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

Research software as a new key performance indicator for evaluation in the Helmholtz Association

Guido Juckeland, Doris Dransch, and Bernadette Fritzsch

DOI: 10.14279/eceasst.v85.2697

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

Research software as a new key performance indicator for evaluation in the Helmholtz Association

Guido Juckeland[1], **Doris Dransch**[2], **and Bernadette Fritzsch**[3]

[1] G.Juckeland@hzdr.de
Helmholtz-Zentrum Dresden Rossendorf (HZDR),
[2] Dransch@gfz.de
Helmholtz Centre for Geosciences (GFZ),
[3] Bernadette.Fritzsch@awi.de
Alfred-Wegener-Institut, Helmholtz-Zentrum für Polar- und Meeresforschung (AWI)

Abstract: The important role of research software in science is being increasingly recognized. Nevertheless, there is still a lack of adequate recognition of research software as an independent outcome of scientific work. Although there are already a number of initiatives to anchor research software in research assessment, there are still no uniform criteria with regard to the necessary quality assurance. This paper presents a possible approach for a quality indicator that has been developed in the Helmholtz Association. First experiences with an initial version of this indicator are discussed, the application of the new quality indicator is illustrated using examples and ideas for a possible automation of the review process for research software are outlined. The paper aims to contribute to the discussion on how research software can be suitably integrated into general research assessment.

Keywords: resesarch software, research assessment, software quality, quality measurement

1 Introduction

Measuring the performance and progress of research based on means of output is a common task of research assessment. While it is commonly accepted to use quality-assured scientific publications as a proxy for quality assessment in research evaluation, other artifacts of scientific work such as research software are often not yet considered. Therefore, researchers and research software engineers were not incentivized for their work on software and the necessary quality assurance. As a result, publishing software with assured quality has long lived in the shadows and only in recent years have developers, users of research software and funding organizations become more aware of it. As research increasingly relies on software to generate, process, and analyze data, and publishing research software impacts validation and reproducibility of scientific findings, research assessment should be put on a broader footing and consider the entire range of scientific output, such as published research software (e.g.[Eur24], [Deu19]).

The focus on scientific papers is currently put into question by several initiatives that reformulate recommendations for enhanced research assessment. One of those is the San Francisco

Declaration on Research Assessment DORA[1] developed in 2012 during the Annual Meeting of the American Society for Cell Biology. In the meantime, it has become a worldwide initiative covering all scholarly disciplines and many key stakeholders including funders, publishers, professional societies, institutions, and researchers. In 2022 the European initiative "Coalition of Advancing Research Assessment CoARA" [2] started the process of drafting an agreement on reforming research assessment. More than 350 organizations from over 40 countries have signed the declaration with the number still increasing. Both, DORA and CoARA clearly state that "there is a pressing need to improve the ways in which the output of scientific research is evaluated by funding agencies, academic institutions, and other parties"[3].

The Helmholtz Association has been taking up these impulses by including research software into account in the research assessment. In this process the challenge arises to assure the quality of artifacts counting in a key performance indicator. In analogy to peer review for text publications, there are already approaches to a code review for software for example at some conferences (see e.g. [KV15] or [NE21]). But these have not yet been widely used, partly because of the greater effort involved for the reviewers. Therefore, a quality indicator was needed which ensures the quality of published software in a more practical way. To address this challenge, the Helmholtz Association is developing and implementing a quality indicator for research software. It should become a driver within Helmholtz in a threefold way:

1. to create awareness and valuation to the diverse research output,
2. to align the assessment of research and research practice to the conditions of openness of science, and
3. as a stimulus and incentive for scientists and research software engineers to improve their software related to specific quality criteria.

A task group has been set up filled with a variety of experts, such as scientists, scientific software developers, data curators, and people reporting on research output, to meet all aspects relevant for the quality indicator. Helmholtz Centers from different research fields contributed in an iterative discussion process to ensure a broad perspective originating from their respective research environment. The process of developing a quality indicator for research software was divided in two phases. It started with a simple initial quality indicator to gain experiences about suitable dimensions of a quality indicator and operational feasibility. Based on the experiences an advanced quality indicator has been developed in a second phase.

This paper presents the concept, experiences and outcomes with the Helmholtz quality indicator for research software. It is structured in the following way: Section 2 describes existing approaches for quality estimation of research software. It is followed by a description of the initial quality indicator (section 3) and the improved quality indicator (section 4) of the Helmholtz Association. To demonstrate practicability and efficiency of the quality indicator, section 5 presents a proof of concept with various examples from a broad application field and possible automation of quality assessment supported by suitable tools. Conclusions derived from the experiences that have been made during the development of the quality indicator are discussed in section 6.

[1] https://sfdora.org
[2] https://coara.eu/
[3] DORA https://sfdora.org/read/

In this paper the following definitions are used:

Research software

Following [GKLe21] research software is defined as software that has been "created during the research process or for a research purpose". All types of software are considered as defined by [SHM18] in the form of software application classes: from personal use, reuse by others, long term support, and criticality.

Published software

The document concentrates on research output which is published. Therefore, published software is defined as digital objects that have been made as permanently available as possible on the internet. This includes different forms of publication, from classical releases in the version control system used for development up to special software publications with Digital Object Identifier (DOI).

Quality indicator

The quality indicator is meant as a quantitative measure for the quality of the published artifact, in our case published research software. Like the peer review process for text publications, the quality indicator aims to ensure a certain level of quality of the publication. Only publications which fulfill the requirements of the quality indicator are counted in the key performance indicator for published software in the research assessment.

Key performance indicator for published software

The key performance indicator (KPI) for research software is a measure for all published software with the required quality. It is aimed to include research software as an important output of research into the assessment.

2 Existing approaches for quality estimation of research software

Metrics for software quality have been the subject of research in software engineering since the 1960s (see e.g. [RH68]). Its ever-present relevance is evidenced by its own conferences and journals on the topic of software quality. In the meantime, several metrics and standards for determining quality have been developed and established in the industry. Pinedo et al [PVN+23] and Nistala et al [NNR19], for example, provide an overview of various software standards such as ISO/IEC 25010 or CMMI (Capability Maturity Model Integration) and other quality models based on these. However, the term "software quality" is understood and used very differently. Kitchenham et al [KP96] therefore posed the question of which quality is considered in which context. It is recognized that software quality is multi-faceted. Moreover, various facets can be in direct conflict, such as improvement in one quality dimension may decrease the quality in another. Depending on the shareholders involved and the focus they have, a different view of software quality is taken.

This also partly explains why the quality models established in industry have not yet been able to gain widespread acceptance in research software, as other interest groups are involved. There is a lack of standardized metrics and benchmarks tailored to the unique needs of research software. The diversity of research domains and the specificities of their computational problems make it difficult to develop universal quality standards. Additionally, the often limited resources available for software development in research settings can hinder the implementation of com-

prehensive quality assurance processes. One approach for evaluating software quality is the use of badges, like the OpenSSF Best Practices Badge Program or the fair-software.eu badge. They offer a light certification of software, but focus only on some aspects of software quality.

