



Electronic Communications of the EASST
Volume 86 (1) Year 2026

deRSE26 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany (Part 1)

Edited by Bernd Flemisch, René Caspart, Bernadette Fritzsch, Florian Goth, Jean-Noël Grad, Florian Thiery

DLR's Inner Source Policy

Carina Haupt, Tanja Führer

DOI: 10.14279/eceasst.v86.1.2814

License: This article is licensed under a CC-BY 4.0 License.

DLR's Inner Source Policy

Carina Haupt[1] **and Tanja Führer**[2]

[1] carina.haupt@dlr.de, https://www.dlr.de/en/sc
Institute of Software Technology
German Aerospace Center (DLR), Berlin, Germany

[2] tanja.fuehrer@dlr.de, https://www.dlr.de/en/sy
Institute of Lightweight Systems
German Aerospace Center (DLR), Braunschweig, Germany

Abstract: The aim of the DLR Inner Source Policy is to encourage institutes to share their software within the DLR. This helps avoid duplication of development work, supports knowledge transfer, and improves exchange between institutes. By publishing this policy, we want to encourage others to adapt an Inner Source Policy in their organizations. Many software projects are developed at DLR institutes, and not all of them are intended for publication—for example, some are kept internally to retain certain intellectual property (IP) within the DLR. Publication is also not advisable for software that addresses DLR-internal issues. Nevertheless, an internal release requires regulations that balance the respective interests of the institutes. DLR Inner Source Policy regulates the use, collaboration, and distribution of software within the DLR. It consists of three parts: the Inner Source Basic Guideline regulates the DLR-wide handling of Inner Source, the User Agreement sets out rules for each software project according to which it may be used at DLR, and the Contribution Guide specifies the conditions under which contributions can be made to the software on a project-specific basis. In this paper we present the obstacles researchers face regarding sharing their software, introduce the DLR Inner Source Policy, and explain how it addresses these obstacles.

Keywords: Inner Source, Policy, Research Software

1 Introduction

Software has become a major factor in enabling researchers to answer increasingly complex research questions by allowing them to perform analyses or simulations or use specialized hardware and infrastructure [HDB+25]. In the spirit of advancing research as fast as possible, such software should be shared as widely as possible, ideally as Open Source; alternatively, as proprietary or even commercial software. However, in practice there are reasons not to share software broadly. In the following, we list three typical types of software that may not be shared:

Software containing protection-worthy intellectual property (IP) Research organizations are in competition for funding and therefore reluctant to share all of their IP as Open Source in the form of software. This competition can even carry forward inside the organization

where different institutes apply for the same funds. Sharing software on a proprietary basis or even commercially might be possible without sharing the respective IP. But even this can be prohibited due to export control restrictions, funding requirements, or due to license restrictions of third-party code.

Software managing internal systems Software that uses, controls, or configures hardware or infrastructure unavailable outside the organization offers little value for outside parties. Examples are software to control self-built hardware or to interact with a self-implemented infrastructure, such as clusters or data storage solutions.

Software implementing administrative processes Software implementing organization-specific administrative processes is also not valuable for those outside the organization. Examples are tools for time recording or for implementing approval processes.

The result is that not all software developed in the research context is published (proprietary or as Open Source), At DLR around 90% of unpublished software even stays in the groups or departments where it was created (see 2), despite its potential of being reused or built upon by others within the research organization. Inner Source (IS) is an approach to address the issue of sharing software inside an organization. The idea behind IS is the use of open source software development best practices and the establishment of an open-source-like culture within organizations [CR16]. The goal is to enable the joint use and development of (research) software beneficial to everyone. To implement this approach within an organization, regulations need to be defined to address, among other things, who is the software owner within the organization, how the software may be used within the organization, and how contributions can be made. Therefore, we implemented *DLR's Inner Source Policy*. Currently, the policy is in the adoption phase; therefore, we will only present the policy itself as well as the obstacles it shall mitigate or even overcome. While the policy was developed for the DLR, we encourage others to adapt this policy for their own organizations.

The remainder of the paper is structured as follows:

- We characterize software development at DLR to illustrate the context (Section 2).
- We describe obstacles faced by researchers regarding sharing their software within their organization (Section 3).
- We present the *DLR's Inner Source Policy* (Section 4).
- We explain how the *DLR's Inner Source Policy* addresses the mentioned obstacles (Section 5).
- Finally, we present a conclusion (Section 6).

2 Software development at DLR

The German Aerospace Center (DLR) is a large German research organization and conducts research in the domains of aeronautics, space, energy, and transportation, as well as security as

a cross-cutting topic. Nearly 12.000 employees work in 54 different institutes and facilities at 30 different locations. Each institute and facility is assigned to one of the main four domains and addresses one topic of this domain. Their research topics are interconnected and sometimes even overlap. At the same time, each institute is independent and has an individual structure and goals.

