

# FAIRSECO: An infrastructure for measuring impact of research software

Slinger Jansen<sup>1,2</sup>, Elena Baninemeh<sup>1</sup> and Siamak Farshidi<sup>1</sup>

<sup>1</sup>*Utrecht University, Utrecht, The Netherlands*

<sup>2</sup>*Lappeenranta University, Lappeenranta, Finland*

## Abstract

Finding research software is a complex task, as research software engineers regularly search for algorithms and methods deeply embedded in large software packages during the creation of research software. Furthermore, they go through lengthy evaluation and extraction processes to find one particular algorithm relevant to their research project. Additionally, for creators of research software, it is hard to show the impact that their code has made on the field, and only very coarse measures exist for evaluating the success of research software. This study introduces the concept of FAIRSECO, which aims to enable research software engineers to rapidly find and extract relevant software fragments from the worldwide research software ecosystem. Research software engineers from all fields can transplant these source code fragments, maintain the provenance of source code, and easily credit the original authors of the software. Simultaneously, the platform also enables research software engineers to report on their software's impact. With FAIRSECO, we introduce a platform for research software engineers that creates a "method economy", i.e., where smaller granularity reuse becomes possible while increasing FAIRness (Findable, Accessible, Interoperable, and Reusable) of the worldwide research software ecosystem.

## Keywords

FAIR Software, research software, software engineering, software repository

## 1. Introduction

Research Software Engineers (RSEs) have many ways to search for source code in the Worldwide Research Software Ecosystem (WRSE) [1]. Unfortunately, these search methods suffer from weaknesses, hampering scientific progress. One of the problems is granularity: it is possible to search through code on a file-level and cover a significant part of the WRSE or search for a line of code but only cover a small part of the WRSE [2], but not both.

RSEs reuse research software to perform extensive searches to find relevant software components for their problem [3]. They are typically looking for a small set of features in a more significant software component. Secondly, they must painstakingly extract the identified source code and adapt it to their projects [4].

This study introduces the concept of FAIRSECO, which aims to enable RSEs to rapidly find and extract relevant software fragments from the Worldwide Research Software Ecosystem (WRSE).

---

*BENEVOL'22: The 21st Belgium-Netherlands Software Evolution Workshop, Mons, 12-13 September 2022*


✉ [slinger.jansen@uu.nl](mailto:slinger.jansen@uu.nl) (S. Jansen); [e.baninemeh@uu.nl](mailto:e.baninemeh@uu.nl) (E. Baninemeh); [s.farshidi@uu.nl](mailto:s.farshidi@uu.nl) (S. Farshidi)

🌐 <https://www.slingerjansen.nl/> (S. Jansen)

🆔 0000-0003-3752-2868 (S. Jansen); 0000-0003-3270-4398 (S. Farshidi)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

RSEs from all fields can transplant these source code fragments, maintain the provenance of source code, and easily credit the original authors of the software. Simultaneously, the platform also enables RSEs to report on their software's impact [5].

FAIRSECO enables RSEs to reuse source code fragments, maintain the provenance of their source code, and easily credit the original authors of the software. Simultaneously, the platform also empowers RSEs to report on their software's impact. These two affordances are a radical and necessary innovation that make the WRSE more FAIR<sup>1</sup>. We propose a new approach to extract software and software meta-data *at the level of software methods*, to enable RSEs to identify code relevant to them, easily extract it from an open-source package, and reuse it on their own. With a relatively small investment, we unearth the relationships between code fragments, code files, and their projects on a worldwide scale for research software, enabling a more robust method of measuring software impact [6]. We can map a significant part of the WRSE with FAIRSECO.

Research software is defined as “software that is used to generate, process or analyse results that you intend to appear in a publication (either in a journal, conference paper, monograph, book or thesis)” [7]. Software is insufficiently seen as an output of the research process. This is surprising, as approximately half the published articles in Nature, mention the word “software” in their title, abstract, or introduction. It is essential that the software engineering community takes a stronger stance on this and actively starts promoting research software as one of the many fruits from our practice. Measuring the impact of research software is harder than, for instance, traditional citation score analysis [8]. With FAIRSECO we aim to make transparent what research software has made the largest impact, is trending, and should be invested in [9].

