Decoding Research Software Impact

Deekshitha[1,2,3,*], Rena Bakhshi[1], Rob van Nieuwpoort[3] and Slinger Jansen[2]

[1]*Netherlands eScience center, the Netherlands*
[2]*Utrecht University, Utrecht*
[3]*University of Leiden*

Abstract

Research software is an integral part of the research. It is developed through the effort and collaboration of researchers, Research software engineers, and other stakeholders. Despite its importance in the research, there is currently no sufficient method for measuring the impact of research software. Assessing research software impact is essential for advancing the careers of those involved in research software development and ensuring recognition of their contributions. This study aims to build a research software impact model for the different stakeholders of the research software by understanding their goals in measuring its impact. By providing a model for measuring research software impact, this study will contribute to building and strengthening the research community around research software, enhance its visibility and impact, promote the development of mature research software, and attract industry partnerships.

Keywords

Research software, Research software impact, Maturity model

1. Problem Definition

Research software includes source code files, algorithms, scripts, computational workflows, and executables created during the research process or for research purposes. Software components (e.g., operating systems, libraries, dependencies, packages, scripts) used for research but not created during or with a clear research intention should be considered software in research, not research software [1]. Different studies show that a significant amount of research generates new code, and researchers recognize that code is an integral part of research [2, 3].

The development of research software involves researchers and a diverse group of stakeholders, each playing a specific role. These stakeholders include funders who support the research financially, research software engineers (RSEs) managers who design, build, and maintain the software, and other contributors such as documentation writers, testers, and community managers. Despite the growing importance of research software, the current academic reward system prioritizes publications, neglecting the contributions of researchers and RSEs who spend their time and effort developing research software [4]. This lack of recognition creates challenges for career advancement and leads to the poor sustainability of research software [5].

The Research Software Engineering movement has emerged to address this gap by promoting the importance of research software and advocating for the role of people, policies, and infrastructure in its development, support, and maintenance [6]. Researchers, RSEs, and funding organizations are increasingly interested in establishing best practices to ensure that research software becomes sustainable, reproducible, and community-driven [7], RSMD guidelines [8], and the OpenSSF badge program[9]—offer best practices that enhance visibility, impact, and long-term sustainability. However, measuring the impact of research software remains a significant challenge. Stakeholders have diverse goals and priorities, making it difficult to capture the research software impact with the existing

The 15th International Conference on Software Business (ICSOB 2024), NOVEMBER 18-20, 2024, Utrecht, the Netherlands
*Corresponding author.
d.deekshitha@uu.nl (Deekshitha); r.bakhshi@esciencecenter.nl (R. Bakhshi); r.v.van.nieuwpoort@liacs.leidenuniv.nl (R. v. Nieuwpoort); slinger.jansen@uu.nl (S. Jansen)
0000-0002-0877-7063 (Deekshitha); 0000-0002-2932-3028 (R. Bakhshi); 0000-0002-2947-9444 (R. v. Nieuwpoort); 0000-0003-3752-2868 (S. Jansen)

approaches [10]. Without a proper framework, the contributions of those involved in research software development often go unrecognized, hindering both personal and professional advancement and the sustainability of the research software.

2. Knowledge Gap

This study aims to address three knowledge gaps identified in the existing literature, as listed below:

