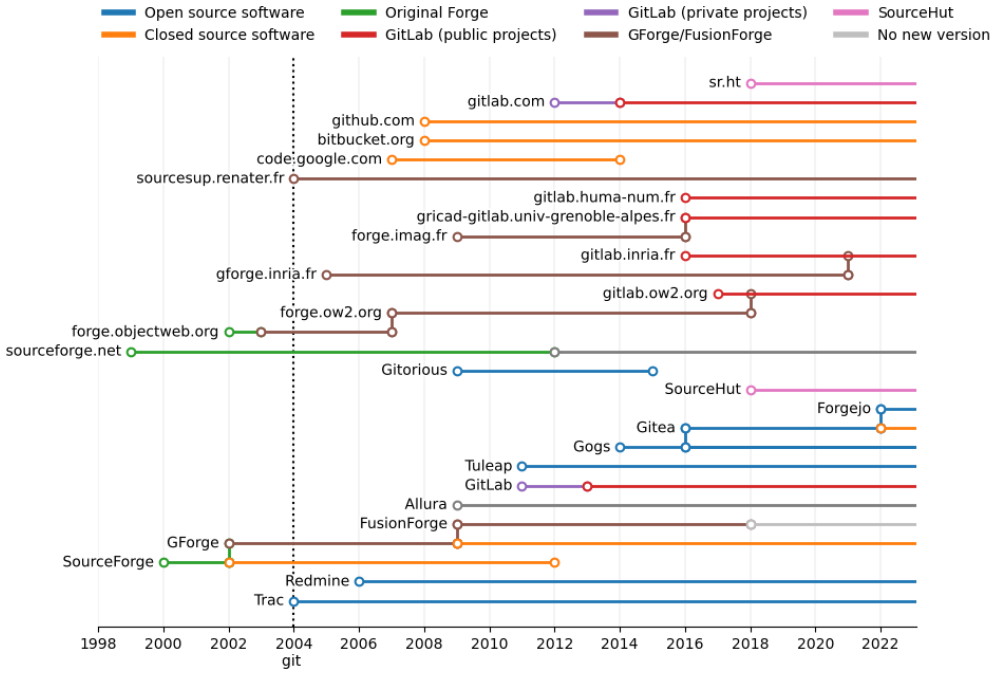

Figure 3.1.: Founding dates for different forges

When an open-source version of SourceForge emerged, this lead to the creation of the first self-hosted forges. When the licenses became non open-source, forks from the community emerged (SourceForge/GForge/-FusionForge, for example). There are regular technological changes. The `sourceforge.net` forge now uses the open-source software Apache Allura.

One noteworthy occurrence at the end of the 2010s was the shift from GForge/FusionForge to GitLab for self-hosting within a number of structures. The initial versions of GitLab could only be used for hosting private projects. GitLab first introduced public projects in octobre 2013 (GitLab 6.2) and became a fully-fledged alternative to GForge/FusionForge. In January 2014, collaborative development on GitLab is made using GitLab itself.

3.5.2. Why are there so many self-hosted forges?

This is a hard question to answer as there are undoubtedly many reasons for this situation. However, some points can be put forward to explain why this is happening.

First of all, it should be noted that almost all of the forges listed are GitLab platforms. GitLab is a piece of software very easy to install and maintain, and is not computer-resource-intensive. Therefore, an institution can deploy it at a lower financial cost and using fewer human resources. In particular, in universities and engineering higher education institutions that specialize in

computing, these forges are needed to help students learn how to use them in an educational setting. There may be a distinction between teaching and research forges in some institutions, while this is not true in others.

Forges are often also deployed in computer laboratories, as these laboratories have both the technical capabilities to put in place these solutions and an awareness of the limits of commercial solutions.

The main benefit of using self-hosted forges is managing where each server is located, thereby ensuring compliance with European Union personal data protection legislation (GDPR) and that sovereignty issues are taken into account (for private projects and for partnerships with industry, for example), where necessary.

Another benefit of self-hosting is being able to control access to this service, its uses and the distribution rules for software. This enables the institution to implement its own policy covering research software.

Finally, from a practical point of view, having its own forge would enable an institution to centralise software development or to showcase all of its software by automatically installing a mirror of the projects hosted on other forges. This way, the institution could have an overall view of the software that it has produced.