Software development plays a continuously increasing role in DLR's research activities. More than 25% of DLR's personnel costs are spent on software development. Reflecting the various research domains, there is also a wide range of developed software. Typically, it includes areas such as simulation and modelling, flight control, signal and data processing, knowledge and data management, visualization, communication, and administration. Most software developed at DLR can be categorized as *research software*, *software managing internal systems*, and *software managing administrative processes*. Most projects at DLR belong to the category of research software, whereby most software is neither published nor shared openly within DLR.

DLR published a brochure offering guidance for using open source software as well as publishing software as Open Source [BHJ+22] and offers further support on this topic [SH18]. While DLR encourages the publishing of software as Open Source, it restricts this to software which does not offer any commercial gain. Therewith, and due to the effort to publish software (see 3), most projects are kept internal. Unfortunately, DLR wide numbers do not exist, but for one of the authors institute only 127 of about 500 projects are published as Open Source.[1]

The internal DLR GitLab instance currently holds over 50.935 projects of which only 5.003 are set to *internal* thus making them available to everybody within the DLR.[2] The visibility level *public* is disabled within DLR's GitLab instance, meaning that 90% of projects have restricted access and only 10% are shared within the DLR. While DLR's GitLab instance also contains non-software projects, these numbers show that publishing projects internally is currently not a common practice, although it already takes place.

3 Obstacles to sharing software (within organizations)

In the spirit of advancing research, all software should be shared as widely as possible. This mindset conflicts with reality.

There are different types of reasons why software is not shared widely: from reasons of human psychology like personal emotional barriers, through practical operational burdens, to strategic business risks resulting in organizational concern.

The following obstacles have been identified during our work in DLR (see 4) and most of them have already been investigated by other researchers. While the referenced literature mainly addresses the industry world and we write of developers, the obstacles are also valid in the research context.

1. Developers think that others won't use their software anyway.

A cultural barrier is the so-called "Not Invented Here" (NIH) syndrome, which states that developers prefer their own solutions. They may hesitate to accept code from outside their

[1] Statistics from 07/05/2026
[2] Statistics from 07/05/2026

departments or teams, for example if it does not follow their standards or is not "generic" enough [ECCM18]. Furthermore, understanding foreign code can be a significant challenge, which limits the usability [SBAF11].

In addition to developers' tendency to prefer using their own code, they often struggle with the poor discoverability of existing components [CUB23]. Because developers often spend a considerable amount of time searching for the needed artifacts, it can be easier to simply write new code from scratch.

In research this obstacle is reinforced by the lack of experience in searching for and evaluating others' code, based on the fact that most researchers develop software for the express purpose of conducting research lack proper training to do so [CWR+22].

2. **Sharing code makes developers anxious regarding the quality of their code.**
   Inner source encourages developers to share their work and code early and often, which can make developers uncomfortable. They fear potential criticism regarding code quality, because others may see the internal product before it is finished or polished [ECCM18]. Furthermore, hierarchical corporate and research structures create pressure to meet delivery deadlines over quality [DGMN02]. Also, developers cannot stay anonymous behind nick names as in open source projects; the code can be linked directly to them as employees [CR16, MM08]. Furthermore, conflicts might arise between producing results for project deadlines and the development of fully general functionality, which is essential for software that shall be shared [GGH06]. Therefore, Inner Source can be resource-intensive, especially when it's added to other priorities. This includes, among other things, ensuring a sufficient level of quality, addressing security and maintenance concerns, and providing comprehensive documentation.

   In research resources are always scarce and additionally reputation is an important factor for the success of researchers. Based on our experience, researchers tend to be especially afraid to share non-ideal work.

3. **Supporting users, implementing feature requests, and integrating contributions means additional work.**
   Developers often struggle to balance long-term refactoring efforts with building new features for internal stakeholders [SBAF11]. Also, evaluating and accepting contributions from interested developers along with keeping up with continuous code updates represents a time-consuming burden [GGH06]. The personal accountability mentioned in the previous section, combined with limited time, creates tension, because the extra effort to review and maintain their work is seen as a challenge. Finally, developers might dread having to deal with support requests from others who use their software due to the additional work combined with a lack of funding and time. Especially in research, were funding for this kind of activities is non-existent in most cases.

4. **Sharing code could lead to strategic disadvantages.**
   Organizations may also fear strategic consequences, especially for software containing protection-worthy IP. Institutes could be concerned about the unauthorized adoption or use of software by other institutes. There is a particular fear that the software could be

used without the developers' or the home institutes' knowledge during the acquisition of new projects or even to produce results within existing projects. Another concern can be the unauthorized download and subsequent resale of the software.

