Electronic Communications of the EASST
Volume 85 Year 2025

deRSE25 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany

Edited by: René Caspart, Florian Goth, Oliver Karras, Jan Linxweiler, Florian Thiery, Joachim Wuttke

Seek and You Shall Find - Or Not! Why Can't We Find the Research Software We Really Need?

Ronny Gey, Tim Wittenborg, Alexander Struck, Daniel Mietchen, Oliver Karras

DOI: 10.14279/eceasst.v85.2712

Electronic Communications of the EASST (https://eceasst.org).
Published by **Berlin Universities Publishing**
(https://www.berlin-universities-publishing.de/)

Seek and You Shall Find - Or Not! Why Can't We Find the Research Software We Really Need?

Ronny Gey[1], **Tim Wittenborg**[2], **Alexander Struck**[3], **Daniel Mietchen**[4], **Oliver Karras**[5]

[1]Helmholtz Center for Environmental Research - UFZ, [5]TIB – Leibniz Information Centre for Science and Technology, [4]FIZ Karlsruhe — Leibniz Institute for Information Infrastructure, [3]Cluster of Excellence Matters of Activity. Image Space Material at Humboldt-Universität zu Berlin, [2]L3S Research Center, Hannover

Abstract: Research software plays a pivotal role in today's research, whether as a result or as a facilitator of research processes. Therefore, it is very important for researchers to find suitable research software for their research project. We develop scenarios for searching research software, as well as personas that act as research software discoverers. We then categorize various discovery pathways for research software and analyze them according to relevant search criteria. Our results show many different discovery scenarios, a very heterogeneous landscape of discovery pathways and a lack of metadata and filtering options.

Keywords: information retrieval, information seeking, research software, FAIR, findability

1 Introduction

Research software is a fundamental part of today's research, as a tool, as a result, or as an object of study itself. And the use and with it the importance of software for research will continue to increase in the future. As a digital artifact, software, like data, has become an integral part of research, which is why it is essential to treat it sustainably. To make the research process transparent and reproducible, software should be FAIR [CKB+21], i.e. findable, accesible, interoperable and reusable. The findability aspect requires that "software and the associated metadata are easy to find for both humans and machines." [CKB+21] Specifically, this means that the software is assigned a persistent identifier (PID) and described with extensive metadata that is searchable and indexable (and FAIR itself).

However, FAIR metadata is only one necessary prerequisite for software to be found. It doesn't guarantee that software actually is found by individuals. If we look at the systems that researchers use to search for software, they have a variety of infrastructures at their disposal in which to search: a general purpose search engine is often the starting point for a search for software. Popular code repositories such as GitHub[1] make software available and also provide search functionality. Software repositories for different programming languages (e.g. python:

[1] https://www.github.com

pypi[2]; or R: CRAN[3]) or operating systems (e.g. Android: play store[4]; or the Debian package archive[5]) offer a development-centered approach for the former and a target system-oriented approach for the latter. Software is also published and can be found on publication platforms, with Zenodo[6] being a prominent example.

These are just a few examples of the wealth of search options that researchers can use when searching for software and which we will discuss in more detail in this article. The bottom line is that software is scattered over several locations which increases complexity of the discovery process. To help researchers choose the right option for their search for research software, we want to list the individual discovery pathways in this article and compare them on the basis of defined search and discovery relevant criteria. From this, we discuss the findings of our preliminary analysis. Since this is still work-in-progress, we discuss the limitations of this paper in detail and outline future directions in which the paper will be developed.

In section Discovery needs (2) we will outline typical user types for the search for research software (2.1). Based on the user types, we will then identify scenarios that trigger a search for research software (2.2). Types and scenarios will in turn be used to develop criteria for evaluating the individual search options (2.3). These criteria are used for the subsequent evaluation of the individual discovery pathways (4) which are introduced beforehands in section (3). In the following discussion (5), we address the fact that software can be searched and found in many places, that the faceting possibilities in many discovery pathways can be improved, and that the hosting or responsibility for the pathway itself lies in the hands of very different institutions.

2 Discovery needs

2.1 Research software discoverers - categorization of user types

To facilitate discovery of research software, it is important to consider the contexts in which such discovery might be taking place or be attempted, particularly by whom, with which prerequisites and for what purposes. A common framework to describe and communicate such user-centric parameters of a technical system is that of a *persona*, i.e. a fictional character described in terms of some characteristics deemed relevant for the system at hand [AP10]. The persona approach has been used in a variety of settings (e.g. HCI [PG03], healthcare [JVS17], and education [MOd15]), yet we are not aware of examples focused on research software discovery.

In this section, we will thus outline some personas we deem relevant for discovering research software. They differ in parameters that can be roughly grouped as **academic** and **technical** dimensions of the discovery of research software (see Table 1). Academic parameters would be the disciplinary background, the career stage, actual position and the institutional context. Technical parameters describe the persona's expertise in terms of using, running, building, designing, maintaining or documenting software and for which hardware.