It may be questioned why so many higher education institutions have set up their own forge when SourceSup, a national forge, exists. A number of arguments can be put forward, including its heterogeneous technological development, the lack of user voices on the oversight committee and even the lack of a two-way communications strategy (from SourceSup to users and from stakeholders to SourceSup). This strategy can be compared with the history of other forges such as the French National Institute for Research in Digital Science (Inria) in Higher Education and Research, or OW2 and Eclipse in open-source communities. At Inria, after a new GitLab-based forge was set up, the GForge-based Inria forge was shut down in December 2020 following an "overlapping" period to ensure the migration of projects. Since January 2023, Inria has had two GitLab platforms: one for public collaborative projects and another for its internal projects. It is the same story for OW2. Its long-standing GForge-based forge was replaced by a GitLab platform in 2018. More recently, Eclipse has decided to migrate its proven technical platform combining Git, Gerrit and BugZilla to GitLab. Denis Roy, from the Eclipse Foundation (Roy, 2022), justified why this migration had been done:

> *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.*

To sum up, in just a few years development practices have significantly evolved with a rapid transition from user-oriented forges to developer-oriented forges, thus making source code the central issue, and these have made a massive contribution to mainstreaming the adoption of a contribution approach built around forks and pull or merge requests.

Observation 8

It is a major challenge for national forges to actively monitor changes in use and technical solutions and to undertake the required transformations for offering all users a familiar environment that is compatible with the tools they use when they collaborate on other software projects.

This is a difficult task because, as has been said, software development is an activity which extends far beyond Higher Education and Research, with development at a rapid pace and technologies evolving quickly.

Failing the adoption of an ambitious strategy around this, and in view of how easy it has been to install recent self-hosted forges, we have ended up with the current fragmented situation with multiple forges scattered across the different institutions.

3.5.3. Difficulties in interacting with society

The main limitation to the forges currently available in Higher Education and Research is the limited audience for these forges (i.e. the individuals who can create an account), as the majority of platforms for the forges available do not allow individuals from outside of Higher Education and Research to create an account on these platforms themselves. This therefore poses a barrier to interacting with society.

While some of these forges allow the creation of external accounts, they are often difficult to access (for example, for `gitlab.inria.fr`, an external account must be "referred" by a member of an Inria project team) and limited (the GitLab external account cannot create its own projects). This often makes it impossible, or very difficult at the very least, to suggest changes

with this type of account, because this would require a fork on the original project, and the external account would therefore need to create its own project on the forge. Users need to have an account beforehand in order to report a bug, which may be prohibitive.

The OW2 approach is a pragmatic one; it allows each person to create an account on its platform, but limits the number of new projects created to two as a default setting. This allows users of OW2 software to easily interact with the issues system and be able to offer code contributions. Forge managers have found that this default open position has not led to phantom accounts being created. However, OW2 delegates account creation to its directory, and not directly to its forge.

> RENATER supports creating a "compte réseau universel" (CRU) that enables individuals outside Higher Education and Research to access these services by accommodating this virtual identity provider. Some forges allow authentication using an UNA account.

Observation 9

As a general rule, forge software like GitLab does not limit project creation to simple contributions (i.e. forks on platform projects). There is also no way of preventing a user from uploading photos or videos to their project space.
However, allowing individuals outside of the institution to create projects may result in legal problems, such as compliance with the French Act of 24 June 2020 on Hate Speech on the Internet.

One approach that is occasionally implemented is EduGAIN / Shibboleth's distributed authentication, which automates the creation and prevents multiple authentication steps. However, the institution managing the authentication must provide the information required for creating the account (user login and email address). Some institutions use a white list of authorized third parties, and their users must explicitly request to be added to this list in order to be able to use this authentication system. Without it, a fairly cryptic message will be returned ("Empty uid", "Email can't be blank", "Email is invalid", or even "500 Whoops, something went wrong on our end"), completely baffling users. It would be better for the institution to use a consent page, enabling users to confirm the request immediately rather than having to guess that they have to ask their local DSI to add them to the white list.

> In February 2023, Shibboleth support was removed from GitLab 15.9 because the Ruby component providing this feature was no longer maintained. In practice, this had prevented upgrades to many Higher Edu-

> cation and Research forges, unless the application code was modified. An appeal for help was issued to the community to upgrade this component. One workaround was assigning Shibboleth's management to a single sign on authentication system. The support for Shibboleth was restored in June 2023, with GitLab 16.1, thanks to the community.

Limiting access to a community also makes managing user accounts more difficult as their status changes (as they come into or leave Higher Education and Research, or as they arrive in or leave the institution or laboratory).