Since the DLR owns the rights on all software created within the DLR, the decision if a software is shared, proprietarily, as Open Source, internally, or not at all, has to be made by the head of the institute, but might be delegated to the head of the department or even to the head of the research group. Therewith, researchers experiencing these obstacles not only have to tackle them, they also have to convince their superiors to allow them to do so. In contrast, at university, the rights for the software are often with the researchers themselves.

These obstacles all seem valid reasons not to share software broadly, even within a large research organization. This is why software often stays in the group where it was created. Nevertheless, it is possible to mitigate or overcome these obstacles and benefit from sharing code (at least) within the research organization.

4 DLR's Inner Source Policy

The Inner Source Policy provides a framework for internal sharing and consists of three parts:

Basic Guideline Sets out the DLR-wide guidelines for the use of Inner Source.

User Agreement Project-specific conditions under which the software may be used within DLR.

Contribution Guide Project-specific conditions under which contributions may be made to the software.

Additionally, an extension to the projects' README is provided to make the inner source status visible to users and to provide links to the documents as well as further helpful information.

For each of these project specific parts (User Agreement, Contribution Guide, README), at least one customizable template and a README describing its usefulness and usage are provided. Additional READMEs describe the role of the respective part as well as the usage of the templates. Technical terms used in the templates are explained in a glossary.

The minimal requirement to make a project Inner Source within DLR is that it is labelled as 'inner-source' software and made available there. For these projects, the Basic Guideline applies automatically. Adding a User Agreement, a Contribution Guide and extending the README (see figure 1) is strongly encouraged, but not mandatory. The provided templates for these documents are options provided by the authors of the policy and can be freely adapted or even created from scratch.

Within DLR, the adaption of the project specific documents has to be verified by the superior responsible for the project (see 3).

The policy itself is publicly available and published under CC-BY-4.0 [BHH+26], thus enabling other organizations to use and adapt it to their needs.

The policy was developed by approximately ten researchers from different DLR institutes over a period of four years, concurrently with their regular work, based on an identified need within the DLR. The group formed at a DLR internal knowledge exchange workshop on software

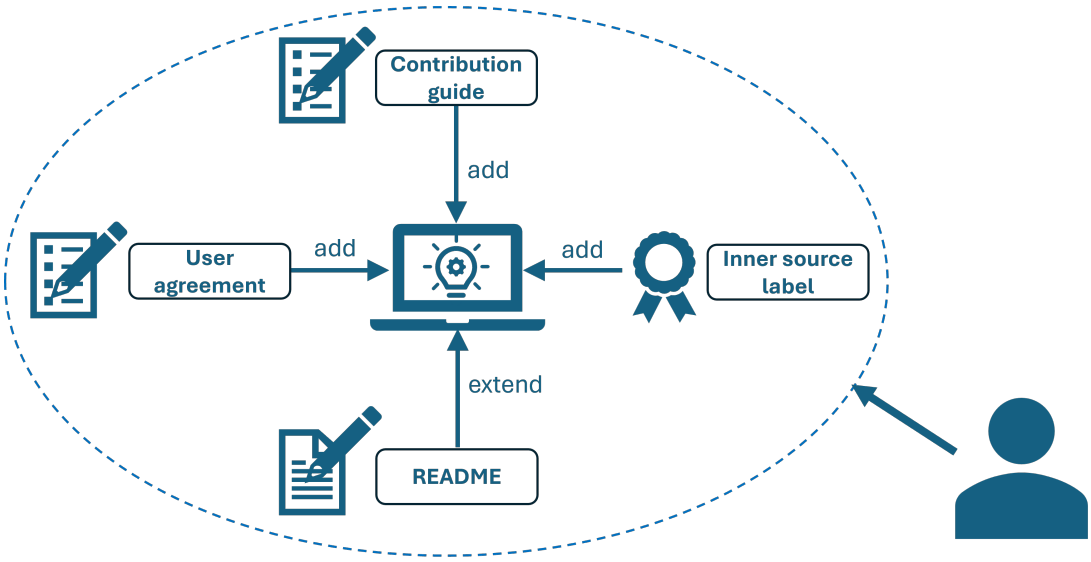


Figure 1: Recommended process for publishing software as Inner Source.

engineering [SHMS19] in 2023, after the topic already came up 2022 at the same workshop series. While having different domain backgrounds, all authors of this policy are highly invested in improving the quality of software engineering within the DLR and have a self-interest in the topic of Inner Source. One researcher is also part of the DLR's open source consulting team. On a yearly basis, feedback and input from the DLR software engineering community were collected via the previously mentioned workshop series. Additionally, the software engineering network, consisting of one member of each institute and facility of DLR holding the official role of the software engineering contact person, was consulted. In total, about 200 persons were involved in the development of the Inner Source policy. While this represents only about 2,4% of the DLR's employees, these employees are among those most actively involved in software development within the DLR [HS17].