- **No foundational method for measuring research software impact:** Several tools are available to measure some aspects of the impact of research software in terms of features such as code quality, citation, and software dependency. One example is the largest open-source platform, GitHub, which provides various metrics like forks, watchers, stars, and user counts, which offer some insights into software usage. However, these metrics do not capture the full impact of research software and the expertise of its developers. Current measures do not fully account for factors such as the qualitative contributions of software, its role in advancing research, or the broader context of its use [11].
  Although a limited number of studies have begun to address this issue by considering metrics such as citations, stars, contributor numbers, code reuse, FAIRness (Findability, Accessibility, Interoperability, and Reusability), and software quality, practical evaluation of the impact of research software requires a more understanding and the ability to make judgments about these various factors.
  Furthermore, existing tools designed to assess the impact of research software, including Research Software Directory (RSD) [12], Software Heritage Graph (swh-graph) [13], Depsy [14], Libraries.io, howfairis [15], Tortellini [16], SearchSECO [17], SQAaaS [18], Sigrid, and GrimoireLab offer only limited insights. These tools often address only specific impact aspects or provide fragmented data, highlighting the need for a more unified and detailed framework to evaluate research software. Thus, all the above tools are limited to some criteria and need to fully cover the requirements to recognize the efforts of the people behind the development.
- **Lack of a foundational method for measuring the maturity of research software projects:** The second knowledge gap is not a sufficient method for assessing the maturity of research software projects since research software projects differ in many criteria, including importance to citation, sustainability, reproducibility, and much more [19].
- **Lack of automated mechanisms for classifying research software:** The third knowledge gap is the Lack of automated methods to distinguish research software from other types of software and classify it by domain. Research software is distributed across various platforms such as GitHub and GitLab, but there is no current solution to automatically identify it or categorize it based on its specific domain [20].

This study aims to fill these gaps by proposing solutions, including developing research software impact models, maturity assessments, and automated research software classification techniques. Through these contributions, it seeks to provide a framework or methods for evaluating, managing, and advancing research software ecosystems.

3. Objectives

To address the knowledge gap mentioned in the previous section, this study establishes the following objectives:

The primary objective of this research is to develop a Research Software Impact Model that meets the needs of all stakeholders within the research software ecosystem. This model will serve as a framework to capture the value of research software and recognize the contributions of all stakeholders, including researchers, funders, developers, and managers.

In addition to measuring impact, this study also explores related areas to improve research software management:

- **Identifying factors that influence research software impact:** Several factors affect the impact of research software, including code quality, citations, and software dependencies. Additionally, stakeholders such as developers, users, and funders have varying goals and perspectives regarding research software impact [21]. The objective is to find the factors that influence the impact of the research software.
- **Maturity model for research software:** The objective is to identify what defines mature research software and improve the management processes of research software projects. The study assesses the maturity of research software projects by evaluating existing practices and identifying best practices for sustainable software development.
- **Automatizing classification of research software:** This study focuses on developing a model to automatically categorize research software based on different factors such as domain, maturity level, and other relevant characteristics.

By developing a structured framework for measuring research software impact, this research aims to bridge the gap between research software development and academic recognition. Ultimately, the proposed model will ensure that research software is sustainable, impactful, and supported by a research community, fostering stronger connections between academia and industry.

4. Research questions

The main research question is
RQ: How to decode the impact of research software?

We divided the above main question into sub-questions that are listed below:

SQ1: How can AI support accessing the impact of research software? To address this, we started by exploring current models or tools used to measure the impact of research software. Then, considering code quality, code reuse, and code citation, we developed Github action to measure the impact of the research software. These impact scores are used to tell how impactful the software is. Brief information about FAIRSECO is provided in 5.1. Additionally, during the National Research Software Day, we conducted a workshop. We discussed various goals for measuring the impact of research software from the perspectives of researchers, research software engineers, funders, and policymakers. We also identified relevant metrics for impact measurement [21].

In the coming months, we plan to conduct focus group discussions with Research Software Producers, Research Software Users, Funders, Institutional Research Software Ecosystem Enablers, and Non-Institutional Research Software Ecosystem Enablers. These discussions will further explore research software impact and help develop an impact model tailored to the needs of research software stakeholders. Additionally, it aims to use machine learning models to measure the impact of research software.

SQ2: How does assessing the maturity of the research software help to evaluate the research software project management process? The second chapter of the thesis (research study) introduces a framework for evaluating research software project management, addressing research question **SQ2**. The framework, Research Software focus area Maturity Model (RSMM) includes 79 best practices and 19 capabilities, grouped into four focus areas. This method employs a systematic literature review to collect best practices in research software project management. Additionally, semi-structured expert interviews are conducted to determine the positioning of these practices within a maturity matrix. A case study is also conducted to evaluate the model RSMM. A brief description of the RSMM is provided in 5.2.