It is therefore necessary to find criteria that capture different aspects of research software quality (as automatically as possible). As stated in [CKH21] a fundamental challenge is to translate the requirements from the quality dimensions into measurable valid metrics. There are often many ways to approach quality dimensions. One example is the catalog developed by a working group of RDA [CBA+23]. This checklist includes information on persistent identifiers of the software and its components, associated metadata, documentation, etc., in each case with reference to the domain-specific standards widely used in the scientific community. A similar catalog developed in EOSC with a total of 126 Quality Attributes and Metrics [DCG+24], which is based on extensive literature research, is considerably more comprehensive. It takes into account 25 different characteristics for software quality. Establishing research software quality for evaluation needs an integration of quality assessment practices into the daily workflow of researchers, who may prioritize scientific innovation over software engineering rigor. So, this catalog seems to be too complex.

Deekshitha et al [DBM+24] present a framework with which research software projects can be evaluated. Divided into four different focus areas, 17 different capabilities are considered, some of which also allow statements to be made about software quality. Especially for an automated evaluation of research software, FAIRSECO [DFM+23] provides information about the FAIRness of software, offers license and dependency checking and code indexing.

Our indicator has a clear focus on open science, for that reason it is based on the FAIR Principles adopted for Research Software (FAIR4RS) [BCK+22] and takes up related activities and attempts to provide a set of metrics to enable a multidimensional analysis of research software in the context of the Helmholtz Association. The catalog of quality criteria developed in the task group extends the FAIR principles by further dimensions (see section 4.2.1) and presents the translation from quality dimensions into operationalizable metrics. It contains feedback from many developers from the various Helmholtz institutions.

3 The initial quality indicator of the Helmholtz Association

3.1 Concept and purpose of the initial quality indicator

For the introduction of the new indicator for software a step-wise approach was used. Starting point was a very simple indicator which considers only two aspects. First, as the approach should be supporting Open Science and reproducibilty, the software should be citable, e.g. should have a persistent identifier (PID). The second aspect concerns the repository in which the software is published. It should have curated metadata and fulfill some further requirements for the trustworthiness. The repository has to be listed or certified by re3data [4]. As this first attempt was a Helmholtz internal measure, also repositories which are operated with participation of a Helmholtz center were added. The corresponding KPI was defined as the number of citable software publications. The main advantage of this approach is that the requirements can be proven very simple and fast.

[4] https://www.re3data.org/

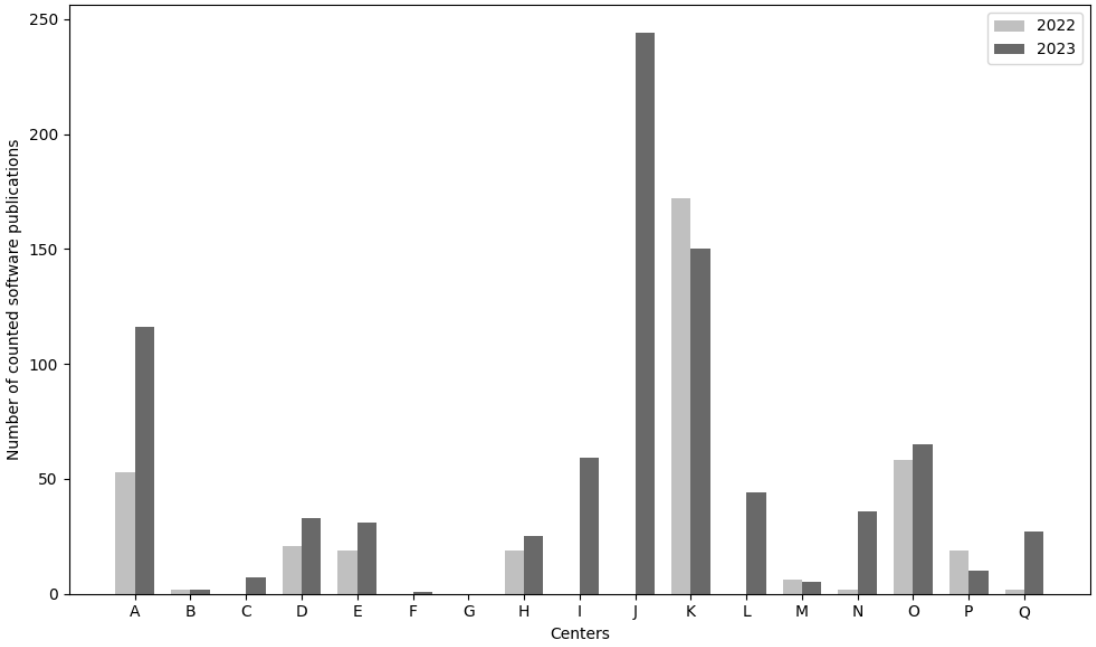

Figure 1: Number of published software reported from different centers for reporting year 2022 and 2023.

The aim of this first step was to test the operational feasibility for the new included research outcomes into evaluation process. It should incentivize the recordability of published research software in a broad approach via a simple count. The indicator should be used to monitor the development over time in each center. The reports were aimed to give a first overview about published software in the Helmholtz Association and to initialize the monitoring of published software in the centers. The resulting landscape of published software could give valuable hints for the second step of developing an advanced quality indicator.

3.2 Experiences gained with the initial quality indicator and the derived KPI

For the reporting year 2022, all centers were asked for the number of software publications with PID and with the mentioned requirements of the corresponding repository. The outcome was very heterogeneous. Although it is known, that research software is developed in all Helmholtz centers, not all centers monitor these activities. So only 11 of 17 centers reported some software publications. This can be explained by the fact that this type of publication is not yet systematically recorded in some institutions or that corresponding publication processes have not yet been established. In this respect, the first survey was a wake-up call for some centers to raise awareness of research software. It inspired in some centers discussions about the processes of software publications and their monitoring. In the following reporting year 2023, already 16 centers were able to record and report software publications. Figure 1 shows an overview about the recorded numbers of software publications for 2022 and 2023.

But even in centers where software publications are already recorded, there is still room for improvement. As text publications have so far been counted as the sole output of research, new types of publications are often not well enough known. This was confirmed by an internal survey in one center about research software, its development and publishing practices [FA24]. It turned out, that only a small amount of software was published with PIDs. The reasons were often a mind which does not fully embrace openness and that researchers sometimes do not know about software publications and how to do it. Therefore, it is essential that researchers are better informed. This should also be seen in terms of the expansion of Open Science and supporting reproducibility.

The reported numbers of software publications increased between the first and second reporting year in almost all centers. This shows, that even the introduction of such a simple and imperfect indicator for research software can increase the awareness for software as an important outcome of research. The previous hidden work in software development and maintenance can be made more visible in research assessment.

The reported data show some outliers, where very high numbers of published software was reported. This reflects the fact that no common understanding about the granularity of counted publications existed at that time. For research software from the point of reproducibility also small changes in the code should be documented by a new publication of the software when it is used for a research paper. But with respect of research output assessment it should be assured that only software publications with a certain amount of modifications should be counted in the key performance indicator.

