Article Type: Software Engineering Track

Multi-Dimensional Research Software Categorization

Wilhelm Hasselbring, *Software Engineering, Kiel University, Kiel, 24098, Germany*
Stephan Druskat, *German Aerospace Center (DLR), Berlin, 12489, Germany*
Jan Bernoth, *University of Potsdam, Potsdam, 14476, Germany*
Philine Betker, *Department for Epidemiology, Helmholtz Centre for Infection Research, Brunswick, Germany*
Michael Felderer, *German Aerospace Center (DLR) & University of Cologne, Cologne, 51147 , Germany*
Stephan Ferenz, *Department of Computer Science, Carl von Ossietzky Universität Oldenburg, 26129 Oldenburg*
Ben Hermann, *Secure Software Engineering Group, TU Dortmund, 44227 Dortmund*
Anna-Lena Lamprecht, *University of Potsdam, Potsdam, 14476, Germany*
Jan Linxweiler, *TU Braunschweig, Braunschweig, 38106, Germany*
Arnau Prat, *German Aerospace Center (DLR), Braunschweig, 38108, Germany*
Bernhard Rumpe, *Software Engineering, RWTH Aachen University, Germany*
Katrin Schoening-Stierand, *Hub of Computing and Data Science, University of Hamburg, 22761 Hamburg*
Shinhyung Yang, *Software Engineering, Kiel University, Kiel, 24098, Germany*

Abstract—Research software has been categorized in different contexts to serve different goals. We start with a look at what research software is, before we discuss the purpose of research software categories. We propose a multi-dimensional categorization of research software. We present a template for characterizing such categories. As selected dimensions, we present our proposed role-based, readiness-based, developer-based, and dissemination-based categories. Since our work has been inspired by various previous efforts to categorize research software, we discuss them as related works. We characterize all these categories via the previously introduced template, to enable a systematic comparison. We report on the multi-dimensional categorization of selected research software examples.

***Keywords**: Research Software, Software Categorization, Technology Readiness Level, Open Source Software*

Research software is software that is designed and developed to support research activities. Research software is developed by researchers themselves or by software engineers working closely with researchers. Research software is typically developed to meet specific research needs, and often has unique requirements that are different from standard commercial software [1]. However, research software is gaining appreciation and endorsement for research and as a research result itself.

XXXX-XXX © 2024 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

Research Software Engineering (RSE) is a specialized field that applies software engineering principles to address the unique challenges posed by developing software for scientific and academic research, with the goal of enhancing the efficiency, reproducibility, and impact of research outcomes. Research software engineers specialize in developing and maintaining software for research purposes.

In this paper, we propose a multi-dimensional categorization of research software, along the dimensions of roles, readiness, developer, and dissemination. We start with a look at what research software is before we discuss the purpose of research software categories. We present a template for characterizing such categories. Subsequently, our proposed role-based, readiness-based, developer-based, and dissemination-based categories are presented. Our work has been inspired by various previous efforts to categorize research software, which we discuss as related works. We characterize all these categories via the previously introduced template, and conclude with an outlook to future work.

Research Software

For the purposes of this paper, we follow the FAIR for Research Software (FAIR4RS) Working Group in their definition of research software, as software that was created during the research process or for a research purpose [2]. This definition distinguishes “research software” and “software in research,” which includes general purpose software. The software components (e.g., operating systems, programming languages, libraries, etc.) that are used for research but were *not* created during research or with a clear research intent should be considered “software in research” and not “research software.” In the present paper, we categorize research software.

Purpose of Research Software Categories

We envision the following benefits from using categories for research software, which may serve

- as a basis of institutional guidelines and checklists for research software development;
- to better understand the different types of research software and their specific quality requirements;
- to recommend appropriate software engineering methods for the individual categories;
- to design appropriate teaching / education programs for the individual categories;
- to give stakeholders (especially research software engineers and their management) a better understanding of what kind of software they develop;
- for a better assessment of existing software when deciding to reuse it;
- for research funding agencies, to define appropriate funding schemes;
- to define appropriate metadata labels for FAIR research software;
- in RSE Research [3] to provide a framework for classifying research software artifacts.

This list is not exhaustive.

Characterization of Research Software Categories

Categorizations can be described through their scope, purpose, context, properties, consequences for creation and use, and their inter-categorial relations. Table 1 provides a template for systematically describing the characteristics of research software categorizations, which we will use later to characterize some individual categorizations in the subsequent sections.

Role-Based Categorization of Research Software

Research software can be used to collect, process, analyze, and visualize data, as well as to model complex phenomena and run sophisticated simulations. Research software is also developed to control and monitor lab experiments and environmental observations. In engineering research, research software constitutes a new paradigm of scientific inquiry next to theory and experiment and acts as a proof-of-concept to invent and evaluate new technological artifacts, including algorithms, methods, systems, tools, and other computer-based technologies. Research software also provides the infrastructure to manage, publish, and archive research data and software.