[2] https://pypi.org/
[3] https://cran.r-project.org/https://cran.r-project.org/
[4] https://play.google.com
[5] https://packages.debian.org/index
[6] https://www.zenodo.org

Charly: PhD Student in Humanities This is a humanities PhD student with no programming experience who also does not need software development skills for their research. They want to automate simple analysis steps such as annotation and stylometry to speed up their research process. They are accustomed to intuitive user interfaces and are looking for user-friendly tools that simplify routine procedures without extensive configuration requirements. They want to manage and analyze their data efficiently. They rely on academic publications and peer recommendations when selecting digital tools. Their research questions are exclusively qualitative in nature and can usually be answered using a few software applications. The ideal discovery system would explain tool functionalities and provide examples of how they can be applied in humanities research.

Sascha: PostDoc in BioStatistics This researcher is an early-career scientist in the field of biostatistics with advanced R programming skills who frequently performs quantitative analyses. They are proficient in the R ecosystem and proactively seek new R packages to expand their statistical method set, improve his workflows, and respond to new questions in their current project. Their main objective is to identify and use software tools to analyse complex biological data. They care about efficiency, accuracy, and reliability of research software and are willing to spent time learning new tools. They frequently search CRAN, GitHub, and Bioconductor repositories for relevant tools. They value the ability to integrate tools into existing R-based pipelines and therefore prefer modular, interoperable solutions. They can also contribute to software projects by reporting problems, suggesting improvements or submitting patches. The discovery system should support filtering by statistical methods, dependencies and version history.

Zhu: Research Software Engineer (RSE) Here we have a research software engineer who creates and maintains software applications for research projects. They have practical programming skills, but did not receive intensive programming training at university. They are familiar with certain programming languages (Python, R, JavaScript) and tools used in their research area, but may not be aware of alternative solutions. They seek software that is robust, well-maintained, and compliant with FAIR principles. Their search is driven by functional requirements, performance considerations, licensing constraints, maintainability and extensibility. They are interested in both general-purpose software and domain-specific tools. The RSE also contributes to open source projects. The discovery system should provide technical metadata, version control history and user feedback on software stability.

Kim: Experienced Developer Entering a New Field This user is an experienced software developer with a background in computer science but no experience in the scientific field. They have exceptional programming skills in languages such as Java, C++ or Python, but are new to the research area with its specific tools and methods. They have been tasked with developing new research software and needs an overview of relevant software tools in this area. As Linux users, they are accustomed to working with command line interfaces and scripting languages. Their main goal is to quickly acquire knowledge of the research software landscape and identify suitable tools for their project. They value comprehensive documentation, tutorials and community support for research software. The discovery system should provide insights into software

Table 1: Dimensional Skill Sets of the Five Personas for Research Software Discovery

Dimension	**Charly** (PhD-Stud)	**Sascha** (PostDoc)	**Zhu** (RSE)	**Kim** (Dev)	**Andrea** (Lib)
Academic	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	⋆	⋆⋆
Technical	⋆	⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆

Note: ⋆ no prior experience, ⋆⋆ beginner, ⋆⋆⋆ advanced, ⋆⋆⋆⋆ expert

popularity, typical use cases, and recommended learning resources for domain newcomers.

Andrea: Data Librarian Cataloging Research Software This is a data librarian who supports researchers in finding and accessing relevant software tools and resources. They also play a crucial role in cataloging and documenting software developed by researchers within their institution, which is essential for research reporting and end-of-year impact assessment. They have a strong understanding of research workflows and data management, as well as information literacy. Their main objective is to provide researchers with advice and recommendations on research software while assessing usability, accessibility, and discoverability of research software and regularly evaluate new tools and resources. They evaluate software based on institutional policies and compliance with open science or FAIR principles. The discovery system should provide well-structured metadata, citation formats, and cross-referencing with repositories such as Zenodo or institutional archives.

2.2 Scenarios for searching research software

The personas presented in Section 2.1 have different reasons for searching for software. In this section, we show various scenarios that can trigger a search for research software. Each individual scenario can be associated with one or more personas as presented in Table 2.

Solve Task. You have a task or a problem which is very tedious, repetitive, or complex. There should be an easier, more efficient way to solve this – ideally semi-automatically. However, you don't know what tools exist or how to formulate your problem as keywords or queries.

Compare software. You want to compare software that solve the same problem. Likely, one is best fit to your specific case. But there is no comparison of properties like performance, features, ease of use, compatibility, or community support.

Develop software. You want to develop new software and make sure you're not reinventing the wheel. By reusing existing solutions, you can make sure not to duplicate effort and focus on novel work. It is complicated to make sure all relevant software is on your radar, especially where even known tools have potentially little known or disclosed features.