3.3 Conclusion

A closer look at the reported publications shows, that the chosen quality criteria (persistent identifier for the software and requirements for the trustworthiness of the repository) has proven to be inadequate for a good new key performance indicator in the context of the research evaluation. The publications are very heterogeneous with respect to their quality and the mere existence of a PID is not enough to ensure that only comparable units are counted . So the development of a more sophisticated indicator is necessary which is shown in the next section 4.

Additionally, the surveys show the high amount of manual work needed to collect the corresponding data from different data sources. While it is relatively easy to check the actual criteria of the initial indicator, the difficulty lies in finding the relevant publications in the first place. In the metadata for the repositories the information about the affiliation are often not mandatory and therefore in many cases not provided by the authors, or affiliation names are written in different ways. The usage of a unique identifier for the affiliation [5] can help here. But overall, the further collection of data with the improved quality indicator must be supported by appropriate tools.

4 Development of an improved indicator

The landscaping in the Helmholtz Association via the initial quality indicator revealed a broad diversity of research software as well as its employed quality practices. This is quite natural

[5] like research organization registry https://ror/org

as a small analysis script that supplements a paper does not have to fulfill the same quality requirements as a software that is an integral part of a large research infrastructure. A reflection of these different levels of research software quality is reflected in the concept of ”research software kinds” [Com22] as well as the application classes as defined in the DLR Software Engineering Guidelines [SHM18].

Looking at quality beyond the challenges identified already with the initial indicator requires a more generic and diverse approach that does not focus on a single number. The quality indicator for published research software as an improved indicator takes inspiration from generic process oriented product quality definitions that also need to reflect a broad spectrum a outputs.

4.1 General quality and maturity concepts

Quality as defined in ISO 8402, the precursor of the ISO 9000 quality standard series, was the starting point of our discussion:

“The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs.” [Ire90]. This very general definition has to be adapted to our needs by a quality model.

Quality models offer a framework that defines what constitutes quality in a given context. They provide a set of criteria to ensure quality (e.g. ISO/IEC 25000:2014(en) Software Engineering — Software product Quality Requirements and Evaluation (SQuaRE)[6]).

One goal of the improved quality indicator is also to improve quality and trustworthiness of software as research outcomes over time. For this reason maturity models are a suitable approach. They help organizations develop their capabilities with respect to certain quality goals (overviews can be found, for example, [SC19] or at https://en.wikipedia.org/wiki/Maturity_model#cite_note-4). Maturity models focus either on the underlying products (like the Technological Readiness Level TRL focusing on technologies[7] or measure process quality, such as the Capability Maturity Model CMM focusing on software development processes [Hum88], but it can also be applied to other processes (like Human Capital Management processes, see [CHM09]. While product-quality oriented models concentrate on the properties of products, process-quality oriented models focus on the production process of the product. The implicit assumption behind process oriented models is that a higher level of maturity of the respective processes also leads to a higher quality of the product. A maturity model consists of a sequence of maturity levels: The bottom level represents an initial state. The highest level represents the target of highest maturity. This outlines an evolution path along which the product to be assessed can move.

4.2 Determine and measure quality: Quality features and maturity levels

Based on the general quality concepts introduced above, the improved indicator is designed as an approach to determine quality features and measure the maturity of these features. The indicator should fulfill several requirements:

[6] https://www.iso.org/standard/64764.html
[7] https://en.wikipedia.org/wiki/Technology_readiness_level

1. represent a diverse set of quality dimensions (requirement 1)
2. provide one single easily comparable quality value in itself (requirement 2),
3. be applicable in general for published software; for the specific implementation this generic approach has to be adopted to the specific research software characteristics (requirement 3), and
4. rely on common quality concepts (requirement 4).

Requirements 1,3 and 4 can be met by choosing appropriate quality features (see 4.2.1), for requirement 2 a method is described in subsection 4.2.2.

4.2.1 Determine quality features

The improved indicator introduces two levels of quality feature descriptions: the *quality dimension* and the *quality attribute* (see requirement 1 and 3). The quality dimensions group quality features in categories, such as "findable" or "reusable"; the quality attributes characterize a quality dimension, they are a refinement in the sense of clear measurable characteristics. For instance, the quality dimension "findable" in the context of software is characterized by quality attributes such as the availability of an open publication repository, versioning, publication with an identifier, and rich metadata. The outcome is a list of quality dimensions and corresponding quality attributes for research software[Tas25]. The specific quality features were determined by following current community best practices (see requirement 4).

In the research data community, the FAIR principles[WDAa16] have now become an integral part of the work. They have been widely accepted as a minimum set of quality features for research data and have been also adopted to research software as FAIR4RS [BCK+22]. The four principles findability, accessibility, interoperability and reusability therefore fulfill requirement 3 and 4 as accepted principles of the communities and are generally applicable (with certain specific characteristics) for research data and software. They are multidimensional (requirement 1) and represent different aspects of quality.

These established principles were complemented by some further criteria as they mainly consider metadata aspects of publishing software and not the software development processes themselves or the actual scientific embedding.

Two dimensions were added to reflect these needs: scientific basis (S), addressing the question of research software being scientifically well grounded, and technical competence (T), capturing aspects of the software being technically well grounded in the sense of applying software development principles. This results in six dimensions, abbreviated as FAIR-ST.

4.2.2 Determining maturity levels

Maturity models have been successfully used in various contexts for decades, also in IT and for digital processes [Cus20]. The improved indicator builds on this concept to benefit from the two main advantages: 1) processes are elements of organizations, such as research institutions, and can be used for steering and controlling quality, and thus continuously improve quality

Table 1: Maturity levels for attribute T1: Project management.

Level	Description
0	No information on project management and code history being provided.
1	A version control system is used.
2	A version control system being part of a code project management platform (e.g. GitHub, GitLab) and an associated ticket system is in place.
3	A transparent process for ticket resolving, code review by other developer, and merge requests is established.
4	A release process with guaranteed changelog generation, testing, and product provisioning is established.

institution-wide, and 2) assessing a process creating a specific result is less costly than assessing each single result.

Determining the maturity level of quality attributes Inspired by existing models, the improved indicator defines the same number of maturity levels for all quality attributes. The development strategy was designed as a community process to ensure broad participation across the various scientific disciplines. For the quality attributes, important criteria were first collected and then sorted according to their relevance in a lengthy discussion process in the task group to determine the extent to which they represent an ascending level of maturity. Then after this initial version, the proposed model was discussed at the various Helmholtz Research Centers and the feedback was incorporated into a refined version. A test phase was agreed upon for the introduction of the model, after which there is to be an evaluation in which further modifications are to be expected. The result of the first loop in this iterative process defines four maturity levels (ranging from 0 to 4) for each quality attribute the published software. As an example, the corresponding maturity levels for the attribute ”project management” in the dimension ”Technical Basis” are here described. The quality attribute aims to evaluate, how strictly methods of project management are applied in the development of the software. The first level is reached when the software development is done in a version control system. The next level adds then the use of code project management features. For level 3 and 4, the processes behind the development are coming into the evaluation. They include processes for ticket resolving, code review and product provision. The levels build on each other. This means that in order to reach a certain level, the conditions of all lower levels must be fulfilled beforehand. In table 1 the corresponding level definitions are shown.