The basis for the policy was the input collected during the 2022 and 2023 workshops of the knowledge exchange workshop series. The input was based on experiences and current struggles the workshop participants faced in their work. These were collected and prioritized during an interactive session (2022) as well as a bar camp (2023). From then on, the policy was created iteratively. Workshops participants and network members revised the documents regarding understandability, completeness and usability. In the first iteration, this was done by merely reading and discussing them in interactive sessions during the knowledge exchange workshop as well as an on-site network meeting in 2024. In the second and third iteration in 2025 and 2026, the respective version of the policy was actively used by the workshop participants on their own software projects under the guidance of the policy's author. Discussions about the obstacles faced in the daily work of the session participants were part of each of these iterations, since the obstacles directly influenced the need for specific content in the documents. Finally, in 2026

everybody interested was invited to improve and polish the final version of the policy. In each iteration feedback was collected, discussed, and, if seemed eligible by the authors, implemented. The discussions happened in biweekly meetings while the implementations were performed in between these meetings. To ensure transparency and allow everybody to participate, the policy was developed in DLR's GitLab instance using branches and forks combined with merge requests.

4.1 Basic Guideline

The Basic Guideline implements the DLR's inner source strategy in practice. It provides rules, obligations, guidelines, and information for all DLR software projects regarding use, maintenance, and further development of the software and regulates the DLR-wide fair handling. Therewith, it builds the basis for a trusting working relationship, whilst protecting the IP of the original development teams. The Basic Guideline enables all institutes to make efficient use of existing expertise and ensures the quality, traceability and long-term usability of the software. Its structure is based on the software life cycle and is divided into the topics ownership/contacts, deployment, usage, external distribution, and quality assurance and support. Thereby, ownership is defined as the right to decide on topics of collaboration, use, and further development. This does not necessarily mean that the agreement of the superiors is not needed for some topics, i.e. external distribution (see 3). Ownership can be identical with authorship, but can also be restricted to certain organizational units or persons. For institutions, such as universities, where the rights to the software lie with the researchers, ownership is identical to the legal concept of authorship. See table 1 for a short description of the topics.

Topic	**Description**
Ownership/Contact	Clearly state the owners and provide a current means of contacting them.
Deployment	Make the source code, user agreement, license information, contribution guide, references, and documentation easily accessible. Regularly review and update these materials. Provide the project without any guarantees.
Usage	Allow usage by everyone within DLR. Provide further details in the user agreement.
External distribution	External distribution requires approval from the software owners stated under Ownership/Contact.
Quality assurance	Ensure use-case-appropriate quality based on the DLR software engineering guideline [SMH18, HS17]. Must be at least application class 1 for inner source software.
Support	Define the support structure in the contribution guide. Providing no support is a valid strategy.

Table 1: Topics covered by the Basic Guideline of DLR's Inner Source Policy.

4.2 Contribution Guide

The contribution guide provides the conditions under which contributions (code, documentation, etc.) can be made to the software and can be adjusted individually for each project.

The customizable template regulates all practical and legal framework conditions through a clearly structured set of single- and multiple-choice options.

The contribution guide covers the topics contributions, technical quality requirements, change management, review and the allocation of authorship and exploitation rights, whereby is defined as the right to distribute, publish, and re-license software. For a description of the contents see table 2.

Topic	Description
Contribution	Specify whether contributions are welcome or not yet desired.
Technical quality	Specify the criteria that contributions must meet, e.g., scope of the software, test coverage, and license compliance.
Change management	Define the contribution workflow, e.g., using a fork, merge request, or issue as the starting point for a merge request.
Review	Specify whether contributions are reviewed and who may perform the review.
Authorship/Exploitation rights	Specify where authorship is documented, who is documented as an author, and whether exploitation rights are granted.

Table 2: Topics covered by the contribution guide template in DLR's Inner Source Policy.

The contribution guide thereby corresponds to contribution guides used for open source projects. This is by design, to allow an easy transition in case the project should be published as open source software. But there is also a major difference to an open source contribution guide. Exploitation rights can be retained by the owner because DLR is a single legal entity; thus, this topic of the contribution guide is governed by internal regulations rather than copyright law.