Thus, research software may take various *roles* in the research process [4]. This is similar to software engineering teams, which involve a range of roles that contribute to the development, maintenance, and improvement of software systems. Some common roles in software engineering are software architect, programmer, and tester. Each role may be taken by several persons, and one person may take several roles. These role assignments may also change during a software project.

We propose a similar role-based categorization of research software, with an emphasis on varying quality requirements for the different *roles* that software may take in research. Accordingly, a research software may take several roles, which may also change during the life cycle of the software.

Research software mainly falls into one of the following three top-level role categories (and sometimes combinations):

1) *Modeling, Simulation, and Data Analytics* of, e.g., physical, chemical, social, linguistic, or biological processes in spatio-temporal contexts.

Criterion	Explanation
Scope	What is the scope of the categorization?
Purpose	What is the purpose of the categorization?
Context	In which contexts are specific categories developed and used?
Properties	What are specific properties of the different categories?
Consequences for Creation	How is and should software of a specific category be developed?
Consequences for Use	How and why is software of a specific category used? What are the differences between the categories in terms of use and reuse, including, e.g., in software publication & citation?
Inter-categorial relations	What are the relations between different categories?

TABLE 1. Template for describing criteria of research software categorizations.

2) *Technology Research Software* in science and engineering research.
3) *Research Infrastructure Software*, such as research data and software management systems.

The assignment of research software to categories may evolve over time. For instance, software specifically developed for a research question (usually Categories 1 & 2) can later turn into infrastructure software (Category 3). In different contexts, a software may also be in multiple categories at the same time.

We further refine Category 1 research software for modeling, simulation, and data analytics with several subcategories:

1.1) Modeling and simulation (e.g., numerical modeling, agent-based modeling)
1.2) Data analytics, on observation and simulation data, with statistical analysis and machine learning as methods
1.3) Software analytics (static, dynamic, evolution, repository mining)
1.4) Integrative analysis (data assimilation and decision analysis)
1.5) Scientific visualization

Category 2) for technology research software is used in structural sciences (mathematics and computer science) and in engineering sciences (software, electrical, mechanical, and civil engineering). Technology research software may be related to target contexts:

2.1) Hardware (usually as embedded software)
2.2) Software (e.g., as part of an operating system)
2.3) Human (with a user interface)
2.4) Process (e.g., as part of a business, development or production processes)

Again, one research software may be in multiple categories. In the next section, we will additionally relate this category to technology readiness levels as secondary sub roles.

We further refine Category 3 for research infrastructure software with several subcategories:

3.1) Control and monitoring software for complex experiments and instruments. This includes embedded control software, as well as native and web-based monitoring software.
3.2) Data collection and generation (survey software, sensor-based data collection, synthetic data generation, etc.).
3.3) Pipelines and tools.
3.4) Libraries, for instance for high performance computing.
3.5) Laboratory notebooks.
3.6) Data management.
3.7) Software management.
3.8) Collaboration and publication.

These categories have varying requirements on their software development. For instance, dedicated requirements engineering may be relevant for Category 3), but not for Category 1). As another example, safety analysis may be relevant for Category 3.1), but not for Categories 1) and 2).

Figure 1, left, shows our resulting role-based categorization.

Table 2 characterizes our multi-dimensional categorization in terms of the template in Table 1. The readiness-based, developer-based, and dissemination-based categorizations are introduced in the following three sections, before we discuss some related categorizations.

Readiness-Based Categorization of Research Software

Technology is the application of conceptual knowledge for achieving practical goals, especially in a reproducible way. The word *technology* can also mean the products resulting from such efforts, including both tangible tools such as utensils or machines, and intangible ones such as software.

Technology readiness levels (TRLs) are a method for estimating the maturity of technologies. TRLs enable consistent and uniform discussions of technical maturity across different types of technology. Figure 1, right, shows the resulting readiness-based categorization with the titles of the European TRL 1 to TRL 9 [5].