The complete overview about the first version of the maturity levels for all attributes is published together with [Tas25]. For a first overview a condensed form of the table from [Tas25] is shown in Table 2

Table 2: Short overview of all quality dimensions and used attributes. This is a condensed version of the table in [Tas25].

Dimension	ID	Attribute	Short Description
Findable	F1	Open Publication Repository	Evaluation of the accessibility of the code in an open publication repository and the corresponding information there. From no information available up to repositories which have structured metadata descriptions and are listed in overarching metarepositories (e.g. e.g. Helmholtz Research Software Directory (RSD), re3data).
	F2	Versioning	Evaluation of the use of versioning systems. From no software versioning up to defined structured release cycles.
	F3	Published with Identifier	Evaluation of the existence of identifiers for the code. From no identifier up to a software publication is identifiable via a resolving globally unique, formalized, standardized, persistent identifier supported by general metadata (e.g. resolvable DOI)
	F4	Rich Metadata	Evaluation of existence and quality of metadata. From no metadata at all up to automatically harvestable metadata information following given metadata scheme.
Accessible	A1	Access Conditions (organizational)	Evaluation of the legal constraints for accessibility. From not specified up to licenses from the FLOSS list.
	A2	Access Options (process)	Evaluation of the accessibility of code and corresponding verification and validation mechanism to ensure correct functionality. From only one specific form of accessing up to provided tests cases for checks.
	A3	Technical Accessibility (run/start)	Evaluation of how easy the installation of the software is. From no information given up to complete package that enables execution (e.g. container).
Interoperable	I1	Input/Output Formats	Evaluation if the data for and from the code can be used in a workflow using established formats. From not specified formats up to building on accepted community standards for input/output data.
	I2	Adaptability/ Flexibility of Use	Evaluation of how easy the software can be integrated into new users' own software framework. From no information given up to documentation of integration.
Reusable	R1	Support	Evaluation of how support for using the code is provided.
	R2	Reusability Conditions	Evaluation of the legal aspects of reuse (licenses). From unclear conditions up to available processes for automatic checks (e.g. REUSE specification).

Table 2: Short overview of all quality dimensions and used attributes. This is a condensed version of the table in [Tas25].

Dimension	ID	Attribute	Short Description
Scientific basis	S1	Community Standards	Evaluation of the compliance with standards of the corresponding scientific community. From no information up to indications on addressing further evolution of community standards.
	S2	Team Expertise	Evaluation of the scientific expertise of the developer team. From no information given up to established interdisciplinary team.
	S3	Scientific Embedding	Evaluation of the relevance of the software for the scientific community. From no information given up to software development is part of a larger scientific initiative with dedicated processes for software development.
Technical Basis	T1	Project Management	Evaluation of the application of project management methods on the software development. From no information up to defined release processes with guaranteed changelog generation, testing, and product provisioning
	T2	Repository Structure	Evaluation of the repository structure. From no information up to documented contribution mechanisms and provided templates.
	T3	Code Structure	Evaluation of the code structure. From no information up to code style is enforced via a review process.
	T4	Reproducibility (Code)	Evaluation of how the code supports reproducibility. From no tests up to automated testing for different system environment.
	T5	Code change process	Evaluation of the change process. From no information up to defined processes for integration of code changes.
	T6	Security	Evaluation of Security aspects of the software. From no security concepts up to regular and automated security monitoring and an automated update process.

Determining the maturity level of quality dimensions To provide one single easily comparable quality value for each quality dimension as well as representing a diverse set of quality features in itself (see requirement 1), the indicator aggregates the maturity levels for all quality attributes in one quality dimension with a weighted summation (see Table 3) The result for all dimensions can then also be visualized in a radar plot (see Figure 2).

Figure 2 exemplifies the result of the quality assessment of three fictitious published research software codes. The aggregation of one quality dimension, i.e. the value along one axis of the radar plot, is determined by a weighted average of all quality attributes of the dimension as shown in Table 3. This enables the assessment process to reflect the prioritization of various aspects of published research software as defined by the various research communities.

Table 3: Example for the aggregation of the attributes for one dimension of the proposed quality indicator.

	Measured Maturity Level (0-4)	Weight	Overall contribution
Attribute 1	4	0.2	0.8
Attribute 2	3	0.5	1.5
Attribute 3	2	0.3	0.6
Sum		**1.0**	**2.9**

Determining the quality indicator of published research software and the corresponding KPI For the creation of a single metric per published software, the new KPI, there are two equally valid approaches:

a. A "0/1 approach" counting all publications that fulfill a minimum threshold. In the case of the examples in Figure 2 both publications A and B would count as "1" as they are in all quality dimensions above the minimum requirements while publication C fails to meet the minimum requirements in dimension 1 and 6. This approach has the benefit that creating a key performance indicator KPI for a single institution is straightforward as one only needs to count all publications in the reporting period that fulfill the minimum requirements. Also, by regularly reviewing and increasing the minimum requirements, the approach can be adapted to improved quality standards of the research communities.

b. A "relative ratio approach" showing as to how close the publication is to the maximum score per quality dimension. This can be achieved by generating a weighted average across the ratio of each quality dimension compared to the maximum score for that dimension. The final result is a percentage reflecting how close the measured publication is to the "ideal" (i.e. maximum score) publication. This approach is illustrated in Table 4. The benefit of this approach is that the generated overall value allows quality comparisons between different publications to a certain extent and allows a more precise classification of the quality through this figure in contrast to the "good enough" approach

   The corresponding overall KPI is then a global average across all publications of one institution and reflects the overall quality of published research software for that institution.

Table 4: Examplified aggregation across all dimensions for publication A from Figure 2 using the proposed approach 2 for aggregating.

Dimension	Aggregated Maturity	Weight	Overall contribution	Max. score	Overall ratio
1	3.5	0.3	1.05	3.7	95%
2	2.2	0.3	0.66	2.8	78%
3	3.7	0.1	0.37	4	93%
4	2	0.2	0.4	3.5	57%
5	3	0.1	0.3	3.3	91%
6	1	0	0	2.4	42%
Sum		**1.0**	**2.78**	**Weighted Average**	**81.7%**