The basis for the contribution guide is a template, designed as a modular checklist, containing a clearly structured range of single- and multiple-choice options. All sections are optional and can be extended or completely removed if desired. In addition, additional sections can be added if needed. The template covers all relevant topics regarding contribution and offers recommended and typical options to address these topics. Thus, it enables project owners to create a contribution guide rapidly and with minimal effort, while on the one hand ensuring not to miss important topics and on the other hand allowing complete flexibility. By choosing appropriate options for each topic a project-specific contribution guide is created.

4.3 User Agreement

The user agreement provides rules and conditions under which the software may be used within DLR.

However, the user agreement is not a legal license, because DLR, as a single legal entity, cannot enter into a contract with itself. Instead, the user agreement is governed by DLR's internal regulations. Similar to the contribution guide, the user agreement can be adjusted individually for each software project.

For the user agreement three different templates are provided, whereas two are based on the open source license types *permissive* and *with copyleft* and one is *customizable*. By building on existing open source licenses, we can build on existing understanding of these license types. Therefore, the user agreement can be easily adapted by project owners while being easy to understand by the users of the software. By also offering a customizable template, we allow the flexibility needed by some projects due to some of the obstacles mentioned in section 3, even if this might reduce the understandability for users.

The copyleft template emphasizes that the inner source software, as well as any software built upon it, must remain available within DLR; it must also be managed as inner source software. While not directly based on it, it follows the idea of the GPL[3] license. The permissive template does not contain such restrictions, but allows (re)use of the software within DLR without any further restrictions. It is an inner source adaptation of the MIT[4] license. The customizable template is designed in the same way as the contribution guide template (Section 4.2). It regulates all practical and legal framework conditions through a clearly structured set of single- and multiple-choice options, from contact persons, author attribution and forking rights, through the allowance not only to use but also to reuse the project, license derivation, publicity, and restrictions in use. See table 3 for a list and description of these topics.

All sections are optional and can be removed if desired. Sections can also be extended, or new sections can be introduced if deemed necessary. By choosing appropriate options for each topic, a project-specific user agreement is created. Despite not being a license, the user agreement is stored as LICENSE.md to benefit from GitLab's and GitHub's license file detection features.

4.4 README template

The README is the basic documentation of the software project and contains or links to all relevant information. For inner source projects we strongly recommend to add the following information, if not already covered by the existing README: Contacts, scope, support, link to contribution guide, license information, and optionally stating the contributors, see also table 4. The contributors could also be stated in other places, which are suggested in the contribution guide template.

A template is offered for the README as well, but is restricted to the mentioned sections. In contrast to the other templates, this one is a mere extension for an (hopefully) existing file. For projects considering to be shared as Inner Source we expect a README to exist, while files like a user agreement or a contribution guide most likely do not exist yet.

[3] https://www.gnu.org/licenses/gpl-3.0.en.html
[4] https://mit-license.org/

Topic	Description
Contact	Provide information on where contact persons can be found, e.g., maintainers of the project, CITATION.cff, or README.md.
Attribution	Specify when and in which publications the authors of the software must be named, e.g., in the original source code distribution, papers, or publications of adapted code.
Reuse	Specify the conditions for reusing the code or parts of the code in another project. Distinguish between different types of projects where applicable, e.g., internal DLR projects and projects funded by third parties.
License	Specify the license constraints that apply to the resulting software when reuse is permitted, e.g., copyleft (reciprocal) licenses or no restrictions.
Publication	Specify the conditions for publishing the software in whole or in part, e.g., including parts of the software in closed-source software or selling it as part of closed commercial software.
Publicity	Specify whether promotion of the software outside DLR is permitted and, if so, which parties must be named.
Restrictions	State any restrictions based on patents, license agreements, or other applicable conditions.
Further	Specify any additional conditions that do not fall under the previous sections.

Table 3: Topics covered by the customizable user agreement template in DLR's Inner Source Policy.

4.5 Findability

Findability is a major element of the inner source concept. Software that is not found will not be reused or contributed to unless recommended directly to others. The topic of findability of software is addressed by many others, i.e. the FAIR movement [WDA+16, CKB+22], and will not be examined here. At DLR an internal GitLab instance exists where inner source software should be listed by attaching the label *Inner Source* and setting the visibility level at least to *internal*. While software outside the GitLab instance exists, e.g., on institute-specific GitLab instances or GitHub, such software in most cases does not qualify as inner source software. But in cases where it does, we suggest to either move the software to the central DLR instance or create an empty project with the same title and project description and a link to the respective project as well as information on how to gain access.

5 How the policy addresses sharing barriers

The content of the Inner Source Policy allows developers to address possible obstacles they face in their specific projects, see section 3. In the following, we describe how the policy and specific

Topic	Description
Contact	Provide specific contact information. Reference this information from other files where applicable.
Scope	Describe the scope of the software.
Support	Describe the level of support provided by the developers.
Contribution	Provide a link to the project-specific contribution guide.
License	State that the software is Inner Source and reference the project-specific user agreement.
(Optional) Contributors	Provide a list of contributors or a link to such a list.