SQ3: How to automatically categorize research software? Research software can be categorized into many categories, such as role-based, developer-based and maturity [20]. The third chapter will discuss the automatic classification of research software using machine learning and NLP techniques.

The next section describes the research method used to address these research questions.

5. Research Method

The research questions are solved using the below methods.

5.1. FAIRSECO: Framework for impact measurement

FAIRSECO is designed to help researchers and research software engineers access the impact of their research software over time. It considers the following features:

- **FAIR Assessment for Research Software**: FAIRSECO evaluates the FAIRness of research software based on five recommendations [22]. It examines whether the research software utilizes a public repository with version control, contains a license and citation file, is listed in a community registry, and follows a quality checklist. Repositories meeting all five criteria receive a perfect FAIRness score of 5 out of 5.
- **Quality Scoring**: The quality of the research software is determined through four criteria: FAIRness, license compliance, maintainability, and documentation score. FAIRSECO checks for license compliance and calculates a score accordingly. Maintainability is evaluated by assessing the percentage of closed issues in the repository, and the documentation score is determined by verifying the presence of documentation and readme files. These aspects collectively contribute to the overall quality assessment.
- **Code Reusability**: FAIRSECO measures the code reuse by using the tool SearchSECO, which looks in its database for occurrences of the method of the research software to find the code reusability score.
- **Citation Analysis**: FAIRSECO identifies the citation score of the software by using OpenAlex and Semantic Scholar databases.
- **Software Bill of Materials (SBOM)**: FAIRSECO generates a Software Bill of Materials for the research software, which includes information about its software dependencies.

Quality score, code reuse and citation are used to calculate impact score. The GitHub action, FIRSECO, is available to assess the impact of any GitHub repository. Detailed information about FAIRSECO is provided in this paper [23]. In future work, we will validate the factors considered for measuring impact with various stakeholders of research software.

5.2. RSMM- Focus area maturity model for research software projects

The maturity model is *"a structured collection of elements that describe the characteristics of effective processes at different stages of development. It also suggests demarcation points between stages and methods of transitioning from one stage to another"* [24]. The maturity models are tools developed for organizations to evaluate and compare, providing a basis for improvement and informed strategies to enhance specific areas within the organization [25]. The maturity models include a sequence of maturity levels for organizations or processes, outlining the expected, desired, or typical evolution path of these as discrete stages [26]. Many existing maturity models are used for software capability management but are not specific to research software project management. Therefore, they do not cover sustainability, reproducibility, impact measurement, promotion, visibility, and adoptability. Capability Maturity Model Integration (CMMI) and its predecessor Capability Maturity Model (CMM) are industry standard maturity models [27]. They include 5 maturity levels. CMM is used to evaluate an organization's software engineering processes regarding maturity. It helps developers to enhance software quality and the overall software engineering process. CMMI v3.0 [27] goes beyond software development and includes process quality assurance, configuration management, monitoring and control, planning, estimating requirements development and management governance, implementation infrastructure, organizational training, process management verification and validation. Therefore, it considers developers and other departments such as marketing, finance, and purchasing. The focus area maturity model is one type of maturity model [28]. It helps organizations to measure their performance in a particular

functional domain. A functional domain consists of different focus areas, each with its capabilities. These capabilities are arranged in a maturity matrix, which helps to identify different maturity levels. Each capability includes various improvement actions. These improvement actions support the organization in gradually improving in that functional domain. Unlike other maturity models, the focus area maturity model does not have fixed maturity levels; maturity levels can start from 0 and end at any positive integer. Each focus area is evaluated separately and has its maturity levels.