Feature missing. You have a tool that's almost perfect, but it is missing a specific feature. It might be possible to compensate using a different tool, but you are unaware if any other tool solves this task. Even if there was, it is questionable how you could find out if it would be compatible with your workflow.

Add features. You have developed a software that's perfect and that makes you suspicious you might be missing something. You want to find similar software to see what features and compatibilities they cover. Other than some forum threads and short related work sections, this information is hard to come by.

Measure impact. You have developed a software and need to measure the impact of your tool. While some citations are picked up by search engines, many more likely use your software without mentioning it. There is little to no indication how many downloaded, (re-)used or derived your work.

Review software. You review software available online as part of your security assessment. You appreciate open source software, especially to ensure adherence to best practices or certain standards in quality or reliability. Even if you have identified an issue, your options are limited in publishing them, especially when you want to reach all third party users of this software.

Reproduce results. You need to reproduce results found in a paper using a (specific) software. In theory, you just follow their description and referenced software stack. In practice, the often rather brief and incomplete descriptions are usually not sufficient to identify versions, niche steps, applications and settings. Difficulties identifying and accessing software are also caused by insufficient citation practices as well as link rot.

Write paper. You write a paper and need to know what other tools exist for the related work section. You don't care for extensive details, just need a reliable list to check for yourself that you're not missing any major players. Those short, but solid lists are hard to come by.

Describe software. You want to describe a software for your paper and make sure to cover all important information. While you have used this tool for a while, you are always only using it for a specific problem. To properly describe it in a cohesive background section, your surface level understanding is not enough.

Review Literature. You conduct a literature review on what software is relevant to your research question. While you extract these from scientific article mentions, you think there should be a better way for this. Sadly, authors rarely link or even reference every tool they used inside papers and the metadata is limited and heterogeneous.

Add to catalog. You want to add a software to a catalog and need more information about it. Your ontology and description fields are well-defined, but the software not so much. Despite

Table 2: Summary of existing use-cases and matching persona from section 2.1

Scenario	**Charly** PhD-Stud	**Sascha** PostDoc	**Zhu** RSE	**Kim** Dev	**Andrea** Lib
Solve task	X	X	X	X	X
Compare Software	X	X	X	X	X
Develop Software		X	X	X	
Feature Missing		X	X	X	
Add Features		X	X	X	
Measure Impact		X	X	X	X
Review Software			X	X	
Reproduce Results	X	X	X		
Write Paper	X	X	X	X	
Describe Software	X	X	X	X	
Review Literature	X	X	X		
Add to Catalog		X	X	X	X

the software still being in use, it is especially hard to come by details due to the main developer leaving the project.

2.3 Evaluation criteria for discovery pathways

Based on the personae and their characteristics defined in the previous sections, as well as the various scenarios for searching for research software, we derive evaluation criteria for the discovery pathways, presented in section 3. Furthermore, we cross-checked, refined, and extended our criteria using existing evaluation criteria for similar services, such as repositories [Con22] or library discovery systems [CY14]. The resulting list of criteria is summarized in table 3.

3 Discovery pathways

Researchers have an extensive number of systems at their disposal in which they can search for research software. In this section, we introduce various available discovery pathways. An overview of all options can be found in table 4. Each of these options is then briefly introduced.

3.1 Code repositories

Code repositories are public platforms that host code and offer management and collaborative functions. The search functions vary and are rather basic; for example, they may contain filters for programming language, license, category or deployment options. These platforms facili-

Table 3: Criteria for the analysis of discovery pathways for research software

Category	Criteria	Description	Unit
Content	Quantity	Amount of available software items in the pathway	Number
	PIDs	PID to identify items in the pathway	List
	Scope: Scientific domain	Pathway focus on specific scientific domains	List
	Scope: Publication type	Types of content available in the pathway	List
	Metadata Schema	Standardized metadata structure used for software description	List
Search	Filter options	Available filters to refine search results	List
	Search Functionality	Search capabilities provided by the pathway	Text
	Science Information	Linkage to text/data publication	Boolean
	Recommender	Recommendation service for related software items	List
Organisation	Host institution	Organization maintaining the pathway	Text
	Funding	How maintenance & dev of pathway is funded	Text
	Documentation	User documentation of the pathway and the search functionality	Text
	Ease of use	Subjective Evaluation of User Experience	Text
	User Engagement: Content	Opportunities for users to contribute content	List
	User Engagement: Governance	Opportunities for users to govern the pathway	List
	Community Size	Number of active users engaging with the pathway	Number
Technology	APIs	Availability of programmatic access to the pathway	List
	Integration w/ other Platforms	Ability to connect with repositories/databases	List
Item Evaluation	Licensing Information	Availability of software license details	Boolean
	Usage Statistics	Software downloads, citations, community size	List
	Quality indicators	Badges, Tests, Stars, Documentation,	List
	Dependencies	Information about dependencies of the software package	Boolean
	Operating system	Information about operating system support	Boolean
	Functional Category	Information about the functioning of the software package	Boolean
	Description	Description of the software package	Boolean

tate contribution and reuse. Popular examples are GitHub[7], GitLab[8], Codeberg[9], Bitbucket[10], SourceForge[11].