> The GitLab forge supported by HumaNum infrastructure expands this EduGAIN / Shibboleth approach by enabling users, thanks to its HumanID system, to authenticate themselves using their ORCID, HAL, Twitter, LinkedIn or Google accounts, as well as their eduGAIN account.

Spam is another complicated aspect to manage, due to the possibility that users may sign up for the platform without being validated. It may be uploaded via code snippets and comments or issues for public projects.

> In practice, OW2 has stated that it does not have issues with spam on its platform, because accounts are created away from GitLab. For the Eclipse foundation, the situation is more complex, particularly in relation to their Wiki service.

Observation 10

To summarize, on multiple existing Higher Education and Research forges the key issue that needs to be addressed revolves around setting out a coherent access policy which maximises interactions without endangering the infrastructure, both for interaction between staff members and with society.

3.5.4. Support levels and the need for trust

There may be a difference between a forge's level of support expected and the level of support provided, particularly as this support will always be compared with the support provided by commercial forge. While in the majority of cases the support provided meets the users' needs, there may be occasions where technical changes need to be implemented in order to restore features to cope with a constantly changing technical environment.

> For example, on GitLab forges, continuous integration based on Docker images is impacted by changes in Docker's image registry policy (Docker Hub, GitLab's image registry), which limits the number of accesses per

> machine and per day. As a result, every day, after a specific number of requests to the Docker hub through continuous integration, connections may be refused and continuous integrations may fail. In order to circumvent this issue, the maintenance workers for each platform have to set up a local registry which acts as a cache, to reduce the number of accesses to limited-access public registries. However, these solutions are not necessarily always put in place as a matter of course. For users who are not necessarily aware of these technical subtleties, this makes it impossible to work with ease.

Observation 11

More broadly, users need to be able to trust that a platform is robust and there for the long term, one they can count on when they want to broadcast their research work, regardless of the form.

3.5.5. An open-source or a proprietary license?

The solutions available for self-hosting can be distributed under an open-source or a proprietary license, with more features or support generally provided under a proprietary license. One good example of this is the "Ultimate" version of GitLab, which is distributed under a commercial license and which offers additional features compared to the version distributed under an open-source license.

The decision about which type of to choose then arises - would it be better to use the solution with a closed-source in order to enjoy additional features?

Observation 12

Choosing a forge under a commercial license makes it more difficult to open up the forge more widely. In fact, closed-source user licenses generally come with certain restrictions, such as a maximum number of accounts and a limited scope of usage.

However, choosing a paid version of the software may be down to the desire to provide the bodies editing and publishing these tools with the resources to perform their work in order to ensure that the solutions they provide, and therefore the forges using them, are there for the long term. Providing some of their codes under closed-source licenses and the "freemium" economic models implemented are a way of ensuring recurring revenue for these entities and safeguarding jobs for their developers.

The remainder of this chapter will analyse how forges are currently used by highlighting three major difficulties encountered in implementing them. These are: structuring the forge and its life cycle, its scope and managing copyrights and licenses.

4 | Points to consider when it comes to forges

In the course of our study, we have noted a number of points worth considering when setting up a forge. There is no good or bad way to use a forge. It is more about ensuring that the forge operates in a way that achieves its objectives.

4.1. A showcase platform or a simple tool?

There are two orthogonal ways of viewing how these forges function.

The first is to see them as tools provided to the community. In this case, it is about promoting good practice while providing access to quality tools for managing projects. This seems to reflect how the majority of Higher Education and Research forges operate.

The second way, but one that is not exclusive to Higher Education and Research forges, is to view public forges as platforms showcasing the community, and the community is therefore entitled and has the power to decide what is published. This is mainly the case within open-source communities (such as Apache, Eclipse, OW2[1]).

However, on many forges, creating public projects does not require a validation step. As a result, there is a range of public projects on these forges, from "hello word" projects to large applications maintained for years. This heterogeneity is a problem, both in terms of offering a platform for showcasing projects of a specific "quality" level, and in terms of listing the software produced by a community. However, it is not necessarily a problem where the main objective is to encourage the usage of a tool and processes associated with it.

Observation 13

The lack of monitoring (or poor monitoring) of project creation can make managing them a complicated task. It is difficult, or even impossible, for administrators to determine which projects should be retained or not.

[1] However, OW2 allows each user to create two public personal projects.

> In order to manage the content of a forge more carefully, it would seem necessary to establish a process and life cycle for projects, for example by establishing a level of maturity (such as the Market Readiness Level proposed by OW2, for example). At Eclipse or Apache, there is an "incubating project" concept for software that are evolving quickly. Eclipse also has the "train" concept, which includes bricks that are theoretically mature. This forge also features the "mentor" concept, with the role of validating a project's creation.