Table 4: Topics suggested to be added to the README file of an inner source project by the README template of DLR's Inner Source Policy.

options in the templates address these obstacles.

1. Developers think that others won't use their software anyway.

The main aspects of this obstacle are developers preferring their own code (NIH) as well as findability.

The aspect of developers preferring to write code on their own cannot be addressed by an Inner Source Policy alone. Such a cultural change requires community practices and leadership support. But the underlying reasons for NIH as well as the aspect of findability can be addressed. By adding an appropriate label and setting the correct visibility (Section 4.5), researchers are able to identify software they legally could use. By adding a README including the scope of the software as well as including a way to contact the software owners (Section 4.4), researchers are also able to assess if the software is what they are looking for.

The *Inner Source* label also directly indicates the minimal expected quality of the software. The Basic Guideline, requiring projects to fulfill use-case-appropriate quality based on the DLR software engineering guideline [SMH18, HS17], which for Inner Source is at least application class 1 (see Section 4.1). With this, potential users and perhaps even contributors are given a well-known standard as well as a better indication than with open source software, where indicators for quality have to be extracted first. The same guideline will be used for any software project that may be developed using this inner source software. While this may still require adaptation of inner source software to fulfil a higher application class if the use case of the overlaying software requires this, it is at least based on the same standard.

2. Sharing code makes developers anxious regarding the quality of their code.

This obstacle has two aspects. On the one hand, developers fear the amount of work necessary to make the code usable by others. On the other hand, developers fear that low quality code might make them look incapable.

As mentioned before, the Basic Guideline requires a minimal set of quality for inner source

software (at least application class 1 according to DLR's Software Engineering guidelines). Thus, the policy might give the impression to increase the requirements regarding the quality of the software. But, in DLR, every software used or contributed to by more than one person has to fulfill the recommendations pertaining to application class 1. Only software developed solely for a developer's own usage, which also has no immediate value for the institute and is not developed over a long time, falls under class 0 and has to fulfill none of the recommendations. Thus, most likely, any software of interest for Inner sourcing has to fulfil this requirement anyway.

However, the recommendations of application class 1 of DLR's software engineering guideline only cover the most basic requirements to make software usable by others. Most developers apply higher standards on software they share, due to the need to present themselves as capable. The solution here is in the README template requiring description of the scope of the software as well as the offered support level (see Section 4.4). By describing the scope, expectations regarding the quality and generality of the software can be managed. For example, a proof of concept or an analysis script accompanying a paper will not raise expectations of high-quality code or focus on usability. Same goes for stating the offered support level. Active expectation management can be a great possibility to lower the pressure on the developers and not disappoint possible users.

3. **Supporting users, implementing feature requests, and integrating contributions means additional work.**
   Yes, these tasks mean additional work.

   While good documentation, including how-to's and tutorials, can decrease the amount of support asked for by users, creating and maintaining those still means additional work. Contributions as well as feature requests can improve the quality, usability, and scope of your software; thus, the benefits might outweigh the work. But when putting software Inner Source you do not have to offer support, implement feature requests, or even accept contributions. While this is recommended, the README (section 4.4) offers an option to state that you will not offer any support. The contribution guide offers similar options regarding contributions and feature request. Software owners can state that they do not accept contributions at all or, if they are welcomed, select from a list of criteria to define what contributions have to fulfil (i.e. accompanying test) to minimize the effort in integrating them. Further, it can be defined how contributions should be performed, whether you offer support for developers, whether there will be any review of the contribution, and whether contributors are required to fix issues occurring due to their contribution.

   Although offering support as well as accepting feature requests and contributions are recommended for inner source projects, we accept that this might not be possible for some projects. But to not close this door completely, the contribution guide suggests to always end with an invitation to contact the project owners if an adaptation of the contribution guide is desired.

4. **Sharing code could lead to strategic disadvantages.**
   The main aspect of this obstacle is that others, within or outside the research organization, not only benefit from freely shared work but may also achieve a strategic advantage by

doing so. Outside the research organization, licenses are used to prevent this. Within the research organization, the Inner Source Policy described in this paper is intended to prevent this.

The Basic Guideline guide states that for each software no distribution is allowed without the approval of the owner. The project specific user agreement as well as the contribution guide templates define this in more detail. They offer a wide range of options to define which usage or distribution is accepted and in which contexts. They differentiate between using the software in DLR internal vs. third-party-funded projects, control external promotion, define allowed usage in commercial contexts, license derivations, ensure attribution in publications as well as that the software creators need to be informed about any use and/or distribution. This ensures complete control and knowledge over how and where their software is used. This is of particular importance for software containing protection-worthy IP.