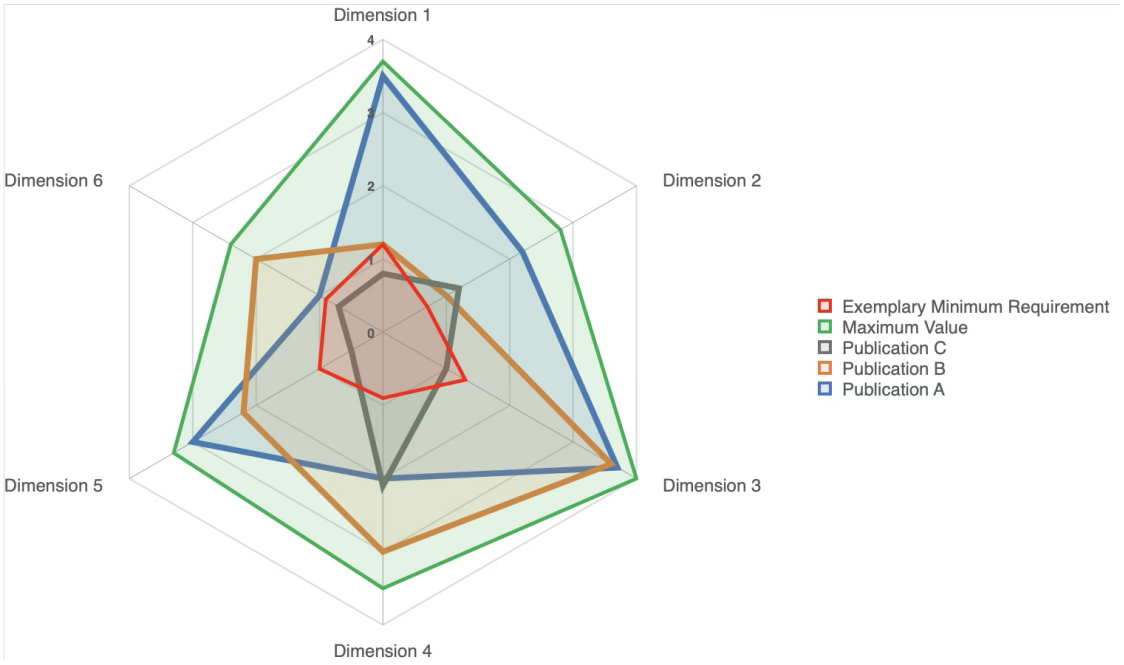

Figure 2: Example of a fictitious assessment for three published research software codes showing how the radar plot can visualize the fulfillment of some criteria at higher quality levels than others. Maturity levels are ordered from center (0, non-existent) to their currently measurable maximum, which is up top 4.

The Helmholtz Association has chosen approach "a" for its KPI because it was perceived as better aligning with its open science strategy, providing an incentive to publish all software that fulfills the KPI's minimum requirements. The corresponding workflow for each Helmholtz center is shown in Figure 3. The new KPI is included in the future reporting of the Helmholtz Association towards its funding agency.

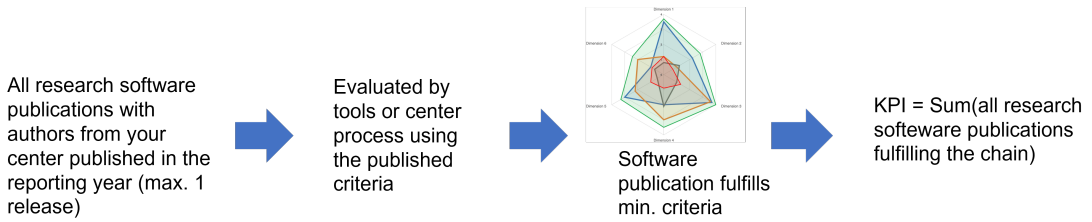

Figure 3: Workflow for determining the reported KPI for each center of the Helmholtz Association that aggregates the number of software publications.

5 Proof of concept and first results

5.1 Proof of concept

The feasibility of the concept was evaluated in various research communities. While the design of all attributes for each quality dimension has been discussed and iterated in several rounds of community engagement, their actual application to assess the quality of given published software has been proven by running a couple of exemplifying assessments. A detailed maturity level questionnaire for all quality dimensions and attributes ([Tas25]) has been provided. It presents a table where the maturity levels are specified for each quality attribute.

The goal of the assessment was to show

1. whether the formulation of the quality dimensions and attributes is clear enough to be applied in a wider community, and

2. if the specification of the maturity levels is understandable and practicable, and where it needs to be improved.

The detailed maturity level questionnaire for all quality dimensions and attributes was manually assessed for a number of representative research software publications. The respondents were asked to both supply the maturity level they currently assess their publication for each quality attribute and also provide comments as needed, e.g. about unclear descriptions of attributes or maturity levels or why it might not be applicable for the particular publication or the science community. The evaluation was conducted by colleagues who had no prior contact to the assessment approach; the intention was to get an unbiased view on the applicability of definitions and to resemble the long-term situation where the collection of this kind of data is the responsibility of colleagues unaffiliated with the developers of the approach. Ten Helmholtz Centers from various scientific disciplines took part in this quality assessment and tested the developed questionnaire using a total of 23 software publications.

5.2 First results

This section presents results of the proof of concept studies described above for two published software examples and gives a first resumé about the feasibility of the introduced quality concept.

Test software 1: alpaka - an abstraction library for access to heterogeneous hardware

One of the software examples from Helmholtz-Zentrum Dresden-Rossendorf (HZDR) is the hardware abstraction library alpaka [ZWW+16][8]. The library implements an abstract hardware model, thus allowing users to write platform and performance portable C++ code that can be compiled for standard processors but also hardware accelerator devices. The code represents a large community code that is used by groups world-wide and, as a result, also has established very high quality software processes. In particular, the GitHub software development reposi-

[8] https://github.com/alpaka-group/alpaka

tory is already linked with Zenodo as a publication repository[9], so that each software release is automatically also turned into a persistent software publication.

The authors of the software could largely assess the quality dimensions and attributes according to the specified maturity levels. However, the authors struggled with the dimension "Interoperable" as alpaka as a library from their point of view fulfills the requirements of interoperability. However, the attribute asking for input/output formats does not match well towards a header-only C++ template library, such as alpaka. Here improved documentation is expected to mitigate the issue. Furthermore, some attributes, especially the ones in the dimension of scientific embedding of the software, were answered from their own point of view and were described as hard to be evaluated by tools other than external reviews.

The evaluation of the software using the improved indicator is shown in Table 5 and also visualized in Figure 4. As infrastructure software alpaka scores, as expected, very high in almost all categories. Since the software itself is a library, it suffers from the current phrasing of interoperability as it is not directly used to work with scientific data. It is also situated in research communities where no community standards towards research software quality have yet been established.

Table 5: Evaluation for test software 1: alpaka.

Dimension	Attribute	Evaluation	Mean
Findable	F1: Published via Zenodo and Helmholtz RSD	4	3.5
	F2: Uses semantic versioning	2	
	F3: Published via Zenodo with a DOI	4	
	F4: Metadata information in .zenodo.json as part of GitHub repo	4	
Accessible	A1: Chosen software license Mozilla is in FLOSS list	4	3.66
	A2: Software includes self-validation as part of Cmake install	4	
	A3: Automated installation supported via Cmake	3	
Interoperable	I1: Software is a library not directly doing input/output	0	2
	I1: Software is a library ready to be integrated anywhere	4	
Reusable	R1: (not yet available)		3
	R2: Separate licenses for code and examples	3	
Scientific Basis	S1: No standard known for the community	0	2.66
	S2: Fixed, international, interdisciplinary team	4	
	S3: Part of both CERN CMS and HZDR DMA	4	
Technical Basis	T1: Established processes followed for all releases and changes	4	2.83
	T2: Contributors.md and clear structure used	3	
	T3: Tools for linting used and also part of review	4	
	T4: Requirements documented in README	2	
	T5: Only maintainers can update main branch	4	
	T6: Not relevant in the science domain	0	

[9] https://doi.org/10.5281/zenodo.595380

Test software 2: Astronaut Analysis Data Publication – an example from the HIFIS education portfolio[10]

This sample code is taken from the HIFIS (Helmholtz Federated IT Services[11]) course materials. It is intended to demonstrate how to prepare your own code and associated data for publication. The assessment deliberately focused on the script and not on the general development of teaching materials. This was done, because the script represents a good example of simple code that is often developed for personal use. As it therefore belongs to a low application class, the necessary requirements are correspondingly lower.