This work addresses the maturity gap by developing a tailored Research Software Focus Area Maturity Model (RSMM), integrating best practices across software project management, research software-specific challenges, and community-driven development." Inspired by the previous FAMMs [29, 30], we have developed our model RSMM to evaluate research software project management by assessing its maturity. We followed the De Bruin design phases to design the model [25]. The explanation for these steps is given below:

- **Scope:** The scope of RSMM is to evaluate and improve the management of research software projects.
- **Design** The design phase focuses on the questions "why," "how," and "who", as outlined below:
  - The **Why**: The purpose of RSMM is to help an organization that produces research software to improve their research software project management by assessing and improving the maturity of their projects.
  - The **How**: RSMM provides a structured framework with practices and capabilities for managing research software projects. It helps organizations learn about the practices that help them reach the desired maturity level and implement them effectively.
  - The **Who**: The intended audience of RSMM is researchers, research software engineers, research software project managers, funders, and policymakers.
- **Populate:** We identified the focus areas, capabilities, and practices of research software project management through a systematic literature review, resulting in RSMM v0.1.
- **Test:** This phase helps to place practices within the maturity matrix. Furthermore, we sent the resulting model, RSMM v1.0, to the interview experts to confirm and evaluate the model.
- **Deploy:** The updated version of our model, RSMM v1.0, is applied to 50 projects and validates the applicability of RSMM.

This RSMM is developed through a systematic literature review and expert interviews. Fifty research software projects are evaluated using RSMM to test its usability.

6. Timeline of the research

This research timeline started in February 2023 at the Netherlands eScience Center. The author is planning to finish the study at the proposed time in January 2027.

7. Expected Contributions

From the literature study, it is clear that there needs to be more effective methods for measuring the impact of research software and addressing its associated needs.

- We will identify and validate factors for measuring research software impact by conducting focus group discussions with stakeholders.
- We will develop an impact model aligned with stakeholders' goals based on the insights gathered from these focus group discussions.
- We developed a maturity model for research software projects, utilizing a systematic literature review and expert interviews.
- We will automate the categorization of research software based on its domain and maturity.

Lastly, this study aims to help researchers, research software engineers, and research organizations improve their project management processes, gain recognition for their contributions, and secure funding for future research software projects.

8. Future studies

Future studies of this research are listed below:

- Future studies could focus on refining and validating the impact model in different disciplines or types of research software.
- Conducting case studies to track the life cycle of research software to get more detailed insights into what affects its sustainability, growth, and retirement of the research software.
- We plan to investigate the influence of impact models and maturity assessments on funding decisions and policy development within research organizations. These findings can help establish best practices for justifying and securing funding for research software projects.

9. Conclusions

Research software impact means the difference that research software makes to research, the research community, and society. This study focuses on developing a research software impact model that can be used by all the research software stakeholders, researchers, research software engineers, funders, and policymakers to assess the impact of their software. Additionally, the study explores related topics, including developing a maturity model for research software and the automated classification of research software, providing an approach to understanding and managing research software impact.

10. Declaration on Generative AI

During the preparation of this work, the author(s) used Grammerly and OpenAI ChatGPT-4 Turbo to: Grammar and spelling check, Paraphrase, and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] 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, A. Struck, A. Lee, A. Loewe, B. van Werkhoven, C. Jones, D. Garijo, E. Plomp, F. Genova, H. Shanahan, J. Leng, M. Hellström, M. Sandström, M. Sinha, M. Kuzak, P. Herterich, Q. Zhang, S. Islam, S.-A. Sansone, T. Pollard, U. D. Atmojo, A. Williams, A. Czerniak, A. Niehues, A. C. Fouilloux, B. Desinghu, C. Goble, C. Richard, C. Gray, C. Erdmann, D. Nüst, D. Tartarini, E. Ranguelova, H. Anzt, I. Todorov, J. McNally, J. Moldon, J. Burnett, J. Garrido-Sánchez, K. Belhajjame, L. Sesink, L. Hwang, M. R. Tovani-Palone, M. D. Wilkinson, M. Servillat, M. Liffers, M. Fox, N. Miljković, N. Lynch, P. Martinez Lavanchy, S. Gesing, S. Stevens, S. Martinez Cuesta, S. Peroni, S. Soiland-Reyes, T. Bakker, T. Rabemanantsoa, V. Sochat, Y. Yehudi, R. F. WG, FAIR Principles for Research Software (FAIR4RS Principles), 2022. URL: https://doi.org/10.15497/RDA00068, version: 1.0.