We come to the following research questions (RQs):

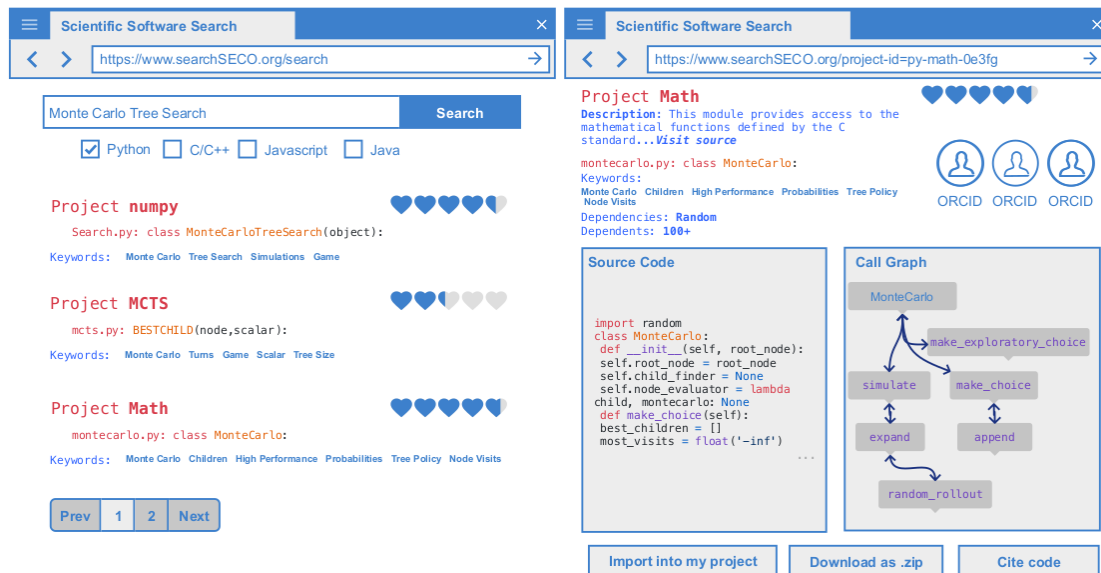
- MRQ: How can research software be made FAIR at the method level?
- RQ1: How can method reuse be identified in the worldwide research software ecosystem?
- RQ2: How can research software impact be measured and made transparent?
- RQ3: How can reusable research software be found at the method level?
- RQ4: How can research software be extracted from existing software packages at the method level?
- RQ5: How can research software engineers be stimulated to report reuse in the software engineering process?

In the next Section we explain how FAIRSECO will be built and what technological challenges are foreseen. In Section 3 is explained how FAIRSECO will in fact contribute to more FAIRNESS, for instance by providing screen shots of what the system interface will look like. In Section 4 we explain how we plan to proceed with this project and what we see as our main challenges in the research work, such as the evaluation of the research software portals that we create. Finally, in Section 5 we present the radical idea of being able to get a PhD on written software alone and subsequently, nuance it.

---

<sup>1</sup><https://fair-software.eu/>

**Figure 1:** FAIRSECO has its main goal to support RSEs in finding code fragments that can support their goals. In this example, the RSE looks for “Monte Carlo Tree Search” and finds a code sample in the Project Math. We can now extract this algorithm relatively easily by exporting the entire class into our project. The call graph can be used to extract only the required piece of code.



## 2. FAIRSECO Architecture

FAIRSECO is dependent on SearchSECO [10], which was presented at BENEVOLE’20. SearchSECO is a large database of methods of the top rated projects (with “stars”) on Github. SearchSECO clones a git project, extracts a number of versions, and extracts the files and authors from those versions. The files are subsequently processed and each method that is encountered that is longer than 5 lines is stored. The method’s abstract syntax tree is extracted and a representation of this abstract syntax tree is hashed. SearchSECO currently parses Java, Javascript, C/C++, and Python. SearchSECO is itself a project on Github and can be found via: <https://github.com/SecureSECO/SearchSECOController>. Furthermore, the database can be accessed through a portal: <https://secureseco.science.uu.nl/portal/>. In this portal (please note this is not the FAIRSECO search portal) visitors can enter their own project link and email address. After the project has been processed and matched, the visitor receives a report of the matches in the SearchSECO database and can determine if there are any potentially vulnerable fragments in their project. Currently (June 14th 2022), the database contains 19 million unique methods from approximately 100 thousand projects from Github. From hereon we refer to the whole infrastructure of SearchSECO as FAIRSECO, as FAIRSECO builds on top of the SearchSECO infrastructure. Please note that SearchSECO is directed at any kind of software on GitHub, while FAIRSECO only targets research software. We are currently developing methods for identifying research software, using indexes and qualifiers used by <https://fair-software.nl>.

A mock-up of the FAIRSECO platform is shown in Figure 1. The figure shows two pages of

the platform: the main search portal and a method that has been identified as potentially useful. In the long run, we can also imagine a rich text index that categorizes methods and functions that are easy to extract, for instance, categorizing methods along with characteristics such as categories (machine learning, graph mining, etc.) and technologies (graph databases, microscope software, etc.). Furthermore, we plan to develop an integrated development environment that automatically suggests code for an existing project. FAIRSECO experimentally prioritizes those annotated methods that best match the search of the RSEs, and that are relatively easy to extract. It is also essential that well-structured and written software is ranked higher than poorly written software. Some methods will be visited often and thereby might also be more successful in providing the RSE with what they need. The result page, shown on the right side of Figure 1, shows a fragment of the code that is to be extracted, its call graph, and an extensive description of the code to help the RSE in determining whether this code is relevant. As the code is extracted, we also add a comment tag to the code that references back to the original code, including license information to promote FAIR software reuse and impact measurement [11].