3.2 Mixed Content Publication Platforms

Researchers can use mixed content publishing platforms to publish their research results. Research results can be texts, presentations, posters, reports, research data, but also research software. Zenodo[12] is probably the best known example of a mixed content platform that provides

[7] https://github.com
[8] https://gitlab.com
[9] https://codeberg.org/
[10] https://bitbucket.org/
[11] https://sourceforge.net/
[12] https://www.zenodo.org

Table 4: Summary of existing discovery pathways.

Discovery Pathways	Code	Application	Metadata
Code Repositories	X		X
Mixed Content Publication Platforms	X		X
Software Repositories			
Programming Language Centered		X	X
Operating System Centered		X	X
Software archives	X	X	X
Catalogs			
Domain Specific			X
Institutional			X
National or Supranational			X
Curated Lists			X
Search Engines			
General purpose search engines			X
Specialised search engines			X
Federated search engine			X
Software Journals			X
Knowledge Graphs			X
Social Networks			X
Large Language Models	X	X	X

various research results including code. Institutional repositories, which often provide mainly texts but also software, are also considered mixed content publishing platforms.

3.3 Software Repositories

Software repositories often also refered to as package repositories are central storage locations for software packages for a specific environment or ecosystem. We distinguish between software repositories for programming languages and repositories for a specific operating system.

Programming language centered repositories For many of the common programming languages, there is a central repository in which software developed in the programming language is stored and curated (e.g. PyPi for Python[13], CRAN for R[14], or crates.io for Rust[15]). Often a Package manager is offered around the software repository, which simplifies the search and installation of software packages as well as the handling of dependencies (e.g. pip for Python[16],

[13] https://pypi.org/
[14] https://cran.r-project.org
[15] https://crates.io/
[16] https://pip.pypa.io/

Cargo for Rust[17], or Cabal for Haskell[18].

Operating system centered repositories Similar to programming language centered repositories, there are central repositories for many common operating systems that store and curate software that is offered for the respective operating system. Each of the major Linux distributions has one or several central repositories as well as one or more package managers that make it easier to search for and install packages and manage dependencies (e.g. Debian based systems rely on APT[19], pacman for Arch based systems[20], or DNF for Red Hat based systems[21]). FlatPak[22] and Snap[23] are popular examples for linux distribution agnostic package delivery. HomeBrew[24] is a popular package manager for macOS while it can also be used under Linux. Conda[25] in turn is a popular package manager, known from the Anaconda distribution popular among data scientists, which is both operating system and programming language agnostic. The app stores and the underlying software repositories of mobile operating systems also fall into this category (e.g. the App Store of iOS[26], as well as for Android the Play Store[27] or the free and open source package manager F-Droid[28]).

3.4 Software archives

Software archives preserve code for future generations. Software Heritage[29] is an academically funded initiative to collect and index source code [DC20]. They are archiving 31 different code management platforms, from which 27 are regularly crawled while 4 of them are being crawled on demand. 3 code hosting platforms do not exist anymore while the code is archived for future access on Software Heritage. In addition to the disappearance of single code hosting platforms such as Gitorious, Google Code or Bitbucket, the repositories of individual software projects can also be deduplicated or deleted. Software archives prevent this loss of software code and enable future access to software code to understand and reproduce research carried out with it.

3.5 Catalogs

While there are code management and publication platforms, there are also efforts to make research software better known and findable. Catalogs are an established concept to collect metadata and point to (a persistent) location elsewhere. Such catalogs are often hosted and sometimes

[17] https://doc.rust-lang.org/stable/cargo/
[18] http://www.haskell.org/cabal
[19] https://wiki.debian.org/AptCLI
[20] https://archlinux.org/packages/core/x86_64/pacman/
[21] https://rpm-software-management.github.io/
[22] https://flatpak.org/
[23] https://snapcraft.io/
[24] https://brew.sh/
[25] https://conda.io/
[26] https://appstore.com/
[27] https://play.google.com/
[28] https://f-droid.org/
[29] https://www.softwareheritage.org/

even curated by different institutions to ensure their long-term availability and high quality of entries.

Based on different use cases for the use of a catalog, we distinguish three basic subtypes of catalogs for research software: domain-specific, institutional, and national/supranational catalogs.

Domain specific Domain-specific catalogs focus on software that is used in a specific research domain. We find, for example, catalogs for domains like biology[30], physics[31], mathematics[32], humanities[33], astrophysics[34] or social sciences[35]. The catalog of swMATH, for example, collects URLs to mathematical software and accompanying text publications. Users benefit from pointers to related software and also (software) citation information as an indicator for its popularity.