Furthermore, without distinguishing between the simple tool and showcase platform concepts, it is worth noting that the developments contributed by a community, a research unit or a supervisory body help to ensure that any software creations by these groups can be tracked and will last long term. Reference sites like code.gouv.fr or the Software Heritage universal software archive are built based on the indexing feature offered by forges.

4.2. Project organisation

Observation 14

One of the main difficulties in managing a forge is knowing how to organise projects. One solution may be to create a space devoted to each sub-structure (research project, team or laboratory), which would be responsible for creating its projects in this space. However, this solution leads to two other problems: where should projects that are shared by more than one sub-structure be placed? How do you manage sub-structures which do not have the expertise or internal resources to perform this task?

> For example, at an IRD (French Research Institute for Development) level, a *unité mixte de recherche* (a joint research laboratory) is primarily responsible for managing projects, and then declares sub-projects. Where there is a justified request, a root project may be declared for long-term projects between different *unité mixte de recherche* laboratories. As a default, the projects can be accessed by logged-in users. The IRD is moving towards a principle of providing access to signed-in individuals as soon as projects are launched, in order to encourage collaborations as early as possible. This policy is still under consideration, but will likely include the above aspects.

4.3. Copyright management

If the software developers all have the same employer or all belong to the same structure containing multiple supervisory bodies, there are no particular problems around copyright management.[2]

Conversely, as soon as code from an external developer is incorporated, it is important to manage copyrights. This is generally done in two ways, either with a lighter touch through Developer Certificate of Origin (DCO) or in a more structured way through Contributor License Agreement (CLA). Irrespective of which solution is chosen, there is still the issue of when to do it. The situation must be clear as soon as the code is contributed, because this information plays a role in determining whether or not to integrate the code into the software.

Asking to include the contribution in the forge may discourage individuals who simply want to flag up a defect. In addition, regular checks must be made to ensure that the chosen document is always correct (in particular, because the contributor's employer may change). At Eclipse, contributors must regularly provide their approval to their Eclipse CLA.

Asking them to do so before even making an initial contribution, for example, in order to be able to create and MR/PR, may strike fear into a developer who is not used to this process.

Asking them to do so before integrating the code into the project is standard practice; Developers know when their code is ready to be integrated and only lack the "administrative" rights to complete this process. When the contribution is extremely small this process can be bypassed, and the correction can be made by the main developers by mentioning the contributor in the commit text.

4.4. Managing a project across a number of forges

Self-hosted (Higher Education and Research) forges can be synchronized with commercial forges (gitlab.com or github.com). However, doing so would mean that information is no longer centralized. This means that it is a good idea to deactivate the public issues system in order to store the issues on the local forge, if the local forge allows the free registration of external contributors (who can create issues but not offer code contributions) to freely submit issues. However, this solution is not practical for contributors

[2] It is assumed here that the copyrights from different contributors are transferred to the employer, and that the employees are authorized to publish the code that they produce in open source.

who need to use two accounts, i.e. one for alerts (issues, bug reports and suggestions) and the other for contributions (code and documentation). It is even more difficult to adhere to this discipline when there is a major turnover in contributors to research projects.

> The Eclipse foundation uses a number of forges (one self-hosted forge and other commercial forges). Even when a mirror is put in place between the forges, each project is managed on a single forge in order to avoid the issues mentioned.

In order to deal with this limitation, it is worth noting the current initiatives being undertaken to target the forges federation. One example is the work undertaken as part of the Forgejo open-source project, a community alternative to GitLab and GitHub, aiming to specialize the ActivityPub protocol created through the more generic work around the Fediverse. The ForgeFriends initiative is also working along these lines.

We cannot conclude this chapter without focusing on a growing need, which will become increasingly apparent and significant, that of continuous integration.

4.5. More and more continuous integration services

The management of software projects is not the only functionality expected of a forge. Providing up-to-date documents is key, and having the means to publish websites from a forge is an added bonus (for example, the *GitHub/GitLab/SourceHut Pages* service). In addition, many repository-content-analysis-based tools may prove to be important for maintaining the software:

- analyzing the legal compatibility of software licenses and of software components;
- detecting components with known vulnerabilities;
- detecting vulnerabilities in the code produced;
- detecting bad development practices in the project;
- etc.