[2] M. Barker, N. P. Chue Hong, D. S. Katz, M. Leggott, A. Treloar, J. van Eijnatten, S. Aragon, Research software is essential for research data, so how should governments respond?, 2021. doi:10.5281/zenodo.5762703.

[3] J. Carver, N. Weber, K. Ram, S. Gesing, D. Katz, A survey of the state of the practice for research software in the united states, PeerJ Comput Sci. 8 (2022) e963. doi:10.7717/peerj-cs.963.

[4] T. Greenhalgh, J. Raftery, S. Hanney, M. Glover, Research impact: a narrative review, 2016.

[5] H. Anzt, F. Bach, S. Druskat, F. Löffler, A. Loewe, B. Y. Renard, G. Seemann, A. Struck, E. Achhammer, P. Aggarwal, et al., An environment for sustainable research software in germany and beyond: current state, open challenges, and call for action, F1000Research 9 (2020).

[6] A.-L. Lamprecht, C. Martinez-Ortiz, M. Barker, S. L. Bartholomew, J. Barton, N. C. Hong, J. Cohen, S. Druskat, J. Forest, J.-N. Grad, et al., What do we (not) know about research software engineering?, Journal of Open Research Software 10 (2022).

[7] M. Barker, N. P. Chue Hong, D. S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, T. Honeyman, Introducing the fair principles for research software, Scientific Data 9 (2022). doi:10.1038/s41597-022-01710-x.

[8] M. Gruenpeter, N. Chue Hong, S. Granger, E. Breitmoser, M. Priddy, M. Antonioletti, P. A. Martinez, T. Honeyman, J. A. Collins, R. Meneses, Research Software Workshop: Guidelines and Metrics for Metadata Curation, 2023. doi:10.5281/zenodo.8075097.

[9] A. A. Younis, Y. Hu, R. Abdunabi, Analyzing software supply chain security risks in industrial control system protocols: An openssf scorecard approach, in: 2023 10th International Conference on Dependable Systems and Their Applications (DSA), IEEE, 2023, pp. 302–311.

[10] A.-M. Istrate, B. Veytsman, D. Li, D. Taraborelli, I. Williams, New data reveals the hidden impact of open source in science, 2022. URL: https://medium.com/czi-technology/new-data-reveals-the-hidden-impact-of-open-source-in-science-11cc4a16fea2.

[11] H. Borges, M. T. Valente, What's in a github star? understanding repository starring practices in a social coding platform, Journal of Systems and Software 146 (2018) 112–129.

[12] J. H. Spaaks, T. Klaver, S. Verhoeven, F. Diblen, J. Maassen, E. Tjong Kim Sang, P. Pawar, C. Meijer, L. Ridder, L. Kulik, T. Bakker, V. van Hees, L. Bogaardt, A. Mendrik, B. van Es, J. Attema, W. van Hage, E. Ranguelova, R. van Nieuwpoort, R. Gey, H. Zach, **Research Software Directory** (2020). URL: https://github.com/research-software-directory/research-software-directory. doi:10.5281/zenodo.1154130, version: 3.0.1.

[13] P. Boldi, A. Pietri, S. Vigna, S. Zacchiroli, Ultra-large-scale repository analysis via graph compression, in: 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2020, pp. 184–194.

[14] D. Singh Chawla, The unsung heroes of scientific software, Nature 529 (2016) 115–116. doi:10.1038/529115a.