Institutional Institutional catalogs, on the other hand, list software developed by members of the respective institution. The Research Software Directory (RSD) of the Helmholtz Association[36] or the RSD of the eScience Center of the Netherlands[37] are examples of institutional research software catalogs.

National or supranational The acquisition of software can also take place at a national or even supranational level. For example, for Germany and its national research data infrastructure, NFDI.software[38] will be launched. NFDI.software is a basic service of the BASE4NFDI consortium, which aims to become a central marketplace to improve access to NFDI research software.

3.6 Curated lists

Curated lists are collections of particularly high-quality content on a specific topic. The best known of these lists are the awesome lists on GitHub[39]. Curated lists are very similar to catalogs, but often contain less metadata, if any. They are curated by individuals, institutions or communities. The content ranges from lists on every conceivable programming language, computer science, big data, testing, domain-specific ones[40], a list about research software registries[41], lists about general research software engineering[42] and many more.

[30] https://bio.tools
[31] https://physics.tools/
[32] https://www.swmath.org
[33] https://tapor.ca/tools
[34] https://ascl.net/
[35] https://www.sosciso.de
[36] https://helmholtz.software
[37] https://research-software-directory.org/
[38] https://base4nfdi.de/projects/nfdi-software
[39] https://github.com/sindresorhus/awesome
[40] e.g. https://dh-tech.github.io/awesome-digital-humanities/
[41] https://github.com/NLeSC/awesome-research-software-registries
[42] e.g. https://github.com/hifis-net/awesome-rse

3.7 Search engines

General purpose search engines follow broad content aggregation strategies and offer a general search UI. Different approaches to search technology developed over the decades. A common approach is to crawl/harvest resources, building a local index with normalized metadata and offering fast search functionality. Examples are WWW search engines like "Google" or science focused competitors like "BASE"[43]. The term 'meta search' is sometimes utilized to differentiate this approach from, for example, 'federated' search. There, a user initiates an instant search on several remote platforms, often enabled via API calls. An example for a federated search engine dedicated to research software is "Betty's ReSearch Engine"[44] [SRW24].

3.7.1 General purpose search engines

General purpose search engines are information retrieval systems designed to index and search across the entire publicly accessible web, regardless of topic or domain. These systems (e.g. Google, Bing, and DuckDuckGo) employ algorithms and ranking mechanisms to process billions of web pages and return relevant results based on user queries. They aim to provide a broad coverage across all topics, making them the primary gateway for most users seeking information online.

3.7.2 Specialized search engines

Specialized search engines are information retrieval systems designed to focus on specific content domains, document types, or academic resources. In the scholarly context, these engines like BASE (Bielefeld Academic Search Engine)[45], OpenAlex[46], or Semantic Scholar[47] exclusively index academic content from institutional repositories, digital libraries, and academic publishers. They employ domain-specific algorithms and metadata schemas tailored to academic content, enabling precise filtering by publication type, peer-review status, and scholarly metrics.

3.7.3 Federated search engine

Federated search engines serve as meta-search platforms that simultaneously query multiple databases, repositories, and search services through a unified interface. These systems distribute search queries in real-time across various autonomous information sources and aggregate the results into a consolidated display. By implementing protocols for cross-system interoperability and standardized data exchange, federated search engines enable researchers to access diverse collections without needing to search each resource individually. A very prominent example is Bettys (Re)Search engine[48], which aims at finding ressearch software on code collaboration

[43] https://base-search.net/Search/Results?lookfor=doctype%253A6
[44] https://gitlab.com/tuc-isse/public/betty-research-engine/
[45] https://base-search.net
[46] https://openalex.org/
[47] https://semanticscholar.org/
[48] https://nfdi4ing.rz-housing.tu-clausthal.de/

platforms (like GitHub) and connecting/enhancing results with references from other databases, for example, related text publications.

3.8 Software journals

Journals specializing in software publish text articles on software publications. Since scientific communication about research results is mainly in text form, the recognition and reward system is also based on a text-based publication infrastructure, e.g. journals and book publishers. Software journals use the scientific reward system to write about software and make the software visible and discoverable. JORS[49], JOSS[50] and IPOL[51] are community initiatives whereas SoftwareX[52] is an example for a commercial journal for research software papers.

3.9 Knowledge graphs

Knowledge graphs (KGs) offer a robust pathway for identifying research software tailored for specific research purposes as they organize and interlink diverse information in a structured manner. KGs represent fine-grained semantic descriptions of entities such as software, researchers, institutions, and research topics as nodes, with edges depicting the relationships between them. This structured approach facilitates semantic search capabilities, allowing researchers to find software related to specific topics even if not explicitly mentioned in descriptions. Advanced querying capabilities using languages like SPARQL or Cypher enable complex searches for software meeting specific criteria. Visualization tools further enhance the exploration process, offering an intuitive way to understand relationships and identify relevant software. By capturing collaborative networks, knowledge graphs facilitate connections with other researchers using the same tools, fostering potential collaborations. The dynamic nature of these graphs ensures they remain up-to-date with the latest software tools and research developments.