Most of these features are based on the option of being able to implement continuous integration, i.e. programme execution being triggered by a specific event or a certain conditions, such as every code update, for example.

> The need for energy efficiency in digital technologies means that this continuous integration must be configured in such a way as to prevent unnecessary triggers. Rapid testing is generally performed for each

> change, and batch processing may be performed at different intervals based on the different tools used - each evening, each week or even each month. The tests launched may also be filtered, based on the files actually edited.

It should be noted that the machines used for managing shared continuous integration on a Higher Education and Research GitLab platform are generally more powerful than the machines used by the GitLab platform itself. In fact, if a code modification is going to need to use the GitLab server for a number of seconds as a maximum, a number of minutes or even a number of hours may be required to run its continuous integration. Therefore, this feature cannot be scaled up smoothly without a suitable architecture that helps to tailor the resources available to meet the demand, and therefore requires significant amounts of resources.

In addition, implementing continuous integration, which involves automatically building software and running it in production in a specific environment, requires more sophisticated architectures, ones that provide secure access to these various dynamic environments.

The currently recommended solution is based on using containers which can be used for deploying continuous integration resources upon requests, and making it easier to scale them up.

Additional technical issues still need to be addressed, such as signing off software so that it can be installed on recent operating systems without having to change its security level. Combining efforts in this area would make it easier to distribute software created through research to wider society.

Observation 15

Being able to maintain this architecture in the long term requires specific resources and skills, so it makes sense to combine efforts.

Providing runners for continuous integration raises security issues and could lead to abuse. Commercial forges have scaled back or even deactivated accounts which use CPU time on runners free of charge.

5 | An overview of the solutions

The following SWOT matrices summarize the strengths and weaknesses of the different types of forges that are currently available for distributing open-source software and any other artefacts created through research, from the perspective of users of these forges.

5.1. Commercial external forges

We are looking at forges like github.com, gitlab.com, bitbucket.org and sr.ht here.

Strengths

- "All in one" feature integration
- Free core feature set
- Acceptable availability for the majority of projects
- For GitHub: a now familiar user experience
- Pour sr.ht: very responsible human support
- An international profile
- An international community

Weaknesses

- Dependence on the company's commercial policy
- No checks of the resources allocated to features
- No involvement in strategic decisions
- Non-sovereignty
- No long-term guarantees

Opportunities

- Comprehensive training resources for these tools

Threats

- Dependence on the regulations of the country where these companies have their registered offices

5.2. Community external forges

We are considering forges from foundations such as Apache, Eclipse, OW2 and FSF, here.

Strengths

- Features
- Availability
- Support
- Longevity
- Profile at a community level
- The community itself

Weaknesses

- Project must be accepted by the community
- Not all areas covered by the open-source-software communities

Opportunities

- Project promoted in an ecosystem
- Methodology support
- Community confidence in its own project

Threats

- Dependence on the regulations of the country where these foundations/associations have their registered offices

5.3. Local self-hosted forges

Strengths

- Customized features
- Customized availability
- Sovereignty
- Resilience
- Support (depending on the platforms)
- Longevity (depending on the platforms)
- (Proximity)
- Profile at an institutional level
- Institutional community

Weaknesses

- Availabilities (depending on the platforms)
- Support (depending on the platforms)
- Multiple solutions available
- Difficult to access outside of the institution that owns the project

Opportunities

- De facto, institutional software catalogue
- Institution research-software development policy implemented
- Expertise maintained at an institutional level
- Sharing of good practice

Threats

- Evolution of the solution selected (scope of the features under open-source vs versions which are now paid)
- Difficult to find the expertise required to maintain the solution at an institutional level

5.4. National self-hosted forges

Strengths

- Features
- Availability
- Support
- Sovereignty (public projects)
- Longevity
- Sustainability
- National profile
- National community

Weaknesses

- Sovereignty (private projects)
- Opened up beyond the national level

Opportunities

- De facto, national software catalogue
- National recommendations for research software development implemented
- Combined maintenance efforts
- Supported by a comprehensive public policy

Threats

- Evolution of the solution selected (scope of the features under open-source vs versions which are now paid)

6 | Conclusion

Software development practices have developed to an enormous extent with the increasingly widespread use of tools. These have helped to greatly *streamline collaboration around software projects*. This has progressively led to forges being transformed into *developer social networks*, and the ease in which collaborator contributions, even if done infrequently, can be integrated has become *a key factor in helping open-source software to take flight* and in creating communities around them. Beyond software, forges are also used for drafting articles and the community management of data or models.