Thus, DLR's Inner Source Policy shall mitigate or overcome the mentioned obstacles and puts software creators' as well as their superiors mind at rest when software is shared within the research organization.

6 Conclusion

In the spirit of advancing research, software should be shared as widely as possible. Yet not all software created in the research context is suitable for public release as Open Source or proprietary software. Software that manages internal systems or administrative processes offers little value outside the organization, and software containing protection-worthy IP can be strategically sensitive for the developing institute (Section 1). For such software, sharing within the research organization is often a pragmatic alternative: it can reduce duplicated effort, support knowledge transfer and preserve protection-worthy IP while still enabling reuse. At DLR, this potential is large but largely untapped: about 90% of the GitLab projects are not visible throughout the DLR, although the instance also contains non-software projects (Section 2).

We identified four groups of obstacles to internal sharing, most of which also apply to external sharing: doubts that others will use the software, anxiety about code quality and reputation, the burden of support and contributions, and fear of strategic disadvantage (Section 3). These barriers are based on anecdotal evidence from workshops and consultations at the DLR rather than on formal studies, but they align with findings reported in the literature.

In response, we present DLR's Inner Source Policy as a three-part governance framework for multi-institute research organizations: a DLR-wide Basic Guideline, and project-specific user agreements and contribution guides supported by README extensions, templates, and a glossary (Section 4). The policy maps these components to the identified obstacles by clarifying permissions and expectations. This is done, for example, through reuse and publication options, optional support and contributions, quality expectations linked to DLR's software engineering guideline and measures for findability (Section 5). By building on familiar open source concepts, the templates also ease a later transition to Open Source, where appropriate. Inner Source can thus serve as a stepping stone between closed development and public release for some soft-

ware projects. For other types of software, especially software to manage internal systems or implement administrative processes, Inner Source can be the final step.

The first version of DLR's Inner Source Policy is currently being published and advertised. Therefore, we are in the early adaptation phase. Currently, about 20 projects have applied the Inner Source Policy. These early adopters are mainly the authors of the policy as well as the participants of the knowledge exchange workshops in 2025 and 2026. The policy will become part of the next version of the DLR's software engineering guideline. As part of its publication, the Inner Source Policy will be advertised in internal DLR-wide newsletters, in the news section of DLR's intranet, as well as the software engineering network and community of the DLR. Currently, also an inner source template for GitLab is developed to allow easier adaptation of the user agreement, contribution guide and README.

The policy regulates permissions and expectations; it does not by itself increase adaptation, improve discoverability beyond labelling and documentation requirements, or align institute-level incentives for reuse. Sustainable Inner Source at DLR therefore also depends on complementary measures such as community practices, central tooling, training, and institutional support.

By publishing the policy outside of the DLR, we hope for adaptation by further research organizations. This hope is based on the high interest during the policies presentation at the conference for research software engineering 2026 in Germany (deRSE26), as well as the need we identified during our interactions with the DLR community for regulations that govern not only cross-institutional but also the institutes' internal use of software. Therefore, we seem an adaptation of an Inner Source Policy sensible not only for research organizations the size of the DLR, but also smaller institutions. This may require tailoring the policy to an organization's own quality standards, processes, tools, and regulations. While this entails additional effort, DLR now offers an extensive set of regulations to build upon, since this Inner Source Policy as well as the Software Engineering Guideline are published under Creative Commons licenses and reuse is actively encouraged.

Acknowledgements:
The authors wish to thank the members of the inner source DLR working group for their valuable discussions, implementation feedback, and collaborative reviews. In particular, we thank Lukas Goormann, Daniel Mohr, Daniel Krajzewicz, Björn Brötz and Tobias Schlauch.

Data availability: DLR's Inner Source Policy is available on zenodo: [BHH+26]

Declaration on the use of AI: AI was used to correct spelling, grammar as well as make minor phrasing improvements. Apart of this AI was not used in the creation of this document or the underlying policy.