### 3. A More FAIR Worldwide Research Software Ecosystem

FAIRSECO provides mechanisms for making the WRSE more FAIR [7].

- **Findable** - FAIRSECO has two mechanisms that make code in the WRSE more findable. First, we are the first to use code hashing for a worldwide scope of software, thereby creating a highly performant and scalable database of WRSE source code. Secondly, because each method is annotated with its project's meta-data, we enable RSEs to rapidly find the methods they need. In the future we envision enriching this data further.
- **Accessible** - FAIRSECO provides access to the innards of many WRSE projects. The main goal of FAIRSECO is to save the RSE from having to go through the time-consuming process of locating algorithms in source code, only to conclude that it does not fit the RSE's needs.
- **Interoperable** - The main goal of the project is to create a 'method economy' instead of a package economy, where RSEs can go to the method level immediately in reusing code. Code should be easily extractable from existing projects, for instance, by enabling automated encapsulation of components.
- **Reusable** - Once the code has been identified and extracted, it should be easy to reuse in the new code base without violating licenses, and accidentally copying vulnerabilities (because we store this data in SearchSECO). FAIRSECO maintains meta-data about projects to check for license compatibilities. Furthermore, as full project histories are maintained, it becomes possible to support co-evolution across projects.

This section describes several use cases to illustrate how RSEs can use FAIRSECO to make the WRSE more FAIR.

First, one should envision the FAIRSECO code index as a large database of worldwide software fragments, against which any code can be compared. When an RSE takes their code and compares it to the FAIRSECO database, they can gather data on the following: (1) They can look for code clones within their project and use its call graph to establish the project's code quality,

**Figure 2:** This screenshot from the Research-software.nl portal has been extended (in red) with the data that we have available in the SearchSECO DB. The eScience Center owns the Research-Software.nl portal and is willing to extend their software in this way, as it shows dependencies, but also uncited reuse (i.e., invisible impact) of the software package.

Package Name	Date	Methods used	Methods cloned
GPU Kernels for Bhattacharya distance between two point sets	January 30, 2017	5	0
PRNU GPU Desktop Application	November 16, 2017	0	9
KM3Net - Real-time detection of neutrinos using GPUs	November 30, 2016	8	1
GPU Kernels for GaussTransform	January 30, 2017	0	All

Packages citing this package:	4
Packages reusing this package:	154
Packages depending on this package:	5

complexity, and cloning. If matching code clones are found outside of the project, several things become possible. They can perform license checks that establish whether their license conforms to the licenses from the other projects from which they reuse code. (2) They can see whether that code has already evolved within those projects, and they can choose to apply the same patches to their code. Also, they can see whether the code that matches was cloned from their project, whereby they can establish the impact of their code in the WRSE. (3) They can identify whether their code is matched with existing vulnerable code and decide whether they wish to fix these vulnerabilities in their code.

Secondly, the RSE can look for existing code in the FAIRSECO code index to reuse it. They could, for instance, look for “Pareto analysis” in the FAIRSECO code index. If they find appropriate matches, it becomes possible to *lift and transplant* [12] that code into their project. The technique for transplanting this code is still under development, but we use the call graph to decide how to get a minimum code slice to transplant.

Finally, the RSE can use the code index to determine whether their project has been reused in the W(R)SE. Furthermore, they can use the matches with their code to create a “bill of materials”, that can support them to create citations to existing source code.

One of the partners in our project is the eScience Center, who are behind the Research-Software.nl portal. They would like to extend the portal with impact measurement features, such as visualized in figure 2. The most interesting metrics from an impact perspective are found at the bottom of the screenshot, where we indicate how many packages are reusing the research software, how many packages depend on the software, and how often the software is cited.

## 4. Current Status

Currently, SearchSECO is storing little meta-data about projects. One of the extensions needed is the FAIRSECO meta-data component, which stores particular meta-data about a project, such as the read.me file, the authors of the method, and the dependencies on other methods in other projects. These meta-data are required to identify which projects contain usable methods, which projects depend on which others, etc.

A dependency extraction mechanism is needed, such as FASTEN [13], World-of-Code, or the data in libraries.io to extract the dependencies. We propose that we do not build these ourselves, but use common infrastructures such as the ones mentioned above, or extend them. Also, we need a parsing and extraction architecture that enables the rapid extraction of methods from code files, including the call graph in a project.