With this in mind, the world of research needs a forge which is at least open to the entire world of Higher Education and Research, and ideally, open to society as a whole.

Even though France has a national forge for Higher Education and Research, SourceSup, its functional development has moved away from the practices of many Higher Education and Research developers, and no major developments are currently planned. Therefore, in its current form, it does not meet the needs expressed.

The availability of solutions under open-source licenses for installing forges that include the expected features, and how easy it is to install and maintain them, have meant that currently there are multiple forges maintained by individuals, teams, laboratories and institutions within the Higher Education and Research world. On these forges, one of the key issues that needs to be addressed, both for interaction between Higher Education and Research staff and with wider society, is the setting out of a coherent access policy which maximises interactions without endangering the infrastructure, both for interaction between staff members and interaction with society.

In order to solve these problems, many pieces of open-source software created through French research have had to be migrated elsewhere. For academic projects which have gained a higher profile or that have a clearly defined strategy for getting the maximum value of these projects, an alternative is available via forges maintained by open-source-software communities, but this is not a workable route for software which is at the proof of concept stage in laboratories.

Many projects are therefore on commercial forges, which allow unlimited project creation, but in order to use them, Higher Education and Research staff must accept their terms and conditions of use individually, which could

cause a problem, especially as there are no guarantees that these commercial forges will be long-lasting.

All of these issues must be taken into account in a comprehensive way.

On one hand, the Higher Education and Research staff who are developing research software are expressing what they need:

- a forge that they can trust: a robust, long-term platform that they can count on when they want to distribute their research work, in any form whatsoever;
- a forge which evolves with state of the art technology: it must contain tools of a comparable level to those provided by developers on other forges;
- a forge which helps to build and grow the contributor community with ease: registering and creating projects must be easy, and it needs to be user-friendly and ergonomic;
- simplified management of intellectual property processes.

On the other hand, there are challenges facing whoever has to provide these services:

- The lack of checking (or low checking) of account and project creation on a forge is important for creating communities, but it can make managing them a complicated task, and carries technical and legal risks for the infrastructure. It becomes difficult, or even impossible, for administrators to determine which projects should be retained or not, and distinguishing between normal use and misuse may demand substantial resources;
- Keeping up to date with technological developments and changes in use requires monitoring and the operating capacity to roll out new solutions, including those made available on a trial basis, all while keeping the old ones in production, also as part of "overlaps";
- Providing a service to a very large community means that thought has to be given to how projects are organized in this forge, particularly for projects involving multiple supervisory bodies, and the practical implementation of this kind of policy in laboratories with varying skill levels;
- Encouraging collaborative work between Higher Education and Research institutions could make a shared rights-management process or system to be set up within an identity federation necessary.

Finally, there are major strategic challenges in this area. There are no guarantees of the long-term existence of commercial software forges, and courts outside Europe have jurisdiction over the most popular ones. Each code base uploaded to a commercial forge provides a very early insight into the research and industrial activities of the members of staff in the countries

where they are uploaded (as part of private projects), enriches the data that the platform relies upon to teach its AI code writing tool (such as OpenAI Codex/GitHub copilot and GitLab suggestions), and therefore helps to maintain GAFAM's domination in digital technology. This means that managing the way in which code created in our laboratories is used is a major sovereignty issue.

A considered and coordinated response to all of the needs and challenges expressed is needed as a matter of urgency. The roll-out of a self-hosted forge is relatively easy, but ensuring that its infrastructure and features remain state of the art over the long term, putting in place access and project structuring policies, tracking uses and offering support and preventing misuse requires particular resources and skills which must be shared.

A | List of self-hosted public Higher Education and Research forges

A.1. Institutions

Forge	Log-In	Noe-ESR Involvement	Continuous Integration	Other Services
CEA (French Alternative Energies and Atomic Energy Commission) codev-tuleap.cea.fr	LDAP CEA			
CentralSupelec gitlab-research.centralesupelec.fr	LDAP Centrale SupElec (+ guests)			
CIRAD (French Agricultural Research Centre for International Development) gitlab.cirad.fr	Renater	No	GitLab CI	GitLab Pages
CNRS src.koda.cnrs.fr	CNRS (Janus)	No	Yes, but no shared runner	No
gitlab.in2p3.fr	EduGAIN	external users	GitLab CI	GitLab Pages