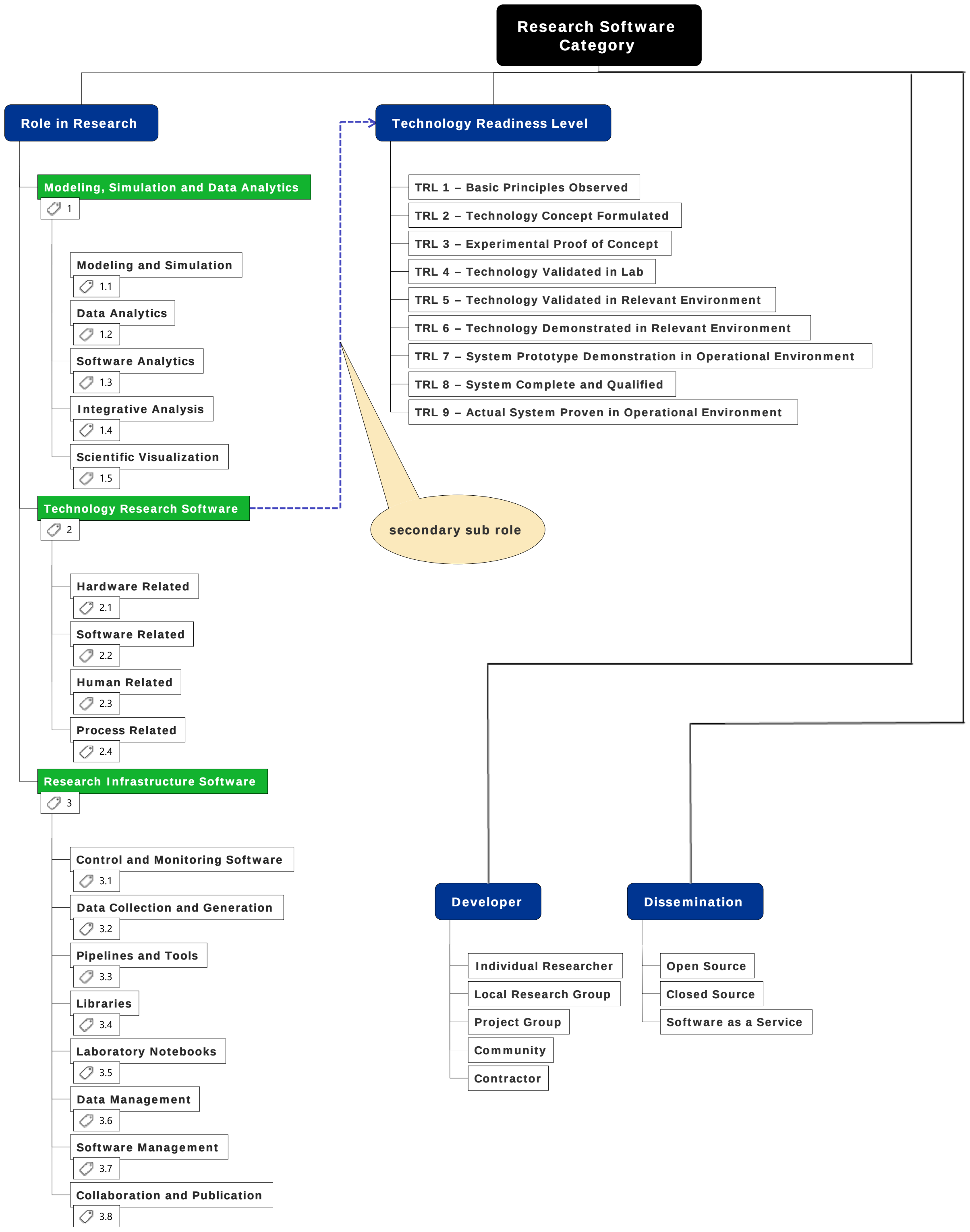
Research Software Category
Role in Research
Modeling, Simulation and Data Analytics
1
Modeling and Simulation
1.1
Data Analytics
1.2
Software Analytics
1.3
Integrative Analysis
1.4
Scientific Visualization
1.5
Technology Research Software
2
Hardware Related
2.1
Software Related
2.2
Human Related
2.3
Process Related
2.4
Research Infrastructure Software
3
Control and Monitoring Software
3.1
Data Collection and Generation
3.2
Pipelines and Tools
3.3
Libraries
3.4
Laboratory Notebooks
3.5
Data Management
3.6
Software Management
3.7
Collaboration and Publication
3.8
Technology Readiness Level
TRL 1 – Basic Principles Observed
TRL 2 – Technology Concept Formulated
TRL 3 – Experimental Proof of Concept
TRL 4 – Technology Validated in Lab
TRL 5 – Technology Validated in Relevant Environment
TRL 6 – Technology Demonstrated in Relevant Environment
TRL 7 – System Prototype Demonstration in Operational Environment
TRL 8 – System Complete and Qualified
TRL 9 – Actual System Proven in Operational Environment
secondary sub role
Developer
Individual Researcher
Local Research Group
Project Group
Community
Contractor
Dissemination
Open Source
Closed Source
Software as a Service

FIGURE 1. Our multi-dimensional categorization of research software, along the dimensions of roles, readiness, developers, and dissemination.