3.10 Social networks

Perhaps not the most obvious option for finding research software, but a surprisingly common one, is the social network of a researcher. In interview studies about software discovery, participants often mentioned their academic network as a source of information about relevant research software [HG16, Gey20]. Possible ways of accessing knowledge in the social network are usually direct contact with colleagues, contact via institutional or domain-specific mailing lists, internet forums or web-based social networks.

3.11 Large Language Models

Large language models (LLMs) are a class of artificial intelligence systems that have been trained on vast amounts of text data to understand and generate human-like language. These models can be used for a wide range of tasks, including natural language processing, text generation, and

[49] https://openresearchsoftware.metajnl.com/
[50] https://joss.theoj.org/
[51] https://www.ipol.im/
[52] https://www.sciencedirect.com/journal/softwarex

machine translation. LLMs can be also applied for searching research software by identifying relevant software tools and providing information about their features and capabilities.

4 Analysis

For the comprehensive analysis of ways to discover research software, which is currently a `work-in-progress`, we derived a set of analysis categories (content, search, organisation, technology, item evaluation) and a set of criteria for each (table 3). We derived categories and criteria after joint discussions among the authors, taking into account the persona and its needs as well as the discovery scenarios. We pragmatically reduced these to the criteria that were most important and meaningful to us in order to keep the analysis clear. We aim to provide a comprehensive and systematic analysis in the final paper (see section 7).

We have initially selected only one example per category. We used the general relevance and popularity of the pathway as well as personal knowledge of the pathway as selection criteria. The data basis for the analysis can be found in [SWG25]. After initial considerations, we decided to omit some of the pathway categories altogether, as these were difficult to analyze using our chosen criteria scheme. We did not analyze the social networks, as most of the analysis criteria were not applicable here. In the case of the large language models, all criteria are present in theory and in the best or greatest possible form, which is why we did not hope to gain any further insights from the analysis. We did not include the software journals in this publication due to their marginal importance for the general case and would then consider them in the next round of analysis.

Table 5 shows examples of the analysis for a code repository (GitHub), a programming language centered software repository (PyPi) and an institutional software catalog (Helmholtz software directory).

5 Discussion

With the categorization of discovery pathways for the search for research software and the subsequent analysis, we have shown that there are a very large number of options for searching for research software. In the following, we will discuss findings from the analysis in more detail.

5.1 No place to find them all

From the sheer number of different and equally valuable options for searching for research software, we conclude in analogy to [DHB+24] that there is currently not yet one universal option. Rather, the context of the search plays a major role in the choice of discovery pathway. If I work as a programmer in a specific programming language, I will start a search in a programming language centered software repository wherever possible. The same applies to the use of the software in a specific operating system, which is why I might start the search in an operating system centered software repository. Curated lists, on the other hand, give me a good initial overview of a particular topic without me wanting to start a specific software search. Talking

Table 5: Analysis excerpt for the first three discovery pathways we analysed. Full analysis is available in [SWG25]. (The analysis was conducted in February 2025. For this publication we last updated content quantity and community size on Sept 02 2025.)

Category	Criteria	Github	PyPi	helmholtz.software
Content	Quantity	240,006,401	609,189	413
	Source ID	repository ID	none	none
	Scientific domain	all	all	all
	Pub. type	all (focus: software)	software	software
	Metadata Schema	custom	custom	custom
Search	Filter options	advanced; customizable	800 nested filters; checkbox	keywords, languages, licenses; checkbox
	Search Function	basic, advanced, full-text	index, full text	basic, full-text
	Science Information	no (manually possible)	no	yes (mentions, reference papers)
	Recommender	no	no	no
Organisation	Host institution	private company	foundation (Python Software Foundation)	research association
	Funding	licenses	sponsors	institution (Helmholtz)
	Documentation	Pathway and search functionality	Pathway and search functionality	Pathway and search functionality
	Ease of use	powerful but not intuitive, advanced gui search not easy to access	very intuitive	easy, very intuitive
	User Engagement	Full interaction with content, depending on user roles	add content, no interaction with others content	add content, no interaction with others content, restricted to Helmholtz
	User Governance	de-jure - non, de-facto - community 'influenced'	Open Source Software, Community Discussions	Open Source Software, none
	Community Size	269.550.403	903,230	n/a
Technology	APIs	REST, GraphQL	several APIs, BigQuery Datasets, RSS Feeds	REST, OAI-PMH
	Platforms integrated	Many (Zenodo, ...?)	GitHub,	OpenAlex, Package Managers
Item Evaluation	License	yes	yes	yes
	Usage Statistics	yes (git clones, visitors, stars, watches)	yes	yes (stars, forks)
	Dev Statistics	yes (forks, git clones, visitors, stars, watches)	yes (stars, forks)	yes (stars, forks)
	Dependencies	yes	yes	no
	Quality	custom (code cov, test, pipeline status, linters)	custom (code cov, test, pipeline status, linters)	no
	Operating system	no (only indirect in e.g. readme)	no - n/a	no
	Functional Category	no	yes	yes
	Description	yes (readme)	yes	yes

to trusted colleagues (social network) who are considered to have expertise in a certain area can provide dialog-oriented information on specific software.