Forge	Log-In	Noe-ESR Involvement	Continuous Integra-tion	Other Services
plmlab.math.cnrs.fr	Renater	Invitation	CI/CD with shared runners	Pages with personalized domains, artefact repository and Docker image registry
gitlab.mbb.cnrs.fr	LDAP MBB + invitations	No	Yes, but no shared runner	Container image registry
forge.ins2i.fr	Self-registration, CNRS (Janus) in progress	Under consideration	Yes, but no shared runner	Artefact repository, Container image registry
gitlab.huma-num.fr	EduGAIN,ORCID	LinkedIn, Twitter and Google	GitLab CI	GitLab pages
IMT gitlabev.imtbs-tsp.eu	Internal Directory + Shibboleth IMT	No	CI	
INRA (French National Institute of Agricultural Research) forgemia.inra.fr	Renater	CRU	GitLab CI/CD (4 shared runners)	GitLab Pages, Artefact Repository, Docker image registry, Mattermost
INRIA gitlab.inria.fr	Inria	referred external guest(s)	GitLab CI	GitLab Pages, Docker image registry
IRD forge.ird.fr	Renater	CRU - others in progress	In progress	GitLab pages

Forge	Log-In	Noe-ESR Involvement	Continuous Integra-tion	Other Services
IRSTEA (French National research institute of science and technology for environment and agriculture) gitlab.irstea.fr	LDAP users	external	Gitlab CI/CD	
TelecomParis gitlab.telecom-paris.fr				
Bordeaux U. gitub.u-bordeaux.fr	Bordeaux University	After validation, limited	No	
gitlab.emi.u-bordeaux.fr	Bordeaux University, students	No	Yes	CI, GitLab pages
Caen U. git.unicaen.fr	Renater		GitLab CI	
Gustave Eiffel U. gitlab.univ-eiffel.fr	Gustave Eiffel email address	No	CI/CD	Pages
Grenoble U. gricad-gitlab.univ-grenoble-alpes.fr	UGA	Yes: upon registration (without validation) but with limited rights.	GitLab-CI (with shared runners)	gitlab pages (both private and public pages), Container registry
La Rochelle U. gitlab.univ-lr.fr	Directory	Invitation	No	
Lille U. gitlab.univ-lille.fr	RENATER			
Limoge U. git.unilim.fr	Internal directory	No	CI	

Forge	Log-In	Noe-ESR Involvement	Continuous Integra-tion	Other Services
Littoral U. gogs.univ-littoral.fr	LDAP ULCO	Guests	No	
U. Lyon1 forge.univ-lyon1.fr	CAS univ Lyon 1			
U. Montpellier 2 gitlab.mbb.univ-montp2.fr	LDAP + guests			
Nantes U. gitlab.univ-nantes.fr	Internal Directory	Invitation	CI/CD	Pages, artefact repository, Docker image registry
Paris Saclay U. gitlab.dsi.universite-paris-saclay.fr	Internal Directory	No	GitLab CI	GitLab pages, artefact repository
Pau U. git.univ-pau.fr	University LDAP			
Strasbourg U. gitlab.unistra.fr	Self Registration	No	CI/CD	GitLab Pages, Security Testing, Analytics, Error Tracking (Sentry)
gitlab.math.unistra.	LDAP IRMA	External users	CI/CD with shared runners	Pages with personalized domains, artefact repository, Docker image registry
ESRF (European Synchrotron Radiation Facility) gitlab.esrf.fr/	Internal Directory	Invitation, ESRF	CI/CD	Pages, artefact repository, Docker image registry

A.2. Laboratories

Forge	Log-In	Noe-ESR Involvement	Continuous Integra-tion	Other Services
CRIStAL gitlab.cristal.univ-lille.fr	LDAP CRIStAL			
IRIT (Toulouse Computer Science Research Institute) gitlab.irit.fr	Directory	LDAP Guests		
FRESNEL gitlab.fresnel.fr	Institut Fresnel + guests			
LIMOS (Laboratory of Informatics, Modelling and Optimisation of Systems) gitlab.limos.fr	LIMOS Directory	Guest in the directory	CI/CD	
LIP6 gitlab.lip6.fr	LDAP LIP6	No	GitLab CI (no shared runner)	
LIRIS gitlab.liris.cnrs.fr	LDAP LIRIS	Invitation	GitLab CI (shared and dedi-cated)	Pages, mattermost
OBSPM gitlab.obspm.fr	LDAP	No	CI/CD avec runners partagés	Pages, Docker image registry, mattermost
OCA (Côte d'Azur Observatory) gitlab.oca.eu	eduGAIN (only OCA members can create projects)	Invitation	CI	Pages, gestionnaire d'artefact