[15] J. H. Spaaks, S. Verhoeven, E. Tjong Kim Sang, F. Diblen, C. Martinez-Ortiz, E. Etuk, M. Kuzak, B. van Werkhoven, A. Soares Siqueira, S. Saladi, A. Holding, **HOWFAIRIS**, 2022. URL: https://github.com/fair-software/howfairis, version: 0.14.2.

[16] S. Verhoeven, F. Diblen, J. H. Spaaks, E. Tjong Kim Sang, **tortellini**, 2021. URL: https://github.com/tortellini-tools/action, version: v3.

[17] S. Jansen, S. Farshidi, G. Gousios, J. Visser, T. van der Storm, M. Bruntink, Searchseco: A worldwide index of the open source software ecosystem., in: Proceedings of the 19th Belgium-Netherlands Software Evolution Workshop (BENEVOL 2020), volume 2912 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020.

[18] P. Orviz, J. Gomes, S. Bernardo, D. Naranjo, M. David, EOSC-SYNERGY, EOSC-SYNERGY EU DELIVERABLE: D3.4 Final release of the SQAaaS, 2022. URL: http://hdl.handle.net/10261/274895.

[19] M. C. Paulk, B. Curtis, M. B. Chrissis, C. V. Weber, Capability maturity model, version 1.1, IEEE software 10 (1993) 18–27.

[20] W. Hasselbring, S. Druskat, J. Bernoth, P. Betker, M. Felderer, S. Ferenz, A.-L. Lamprecht, J. Linxweiler, B. Rumpe, Toward research software categories, arXiv preprint arXiv:2404.14364 (2024).

[21] Deekshitha, C. M. Ortiz, R. Bakhshi, J. Maassen, R. van Nieuwpoort, S. Jansen, T. Smeele, A. Treloar, L. Bezuidenhout, M. Schermer, L. Sesink, P.-K. Fung, C. Bos, Decoding research software impact: A collaborative journey, https://blog.esciencecenter.nl/decoding-research-software-impact-a-collaborative-journey-0db9aa88415e, 2024. Medium article.

[22] C. Martinez-Ortiz, M. Kuzak, J. H. Spaaks, J. Maassen, T. Bakker, Five recommendations for "FAIR software", 2020. doi:10.5281/zenodo.4310217.

[23] Deekshitha, S. Farshidi, J. Maassen, R. Bakhshi, R. Van Nieuwpoort, S. Jansen, FAIRSECO: An Extensible Framework for Impact Measurement of Research Software, in: Proc. IEEE Conf. on e-Science (e-Science), 2023, pp. 1–10.
[24] W. Pullen, A public sector HPT maturity model, Performance Improvement 46 (2007) 9–15. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/pfi.119. doi:https://doi.org/10.1002/pfi.119.
[25] T. De Bruin, M. Rosemann, R. Freeze, U. Kaulkarni, Understanding the main phases of developing a maturity assessment model, in: Proc. Australasian Conf. on Information Systems (ACIS), 2005, pp. 8–19.
[26] J. Pöppelbuß, M. Röglinger, What makes a useful maturity model? a framework of general design principles for maturity models and its demonstration in business process management, 2011, p. 28. URL: http://aisel.aisnet.org/ecis2011/28.
[27] ITG Consulting, CMMI v3 and the transition from CMMI v2, https://consulting.itgonline.com/cmmi-consulting/cmmi-v3-and-the-transition-from-cmmi-v2/, ???? Accessed: May 1, 2024.
[28] M. van Steenbergen, R. Bos, S. Brinkkemper, I. van de Weerd, W. Bekkers, The Design of Focus Area Maturity Models, in: Proc. Conf. Global Perspectives on Design Science Research (DESRIST 2010), Springer, Springer Berlin Heidelberg, 2010, pp. 317–332.
[29] S. Jansen, A focus area maturity model for software ecosystem governance 118 (2020) 106219.
[30] M. Overeem, M. Mathijssen, S. Jansen, API-m-FAMM: A focus area maturity model for API Management 147 (2022) 106890.