This very heterogeneous landscape makes it almost necessary to have prior knowledge of the available discovery pathways in advance of the search in order to be able to search in a targeted manner. In addition, the search agent must be aware that the context of a search query determines the discovery pathway. Search agents must familiarize themselves with the application scenario of the software in advance.

5.2 No categories to sort them all

The discovery options we investigated use different metadata or metadata schemas with different levels of coverage. As a result, the filter options that can be applied to the search results also differ.

The mixed content publication platform Zenodo, which we consider to be very popular for the publication of research software, offers surprisingly few filter options. Here, users of the browser frontend have very few faceting options, although a great deal of metadata on software items is recorded. The code repository provider GitHub, another very popular discovery pathway, also offers advanced search options, but here too it is not possible to filter by functionality, for example, as this is not adequately recorded in the metadata of individual software projects.

In our opinion, the search for research software would benefit greatly if extensive metadata on software entries were recorded and these were also used to filter the search results. It would also significantly improve the search experience if the functionality of research software could be better captured through metadata and expressed in filtering options.

5.3 Software seeking vs software retrieval

In the field of information science, a distinction is made between seeking information and retrieving information, which was also significant for our analysis and the direction of our work. Information seeking is "*[h]uman information behavior dealing with searching or seeking information by means of information sources and (interactive) information retrieval systems*" [IJ05] while information retrieval is "*the purposeful searching for information in a system, of whatever kind, in which information - whether in the form of documents, or their surrogates, or factual material ('information itself'), are stored and represented*" [SW97].

Finding software is the result of either an *active* process of searching – consulting resources with the intention of retrieval – or the rather *passive* discovery, for example, via citations/reference lists or serendipitous occasions during communication in a social network. The latter, more passive discovery has not been described further here as we investigated active discovery pathways.

However, we are aware that a considerable portion of research software is found through a less targeted route, almost without being actively sought. There is still enormous potential for providers of discovery pathways or for individual software projects to improve the discoverability of research software, for example by using semantic linking via citations or by helping chance along, whatever that may look like.

6 Limitations

At this stage of our work (work-in-progress), this paper has some significant limitations.

The list of criteria for the analysis is currently still a reduced version and will be finalized at a later date. In addition, the criteria were selected on the basis of joint discussions. This does not necessarily constitute a limitation, but it is not a systematic approach.

Only a limited set of discovery pathways has been analyzed so far. Therefore, it is not yet possible for us to make comparisons within a pathway category to answer questions such as which domain-specific catalog offers the best faceting options.

The personas and discovery scenarios identified in the first sections have not been sufficiently considered in the discussion so far. However, returning the results to this starting point offers a great deal of analytical potential for answering questions about the preferences of the individual personas with regard to the discovery pathways to be selected or the suitability of the scenarios for specific pathways.

7 Future research

While we started working on the paper, we were simultaneously working on a research proposal on the same topic. The paper can be seen as conceptual groundwork for the proposal. The research proposal has since been approved[53]

The project `find.software` started in January 2025. As part of the find.software project, we will continue our work on this paper and address more aspects than this work-in-progress paper could do. This primarily concerns a systematic analysis of the discovery pathways, a critical and, above all, comprehensive discussion, and the inclusion and knowledge gain from the personas and discovery scenarios developed in the first sections.

8 Call for open infrastructures

The aspect of the provision of discovery pathways only played a marginal role in our analysis. However, looking at the current research landscape, especially in the US, it is important to talk about the hosting and governance of research infrastructures in general and discovery infrastructures in particular.

We believe that research infrastructures belong in the hands of a global and open community. Science should be as open as possible. This applies to publications, infrastructures and communication. A community can ensure that all interests can be taken into account in a global framework. The interests of private and profit-oriented companies should not play a role in science. If we do science openly, then knowledge can develop and spread very quickly because it can be shared more easily and researchers can collaborate more easily when they have all the knowledge available to them. Open infrastructures also protect against malicious interference in the code.