B | Contributors

B.1. Members of the Software and Sources Codes College

- Ludovic Courtes, Inria
- Roberto Di Cosmo, Inria/Paris Cité University
- Sébastien Gérard, Paris-Saclay University/CEA List
- Timothée Giraud, CNRS
- Jean-Yves Jeannas, Lille University/AFUL
- Nicolas Julien, IMT Atlantique
- Daniel Le Berre, University of Artois/CNRS
- Violaine Louvet, CNRS/GRICAD (Grenoble Alpe Research - Scientific Computing and Data Infrastructure)/University of Grenoble Alpes
- François Pellegrini, Bordeaux University/CNIL
- Nicolas Rougier, Inria/Bordeaux University/CNRS
- Francois Sabot, IRD
- Samuel Thibault, Université de Bordeaux

B.2. Guests

- Denis Arrivault, David Margery Inria
- Céline Blitz CIRAD
- Gérald Dherbomez CNRS INS2I
- Alban Espie-Guillon, Pierre-Yves Gibello, Antoine Mottier OW2
- Bastien Guerry DINUM (French Interministerial Directorate for Digital Services)
- Alexis Kauffmann DNE (Division for Digital Technologies in Education) MENJ
- Philippe Krief Fondation Eclipse
- Christian Poirier INRAE
- Jean-Christophe Souplet OpenEdition

B.3. Communities

- GDR GPL (Programming and Software Engineering Research Group)
- Réseau Calcul (Calculation Network)
- Réseau DevLog (DevLog Network)

C | Questionnaire

The following questionnaire was sent to the GDR GPL, Réseau Calcul and Réseau DevLog in September 2022 with a preliminary version of the report.

C.1. Forge URL

This information will help to provide unique identification of the forge in question.

C.2. Who is providing / maintaining this forge?

Please select a response below

- A person
- An internal structure (team)
- A laboratory
- An institutional structure: Unité de formation et de recherche (Training and Research Unit), higher education institution, university
- Other:
- No response

C.3. Is this a public forge?

A forge is deemed to be public if all or even part of the source codes that it hosts can be accessed by everyone without being identified.

For a private (or internal) forge, you must be identified to access projects.

If you provide information about a private forge in this questionnaire, it will not be published in the report but it will give us an insight into the names and features of private forges.

- Yes
- No
- No response

C.4. How do you log in to this forge?

Users generally log in using a directory.

Some forges allow the use of identity forges in order to log in to them (renater and eduGAIN).

Some forges allow account creation while others do not.

Tick one or more response

- Internal directory
- RENATER
- EDUGAIN
- Invitation
- Self-registration (free account creation)
- Other:

C.5. Which services are available?

There are many potential additional services related to the forge. The most common are continuous integration, artefact management, Docker images and static code analysis tools, such as Sonarqube.

Tick one or more response

- Continuous integration
- “GitLab pages”
- Continuous deployment
- Artefact manager
- Docker image manager
- Quality assurance (Sonarqube)
- Other:

C.6. How are the projects structured?

One of the difficulties of public forges is managing how projects are structured. There is not necessarily a project linked to a particular person, but instead it is linked to a research team, a scientific project or even a research laboratory.

In this case, a tree-structure for projects can be created.

This in turn may cause an issue around cross-team, cross-project and cross-laboratory projects.

In response to this question, please feel free to outline how the software projects on this forge are structured.

C.7. Free comment

Feel free to comment on the working document.

D | Glossary

C

commit modifying unit

continuous integration ability of a forge to allow the automatic construction of the software from all of its sources, based on certain parameters

F

forge collaborative software development tool

***fork*/divergence** alternative development of source code

G

git outil de gestion de versions

I

issue an incident or malfunction reported online, or a proposed software improvement

M

***merge request* (MR)** proposed amendment

P

platform website running software that makes it possible to access specific features via a web browser (for example: a remote learning platform)

***pull request* (PR)** a synonym for merge request (the term used varies from platform to platform)

R

RENATER the French National Telecommunications Network for Technology, Education and Research

S

software engineering computer-science field focusing on the life cycle of software projects and how to manage them

Software Heritage international initiative aiming to preserve for the future the source codes of software with public source codes

sovereignty the ability of an individual, a group or state to preserve its access or usage data without any control by outside individuals or bodies

Selective bibliography

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/informat

ion-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[illegible]7. Disponible sur : https://doi.org/10.1371/journal.pbio.1001745