Bibliography

[BHH+26] B. Brötz, B. Hendriks, C. Haupt, D. Krajzewicz, D. Mohr, D. Seth, K. Franke, L. Goormann, T. Führer, T. Schlauch. DLRs Inner Source Policy. 2026.
doi:10.5281/zenodo.18860824
https://doi.org/10.5281/zenodo.18860824

[BHJ+22] M. Brochhagen, C. Haupt, T. Jäger, V. Müller, T. Schlauch, K. Stumvoll. Nutzung von Open-Source-Software im DLR Mathias Brochhagen, Carina Haupt, Dr. Till Jaeger (JBB Rechtsanwälte), Verena Müller, Tobias Schlauch, Dr. Konstantin Stumvoll. 2022.
https://elib.dlr.de/193607/

[CKB+22] N. P. Chue Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, T. Honeyman et al. FAIR Principles for Research Software (FAIR4RS Principles) Version 1.0. 2022.
doi:10.15497/RDA00068
https://doi.org/10.15497/RDA00068

[CR16] M. Capraro, D. Riehle. Inner Source Definition, Benefits, and Challenges. *ACM Comput. Surv.* 49(4), Dec. 2016.
doi:10.1145/2856821
https://doi.org/10.1145/2856821

[CUB23] X. Chen, M. Usman, D. Badampudi. Using InnerSource for Improving Internal Reuse: An Industrial Case Study. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*. EASE '23, p. 348–357. Association for Computing Machinery, New York, NY, USA, 2023.
doi:10.1145/3593434.3593466
https://doi.org/10.1145/3593434.3593466

[CWR+22] J. C. Carver, N. Weber, K. Ram, S. Gesing, D. S. Katz. A survey of the state of the practice for research software in the United States. *PeerJ Computer Science*, 2022.
doi:10.7717/peerj-cs.963
https://doi.org/10.7717/peerj-cs.963

[DGMN02] J. Dinkelacker, P. K. Garg, R. Miller, D. Nelson. Progressive open source. In *Proceedings of the 24th International Conference on Software Engineering*. ICSE '02, p. 177–184. Association for Computing Machinery, New York, NY, USA, 2002.
doi:10.1145/581339.581363
https://doi.org/10.1145/581339.581363

[ECCM18] H. Edison, N. Carroll, K. Conboy, L. Morgan. An Investigation into Inner Source Software Development: Preliminary Findings from a Systematic Literature Review. In *Proceedings of the 14th International Symposium on Open Collaboration*. OpenSym '18. Association for Computing Machinery, New York, NY, USA, 2018.

doi:10.1145/3233391.3233529
https://doi.org/10.1145/3233391.3233529

[GGH06] V. K. Gurbani, A. Garvert, J. D. Herbsleb. A case study of a corporate open source development model. In *Proceedings of the 28th International Conference on Software Engineering*. ICSE '06, p. 472–481. Association for Computing Machinery, New York, NY, USA, 2006.
doi:10.1145/1134285.1134352
https://doi.org/10.1145/1134285.1134352

[HDB+25] W. Hasselbring, S. Druskat, J. Bernoth, P. Betker, M. Felderer, S. Ferenz, B. Hermann, A.-L. Lamprecht, J. Linxweiler, A. Prat i Sala, B. Rumpe, K. Schoening-Stierand, S. Yang. Multi-Dimensional Research Software Categorization. *Computing in Science and Engineering* 27(2):59–68, März 2025.
doi:10.1109/MCSE.2025.3555023
https://elib.dlr.de/213484/

[HS17] C. Haupt, T. Schlauch. The Software Engineering Community at DLR: How we got where we are. In Hong et al. (eds.), *Proceedings of the Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE5.1)*. September 2017.
doi:10.6084/m9.figshare.c.3869782
https://elib.dlr.de/114050/

[MM08] C. Melian, M. Mähring. Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard. In Russo et al. (eds.), *Open Source Development, Communities and Quality*. Pp. 93–104. Springer US, Boston, MA, 2008.
doi:10.1007/978-0-387-09684-1$_8$
https://doi.org/10.1007/978-0-387-09684-1_8

[SBAF11] K.-J. Stol, M. A. Babar, P. Avgeriou, B. Fitzgerald. A comparative study of challenges in integrating Open Source Software and Inner Source Software. *Information and Software Technology* 53(12):1319–1336, 2011.
doi:https://doi.org/10.1016/j.infsof.2011.06.007
https://www.sciencedirect.com/science/article/pii/S095058491100142X

[SH18] A. Schreiber, C. Haupt. Raising awareness about open source licensing at the German aerospace center. In *2018 IEEE Aerospace Conference*. IEEE, Juni 2018.
doi:10.1109/AERO.2018.8396684
https://elib.dlr.de/126119/

[SHMS19] T. Schlauch, C. Haupt, M. Meinel, A. Schreiber. Analytics and Insights About Cultivating a Software Engineering Community at DLR. In *2019 IEEE Aerospace Conference*. Pp. 1–8. 2019.
doi:10.1109/AERO.2019.8741902

[SMH18] T. Schlauch, M. Meinel, C. Haupt. DLR Software Engineering Guidelines. Aug. 2018.

doi:10.5281/zenodo.1344612
https://doi.org/10.5281/zenodo.1344612

[WDA+16] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3:160018, 2016.
doi:10.1038/sdata.2016.18