[53] DFG project number: 567156310, https://gepris.dfg.de/gepris/projekt/567156310?language=en; Project proposal as RIO publication: https://doi.org/10.3897/rio.11.e179253; Project website: https://www.find-software.org

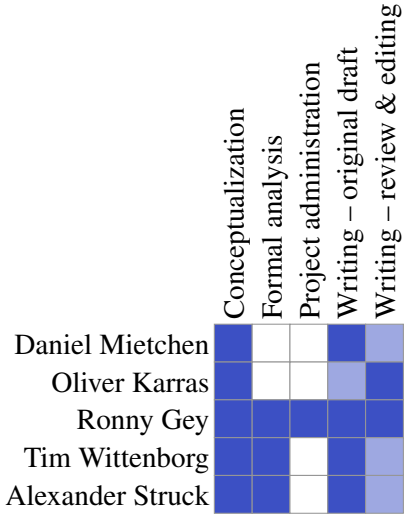
We therefore call on the scientific community to focus on open and community-oriented infrastructures to minimize commercial and political influence for the benefit of the free exchange of knowledge.

9 Conclusion

In our paper, we set out to clarify the information discovery process for research software. We first described five personas that act as research software discoverers. Further, we developed 12 scenarios for searching research software. We then categorized various discovery pathways for research software and analyzed them according to relevant search criteria. Our results show a very heterogeneous landscape of discovery pathways and a lack of metadata and filtering options. We also ask the research community to place critical research infrastructure in the hands of the global research community. This is intended to prevent a potential threat to the open infrastructure by national or private interests.

Author contributions

We use the Contributor Role Taxonomy (CRediT) [BAA+15] to show the contributions of each author.



References

[AP10] T. Adlin, J. Pruitt. *The Essential Persona Lifecycle: Your Guide to Building and Using Personas*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.

[BAA+15] A. Brand, L. Allen, M. Altman, M. Hlava, J. Scott. Beyond Authorship: Attribution, Contribution, Collaboration, and Credit. *Learned Publishing* 28(2):151–155, Apr. 2015.

doi:10.1087/20150211
https://onlinelibrary.wiley.com/doi/10.1087/20150211

[CKB+21] 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. FAIR Principles for Research Software (FAIR4RS Principles). 2021.
doi:10.15497/RDA00068

[Con22] Confederation of Open Access Repositories. COAR Community Framework for Good Practices in Repositories, Version 2. Technical report, Zenodo, July 2022.
doi:10.5281/ZENODO.7108100
https://zenodo.org/doi/10.5281/zenodo.7108100

[CY14] F. W. Chickering, S. Q. Yang. Evaluation and Comparison of Discovery Tools: An Update. *Information Technology and Libraries* 33(2):5–30, June 2014.
doi:10.6017/ital.v33i2.3471
https://ital.corejournals.org/index.php/ital/article/view/3471

[DHB+24] S. Druskat, N. P. C. Hong, S. Buzzard, O. Konovalov, P. Kornek. Don't Mention It: An Approach to Assess Challenges to Using Software Mentions for Citation and Discoverability Research. Feb. 2024.
doi:10.48550/ARXIV.2402.14602

[DC20] R. Di Cosmo. Archiving and Referencing Source Code with Software Heritage. *Mathematical Software – ICMS 2020*, pp. 362–373–362–373, 2020.

[Gey20] R. Gey. *Seeking Research Software. A Qualitative Study of Humanities Scholars' Information Practices*. https://doi.org/10.5281/ZENODO.4320954, Institute for Library and Information Science, HU Berlin, 2020.

[HG16] M. Hucka, M. J. Graham. Software Search Is Not a Science, Even among Scientists: A Survey of How Scientists and Engineers Find Software. *Journal of Systems and Software, 141:171-191, 2018*, May 2016.
doi:10/ggc6j4

[IJ05] P. Ingwersen, K. Järvelin. *The Turn*. Springer-Verlag, 2005.
doi:10.1007/1-4020-3851-8

[JVS17] A. Jansen, M. Van Mechelen, K. Slegers. Personas and Behavioral Theories: A Case Study Using Self-Determination Theory to Construct Overweight Personas. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Pp. 2127–2136. ACM, Denver Colorado USA, May 2017.
doi:10.1145/3025453.3026003
https://dl.acm.org/doi/10.1145/3025453.3026003

[MOd15] M. Mesgari, C. Okoli, A. O. de Guinea. Affordance-Based User Personas : A Mixed-Method Approach to Persona Development. In *Americas Conference on Information Systems*. 2015.

[PG03] J. Pruitt, J. Grudin. Personas: Practice and Theory. In *Proceedings of the 2003 Conference on Designing for User Experiences*. Pp. 1–15. ACM, San Francisco California, June 2003.
doi:10.1145/997078.997089
https://dl.acm.org/doi/10.1145/997078.997089

[SRW24] V. Seibert, A. Rausch, S. Wittek. Betty's (Re) Search Engine: A Client-Based Search Engine for Research Software Stored in Repositories. *ing. grid* 1(2), 2024.

[SW97] K. Sparck Jones, P. Willett. *Overall Introduction*. Pp. 1–7. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[SWG25] A. Struck, T. Wittenborg, R. Gey. Review of discovery pathways for research software discovery. Mar 2025.
doi:10.5281/ZENODO.15430737