The search engine front end needs to be built. While it will be relatively easy to create a mock-up inspired by Figure 1, it will be hard to actually create the working parts. For an MVP, we are most interested at this point in a simple search function and an extraction function.

The easiest to create in this software project is probably the reporting service for research software projects. Basically, we need to take the output from SearchSECO and process it to generate a report about the scientific project reuse. We want to know which projects are dependent on this project, which of those are scientific, and which projects have copied code from this project or are copied into this project. The citation.cff file in Github can be used as a marker to identify whether a project is academic or not. As this project is starting later in the year, this project may be already finished by then.

When the artifacts envisioned in this project are created, we need success criteria to define and measure our own impact. Ideally, our software will be integrated with the Research-Software.nl portal, but we also need to evaluate artifacts such as the search engine in our work. We plan to do this by evaluating the artifacts with RSEs in an incremental process, where new features are added after each round of evaluation.

## 5. Discussion and Conclusion

This study presents the concept of FAIRSECO, a search engine and research platform that searches through abstract representations of research software. FAIRSECO may become the entry point into the WRSE, where all RSEs go to find and extract source code for their complex compositions of projects. Furthermore, FAIRSECO aims to provide a significant contribution to the WRSE, by making software available that makes research software impact analysis fast, easy, and accessible.

A current trend in research evaluation is that we are increasingly asked to look at other things than citation analysis. A contributing member of the research community does not only write papers and articles, but also writes software. We are not yet convinced that we can one day provide RSEs with a PhD solely on their software, but we do believe that FAIRSECO can help RSEs better show the impact of their work.

We aim to further extend FAIRSECO by adding the following features. First, we have not implemented attribution into the platform yet, i.e., projects are stored from Github with their

descriptions without any extra annotations about them being research software. Secondly, we do not have any citation generation capabilities, including a scientific bill of materials with a new release of research software. Thirdly, we are looking for manners to 'lift' software fragments out of their code base for effective code transplantation. Fourthly, we wish to support our tooling with a licensing compatibility checker to ensure that the research software follows intellectual property rules and policies, also because we currently cannot handle custom licenses and only take licensing data from GitHub. Six, we want to add publications as meta-data to software projects, for easier reference. Finally, we intend to soon integrate other sources of software such as GitLab.

## References

- [1] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, G. Wilson, How do scientists develop and use scientific software?, in: 2009 ICSE workshop on software engineering for computational science and engineering, Ieee, 2009, pp. 1–8.
- [2] S. Jansen, M. A. Cusumano, S. Brinkkemper, *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*, Edward Elgar Publishing, 2013.
- [3] J. Howison, E. Deelman, M. J. McLennan, R. Ferreira da Silva, J. D. Herbsleb, Understanding the scientific software ecosystem and its impact: Current and future measures, *Research Evaluation* 24 (2015) 454–470.
- [4] 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 (2018) 171–191.
- [5] W. Hasselbring, L. Carr, S. Hettrick, H. Packer, T. Tiropanis, From fair research data toward fair and open research software, *it-Information Technology* 62 (2020) 39–47.
- [6] G. Gousios, E. Kalliamvakou, D. Spinellis, Measuring developer contribution from software repository data, in: *Proceedings of the 2008 international working conference on Mining software repositories*, 2008, pp. 129–132.
- [7] A.-L. Lamprecht, L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. Martin Del Pico, V. Dominguez Del Angel, S. Van De Sandt, J. Ison, P. A. Martinez, et al., Towards fair principles for research software, *Data Science* 3 (2020) 37–59.
- [8] K. Li, P.-Y. Chen, E. Yan, Challenges of measuring software impact through citations: An examination of the lme4 r package, *Journal of Informetrics* 13 (2019) 449–461.
- [9] S. Jansen, Measuring the health of open source software ecosystems: Beyond the scope of project health, *Information and Software Technology* 56 (2014) 1508–1519.
- [10] S. Jansen, S. Farshidi, G. Gousios, T. van der Storm, J. Visser, M. Bruntink, Searchseco: A worldwide index of the open source software ecosystem, *Proceedings of BENEVOL 2020* (????).
- [11] F. Hou, S. Farshidi, S. Jansen, Trustseco: A distributed infrastructure for providing trust in the software ecosystem, in: *Proceedings of the International Workshop on Blockchain for Information Systems*, 2021.
- [12] J. Petke, M. Harman, W. B. Langdon, W. Weimer, Specialising software for different downstream applications using genetic improvement and code transplantation, *IEEE Transactions on Software Engineering* 44 (2017) 574–594.
- [13] P. Boldi, G. Gousios, Fine-grained network analysis for modern software ecosystems, *ACM Transactions on Internet Technology (TOIT)* 21 (2020) 1–14.