Table 6: Evaluation for test software 2: Astronaut Analysis training example.

Dimension	Attribute	Evaluation	Mean
Findable	F1: Some kind of description in README	2	2.25
	F2: Some kind of versioning https://calver.org/	1	
	F3: Has a DOI after doing the steps in the workshop	4	
	F4: Metadata information in CITATION.cff	2	
Accessible	A1: Chosen software license Apache is in FLOSS list	4	3
	A2: Some information about installation in README	2	
	A3: Semi-automated installation supported by test.sh	3	
Interoperable	I1: Input/output in standard formats (JSON, PNG)	2	1.5
	I1: Set of input data provided	1	
Reusable	R1: (not yet available)		4
	R2: Process for automatic checks following REUSE	4	
Scientific Basis	S1: No information given	0	0.33
	S2: Expertise in software development?	1	
	S3: No scientific use case (only teaching material)	0	
Technical Basis	T1: Version control system is used	1	1.02
	T2: Structured repository	1	
	T3: Developer is free to use own style of coding	1	
	T4: Requirements in README documented	2	
	T5: Transparent processes for code changes (pull requests, merge)	2	
	T6: No security concept	0	

Table 6 shows the ratings for the individual attributes and the resulting radar plot is shown in Figure 4. The evaluation reflects that the intended use (teaching material) results in a rating of the respective quality aspects. As an example in a course, it must be easy to find and access. As the script serves to demonstrate the steps for publication of software, the aspect of licensing has been particularly well worked out, which stands out in the ”Reusable” dimension, which currently only has one attribute for evaluation. As this is a sample script for teaching purposes, the scientific basis is secondary and a simple use case was selected that can be easily understood

[10] https://codebase.helmholtz.cloud/hifis/software/education/hifis-workshops/foundations-of-research-software-publication/astronaut-analysis-data/
[11] https://hifis/net

by the course participants. In the ”technical basis” dimension, the example shows only low levels, because it is only an example script for teaching and therefore is not to be fundamentally further developed. So it does not need e.g. security concept or defined code styles for external developers.

The evaluation task for this example shows some difficulties. The allocation to a level was not always completely clear or was viewed differently by different reviewers. There is a clear need for appropriate tools to provide an objective basis for this assessment. In the questionnaire, the dimensions have different numbers of attributes, with the ”Reusable” dimension standing out in particular, where only one attribute is assigned a corresponding level and therefore one attribute alone determines this dimension.

Comparing different types of published software

The evaluation of both examples was done as an in person interview. A visualization in a radar plot is shown in Figure 4. The plot also contains the currently defined minimum criteria as well as the maximum level that can be measured automatically. The comparison of both published software products shows that the quantization of quality aspects works as expected. The astronaut analysis as a small analysis script of good enough quality for that purpose receives lower scores in almost all dimensions of the improved indicator compared to the infrastructure software alpaka. The reusibility dimension is still missing one attribute in the current stage of the indicator, hence, the results in that dimension should not be taken as final for now.

As each interview with the authors of the software took about 30-60 minutes, it is clear that this is not an option for all published software of each Helmholtz center. Automated evaluation is a necessity and will also be an integral part of the implementation of the improved indicator.

5.3 Automation of quality assessment supported by suitable tools

The quality assessment should only lead to a minimal increase in resources required for the assessment itself. Therefore, the assessment of all quality dimensions and attributes should be automated as much as possible, the final assessment and visualization should be done in a digital manner. An initial internal evaluation of existing tools (such as HERMES[12], the Helmholtz Research Software Directory (RSD)[13] or the ESCAPE-OSSR onboarding procedures[14]) identified the quality attributes and maturity levels that can already be assessed automatically. Additionally, potential areas of enhancement of the existing tools to assess further quality attributes were evaluated and their effort estimated.

For research software the results from the proof of concept studies revealed a quite diverse landscape of practice. While most software development uses the same development platforms such as GitHub, GitLab.com or self-hosted GitLab instances as well as generic publication repositories such as Zenodo, the actually established practices and the fulfillment of the FAIR criteria differ quite significantly. This is mainly due to the fact that the actual development platforms are designed for all kinds of software in mind and in general are rather project management plat-

[12] https://github.com/softwarepub/hermes
[13] https://helmholtz.software
[14] https://projectescape.eu/services/open-source-scientific-software-and-service-repository-ossr

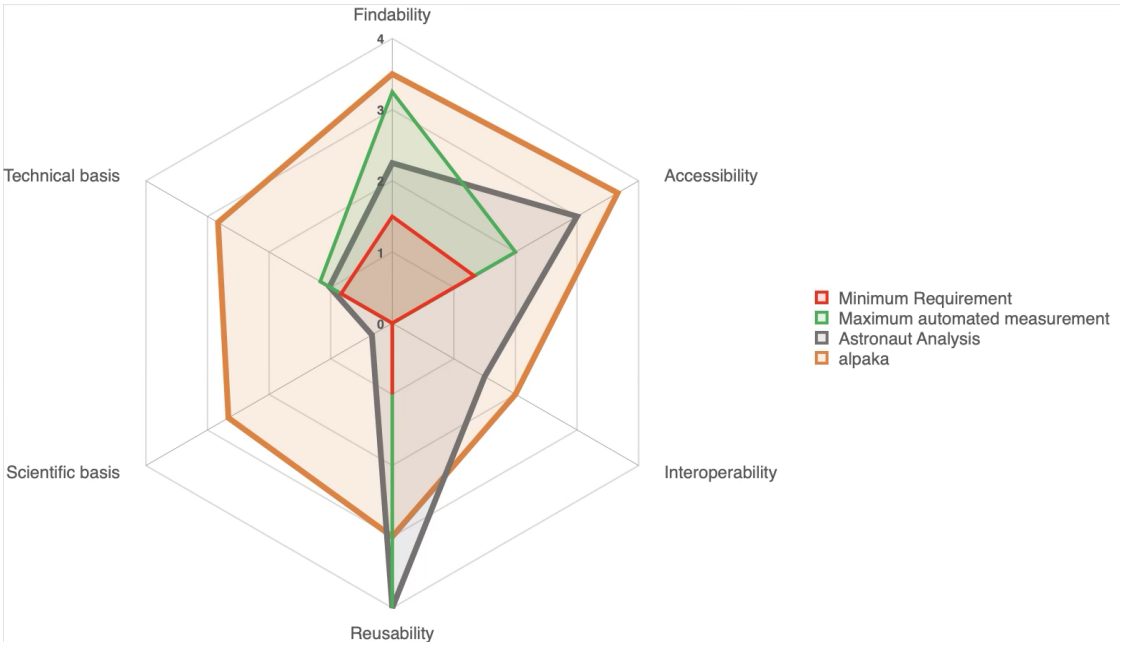

Figure 4: Radar plots for the two examples (alpaka and the austronaut analysis training code) as well as the currently defined minimum requirements and automatically measurable maxima. The two dimensions interoperability and scientific basis are currently not measurable, hence their minimum requirements are set to 0. This results in the slightly distorted visualization in the bottom half of the radar plot.

forms. As a result they do not incorporate the FAIR criteria into their required metadata of each development repository. Also generic publication repositories tend to fulfill only the minimal FAIR criteria, thus, also not enforcing specific standards towards research software quality.

Therefore, each published research software requires an individual evaluation with respect to the established dimension, attributes and maturity levels. Such an evaluation, however, is only feasible if it is fully automated and does not require any manual steps from the evaluators. At the same time, it is possible for the software authors to provide the data and metadata to be evaluated as part of their software development repositories. Special metadata files such as citation.cff[15] or codemeta.json[16] can contain the metadata for the published software. Tools such as HERMES [KMD+25] can harvest this structured information and, thus, provide a mechanism to evaluate research software. The modular structure of HERMES (see Figure 5) with its harvesting plugins, also allows to generate a deposit that contains all information that can currently be acquired for the quality indicator. Furthermore, Helmholtz operates the Helmholtz Research Software Directory (RSD) as a registry of research software of the Helmholtz Association. The RSD can be extended to also hold the data/metadata required for the assessment of the quality attributes defined for the here described attributes of the new quality indicator and can also accumulate

[15] https://citation-file-format.github.io
[16] https://codemeta.github.io

the reported KPI. The authors of a research software can then either manually provide answers to determine the maturity levels of all attributes as part of the onboarding of their software in the RSD or also rely on tools like HERMES to automatically import this information from their software development repositories.

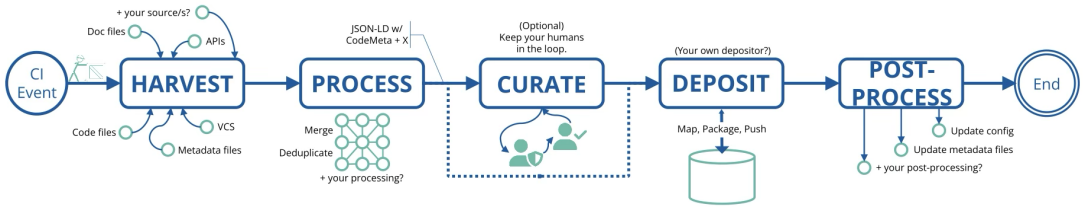

Figure 5: The HERMES workflow harvests (meta) data from the software development repository to generate a deposit of this data. This can also accumulate the information needed for the quality indicator. (Figure directly taken from [KMD+25])

The initial tool landscape also highlighted that not all attributes or maturity levels can currently be automatically evaluated. This is due either to a lack of standardization within the research communities themselves (e.g. the expectations of research communities themselves towards a research software product), a lack of metadata vocabularies to express certain aspects (e.g. documenting versioning schemes) or simply a lack of available tools for the evaluation (e.g. checking certain software development principles like peer-review of changes). Such attributes and maturity levels are currently not included in the indicator by either adapting the weights within a dimension to exclude an attribute or by setting the minimum requirement to 0. This is also reflected in the visualized maximum in Figure 4 that depicts the current evaluation of the task group on what maturity levels the available tools can measure.

The aggregation for each dimension but also for each Helmholtz Center is also fully automatable as it operates on the previously harvested data. Tools that generate the radar plots for each publication as well as generate a reportable KPI use the criteria established by the task group. As a result the KPI also delivers on the objective of a minimal overhead in generating the indicator numbers for the reporting centers.

6 Summary and outlook

Research software still lacks adequate recognition as an independent outcome of scientific work. To change this situation, the Helmholtz Association decided to including research software into its research assessment. An indicator has been developed to ensure the quality of a software that is counted in the KPI. The quality indicator has been designed to meet certain demands specified by the Helmholtz Association as outlined in the introduction of this paper.

It should create awareness and valuation to the diverse research output. First experiences with an initial quality indicator could show that including research software in research assessment increases the awareness and valuation for software; the reported numbers of software publications increased in almost all Helmholtz centers.

It should align the assessment of research and research practice to the conditions of openness of science. To address this demand a multidimensional quality indicator has been composed focusing on the FAIR principles. All quality dimensions have been characterized by measurable quality attributes.

It should be a stimulus and incentive for scientists to improve their software related to specific quality criteria. For this purpose maturity models are a suitable approach. They help scientists to develop their software with respect to certain quality goals. Inspired by existing models, the quality indicator defines the same number of maturity levels for all quality attributes. Radar plots are introduced to visualize the maturity levels.

The feasibility of the quality indicator was evaluated in various research communities by community engagement and running a couple of exemplifying assessments. Since the assessment of all quality dimensions and attributes should be automated as much as possible, appropriate tools were evaluated.

The quality indicator for published research software of the Helmholtz Association sets itself apart from other approaches: It creates a multi-dimensional abstract quality model that is related to specified goals and that captures aspects supporting these goals. The indicator describes ideal conditions for research software without any regard how this could be measured or even whether research communities have yet developed any guidelines on research software. The indicator is then mapped to what is currently measurable and how- not the other way round. In this aspect the new indicator is also different from other initiatives, e.g. from within the EOSC, that provide more of a checklist of required and optional characteristics instead of the multi-dimensional and maturity level driven approach. The task group developing the presented indicator fully acknowledges that not all attributes and maturity levels can currently be measured. Nevertheless, the indicator provides a quickly usable solution that can be established with what is currently achievable and its abstract quality model enables a future implementation of then established mappings to methods for measuring the attributes.

This approach could also be taken up by other research institutions as the main data source for the indicator is what the authors of research software provide in their development repositories. It does, however, require a clear mapping of software authors to reporting institutions, e.g. via persistent identifiers such as RORs[17] for organizations as part of the software metadata.

Future work in the context of the new indicator will have to cover three areas: 1. accompanying the introduction in the regular reporting of the Helmholtz Association by documenting frequently asked questions as well as cross-checking the validity of the reported results, 2. working on improving the tools to measure the attributes and maturity levels, e.g. by contributing to standardized ways for software authors to express this information in the metadata of their software development repositories, and 3. working with the scientific communities on establishing and documenting good practices and minimum requirements towards research software in their domain.

With the quality indicator for research software, the Helmholtz Association made a step forward to a more comprehensive assessment of scientific output, to more openness with respect to the FAIR principles, and to higher valuation of the work of scientists and software engineers on

[17] https://ror.org

well-developed code and the necessary quality assurance. The Helmholtz wide implementation of the improved quality indicator follows the test phase with the initial indicator (2023-2025) and will start in 2026. Then, all Helmholtz Centers have to apply the concept of the indicator in their annual research assessment that has to be prepared within the Helmholtz association, and report -besides on research papers- also on research software. Suitable workflows to gather values for determined quality attributes and to derive the KPI will be established in the centers. The implementation is shaped as an iterative process with adjustments of workflows and quality descriptions derived from practical experience.

Acknowledgements: The approach presented here reflects the results of various discussions and consultations in the Task Group Helmholtz Quality Indicators for Data and Software Products of the Helmholtz Association. The authors would particularly like to thank Wolfgang zu Castell for his many ideas and Tobias Schlauch and René Widera for evaluating the examples presented.

Bibliography

[BCK+22] 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. *Sci Data* 9:622, 2022. doi:10.1038/s41597-022-01710-x

[CBA+23] N. Chue Hong, E. Breitmoser, M. Antonioletti, J. Davidson, D. Garijo, A. Gonzalez-Beltran, M. Gruenpeter, R. Huber, C. Jonquet, M. Priddy, J. Shepeherdson, M. Verburg, C. Wood. D5.2 - Metrics for automated FAIR software assessment in a disciplinary context. Zenodo, Oct. 2023. doi:10.5281/zenodo.10047401

[CHM09] W. Curits, W. Hefley, S. Miller. *People CMM: A Framework for Human Capital Management*. Addison-Wesley Longman, Amsterdam, second edition, 2009.

[CKH21] K. Champion, S. Khatri, B. M. Hill. Qualities of Quality: A Tertiary Review of Software Quality Measurement Research. arXiv, 2021. https://arxiv.org/abs/2107.13687

[Com22] A. R. D. Commons. A National Agenda for Research Software. Mar. 2022. doi:10.5281/zenodo.6378082

[Cus20] J. J. Cusick. A Survey of Maturity Models from Nolon to DevOps and Their Applications in Process Improvement. 2020. https://arxiv.org/abs/1907.01878

[DBM+24] Deekshitha, R. Bakhshi, J. Maassen, C. M. Ortiz, R. van Nieuwpoort, S. Jansen. RSMM: A Framework to Assess Maturity of Research Software Project. arXiv, 2024. https://arxiv.org/abs/2406.01788

[DCG+24] M. David, M. Colom, D. Garijo, L. J. Castro, V. Louvet, E. Ronchieri, M. Torquati, L. del Caño, S. H. Leong, M. Van den Bossche, I. Campos, R. Di Cosmo. Task Force Sub Group 3 - Review of Sooftware Quality Attributes and Characteristics. Zenodo, Feb. 2024.
doi:10.5281/zenodo.10647227

[Deu19] Deutsche Forschungsgemeinschaft. Guidelines for Safeguarding Good Research Practice - Code of Conduct. Oct. 2019.
https://zenodo.org/records/14281892

[DFM+23] Deekshitha, S. Farshidi, J. Maassen, R. Bakhshi, R. Van Nieuwpoort, S. Jansen. FAIRSECO: An Extensible Framework for Impact Measurement of Research Software. In *2023 IEEE 19th International Conference on e-Science (e-Science)*. Pp. 1–10. Academic Press, 2023.
doi:10.1109/e-Science58273.2023.10254664

[Eur24] European Commission. Action Plan by the Commission to implement the ten commitments of the Agreement on Reforming Research Assessment (ARRA). 2024.
https://research-and-innovation.ec.europa.eu/document/download/e69aff11-4494-4e5f-866c-694539a3ea26_en?filename=ec_rtd_commitments-reform-research-assessment.pdf

[FA24] B. Fritzsch, M. Andrés-Martínez. Do you know what researchers do and what they want? Experiences from a survey and user interviews. Zenodo, 2024. derse24 - Conference for Research Software Engineering in Germany, Würzburg.
doi:10.5281/zenodo.10788178

[GKLe21] M. Gruenpeter, D. S. Katz, A.-L. Lamprecht, et al. Defining research software: a controversial discussion. zenodo, 2021.
10.5281/ZENODO.5504016

[Hum88] W. Humphrey. Characterizing the software process: a maturity framework. *IEEE Software* 5(2):73–79, 1988.
doi:10.1109/52.2014

[Ire90] L. R. Ireland. International standards on quality. *PM Network* 4(7):41, 1990.
https://www.pmi.org/learning/library/quality-international-standards-3527

[KMD+25] S. Kernchen, M. Meinel, S. Druskat, M. Fritzsche, D. Pape, O. Bertuch. Extending and applying automated HERMES software publication workflows. *Electronic Communications of the EASST* 83, Feb. 2025.
doi:10.14279/eceasst.v83.2624

[KP96] B. Kitchenham, S. Pfleeger. Software quality: the elusive target [special issues section]. *IEEE Software* 13(1):12–21, 1996.
doi:10.1109/52.476281

[KV15] S. Krishnamurthi, J. Vitek. The real software crisis: repeatability as a core value. *Commun. ACM* 58(3):34–36, Feb. 2015.
doi:10.1145/2658987

[NE21] D. Nüst, S. Eglen. CODECHECK: an Open Science initiative for the independent execution of computations underlying research articles during peer review to improve reproducibility. *F1000Research* 10(253), 2021.
doi:10.12688/f1000research.51738.2

[NNR19] P. Nistala, K. V. Nori, R. Reddy. Software Quality Models: A Systematic Mapping Study. In *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*. Pp. 125–134. 2019.
doi:10.1109/ICSSP.2019.00025

[PVN+23] L. Pinedo, M. Valles-Coral, J. R. Navarro-Cabrera, R. Injante, A. Cárdenas-García, F. Ruiz-Saavedra, C. García-Rivas-Plata, C. A. Flores-Tananta. Software quality models: Exploratory review. *EAI Endorsed Transactions on Scalable Information Systems* 10(6), Sept. 2023.
doi:10.4108/eetsis.3982

[RH68] R. J. Rubey, R. D. Hartwick. Quantitative measurement of program quality. In *Proceedings of the 1968 23rd ACM National Conference*. ACM '68, p. 671–677. Association for Computing Machinery, New York, NY, USA, 1968.
doi:10.1145/800186.810631

[SC19] J. B. S. dos Santos-Neto, A. P. C. S. Costa. Enterprise maturity models: a systematic literature review. *Enterprise Information Systems* 13(5):719–769, 2019.
doi:10.1080/17517575.2019.1575986

[SHM18] T. Schlauch, C. Haupt, M. Meinel. Software Engineering Guidelines - From Theory To Practice. zenodo, 1018.
10.5281/ZENODO.1411431

[Tas25] Task Group Helmholtz Quality Indicators for Data and Software Products. A Framework for Assessing Research Data and Software Publications – The Helmholtz Quality Indicator. 2025.
doi:10.48440/os.helmholtz.085

[WDAa16] M. Wilkinson, M. Dumontier, I. Aalbersberg, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* 3:160018, 2016.
doi:10.1038/sdata.2016.18
http://www.example.org/gnats-and-gnus/

[ZWW+16] E. Zenker, B. Worpitz, R. Widera, A. Huebl, G. Juckeland, A. Knüpfer, W. E. Nagel, M. Bussmann. Alpaka – An Abstraction Library for Parallel Kernel Acceleration. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Pp. 631–640. 2016.
doi:10.1109/IPDPSW.2016.50