Criterion	Explanation
Scope	This categorization covers the dimensions of roles, readiness, developers, and dissemination.
Purpose	The categorization aims to enable a better understanding of the different types of research software and their specific quality requirements.
Context	The categorization has been produced in the context of a task force of the special interest group on Research Software Engineering, within the German Association of Computer Science (GI e.V.) and the German Society for Research Software (de-RSE e.V.). It is meant to serve different purposes, in particular RSE research [3].
Properties	The categories follow different relevant dimensions, and are defined collaboratively among software engineering researchers and research software engineers.
Consequences for Creation	Depending on its category, software is expected to meet different quality requirements and follow different development processes.
Consequences for Use	Perceive that there are many different types of research software, fulfilling many different roles and functions.
Inter-categorial relations	Individual research software may change its category within one or more dimensions.

TABLE 2. Characteristics of our multi-dimensional categorization for research software.

These TRLs may be applied to all types of research software, thus, the category dimensions are *orthogonal*: every research software may be classified independently in each dimension.

In addition, for technology research software, these TRL titles can be read as *secondary* sub roles. Examples are:

TRL 3 : The technology research software takes the role as an "Experimental Proof of Concept" within some research project.

TRL 4 : The technology research software takes the role as a "Technology Validated in Lab" within some research project.

Thus, the TRLs constitute *sub roles* of technology research software.

One specific technology research software may take several such sub roles over its lifecyle, with increasing "readiness". It may also take several roles at the same time, within different contexts: In one project, it may serve as experimental proof of concept (TRL 3); in another project, it may already serve as a technology validated in a lab (TRL 4). Eventually, a technology research software may even become an "Actual System Proven in Operational Environment" (TRL 9).

"Readiness" is top-level in the mindmap, thus it is it own dimension. If we had put "Readiness" directly below "Technology Research Software", it would not be its own dimension, thus we added the cross-link from "Technology Research Software" to illustrate the additional, secondary sub-role relationship.

The difference between the categories "Modeling and Simulation" and "Technology Research Software" (without consideration of the TRL sub roles) may be illustrated, for instance, with control engineering research:

- As a control engineering researcher, you may build a *simulation* of a control system.
- As a control engineering researcher, you may also build an *actual* control system as a new software system. In an automation lab, this researcher may then experiment with this system (not with the simulation of the system). If this system (which is a technology research software) matures, it may reach higher TRLs.

Here, both, the simulation and the actual control system are research software.

Developer-Based Categorization of Research Software

For the developer dimension, we see the following stages for research software:

1) Individual Researcher, such as PhD student, PostDoc, or Research Software Engineer.
2) Local Research Group.
3) Project Group, in which several research groups may collaborate.
4) Community on a specific research topic.
5) Contractor (professional software company developing the software on behalf of researchers).

Dissemination-Based Categorization of Research Software

A community or contractor may develop the software open-source, closed-source, or it may provide research software as an online service.

Figure 1, bottom, shows our developer-based and dissemination-based categorizations.

Related Research Software Categories

Research software has been categorized in different contexts to serve different aims. Some of them are discussed here as related works, as they a) represent a good starting point for a discussion on research software categorization, b) provided significant input to our work, and c) may be used to compare and assess our categorization. We characterize these categories via the previously introduced template in the supporting technical report [6].

Role-Based Categorization

Van Nieuwpoort and Katz [4] present a role-based categorization. They categorize research software as an integral component of instruments used in research, as the instrument itself, for analyzing research data, for presenting research results, for assembling or integrating existing components, as infrastructure or an underlying tool, and for facilitating research-oriented collaboration. This categorization inspired our work. Based on discussions with the authors of the present paper, van Nieuwpoort and Katz extended their categorization with our *Technology Research Software* category [4]

Maturity-Based Categorization

In their National Agenda for Research Software [7], the Australian Research Data Commons – an Australian research data infrastructure facility – argue for research software to be recognized as a first-class output of research. They describe a three-level maturity categorization of research software that is related to our readiness dimension:

1) *Research Data Processes* captured as software. The result is analysis code that captures research processes and methodology: the steps taken for tasks like data generation, preparation, analysis, and visualization.
2) *Novel Methods and Models* captured as software. The results are prototype tools that demonstrate a new idea, method, or model for research.
3) *Accepted Methods and Models* captured as software. The result can become research software infrastructure that captures more broadly accepted and used ideas, methods, and models for research.

Each category faces specific challenges with regard to recognition, from making research practice transparent, to creating impact through quality software and safeguarding longer-term maintenance.

Application classes in institutional software engineering guidelines

Institutional guidelines typically define so-called application classes for research software, which require appropriate quality properties, and, thus software engineering methods [8]:

› For software in Application Class 0, the focus is on personal use in conjunction with a small scope.
› For software in Application Class 1, it should be possible, for those not involved in the development, to use it to the extent specified and to continue its development.
› For software in Application Class 2, it is intended to ensure long-term development and maintainability. It is the basis for a transition to product status.
› For software in Application Class 3, it is essential to avoid errors and to reduce risks. This applies in particular to critical software.

The application classes relate to our readiness domain and to some extent to our developer-based categorization.

EOSC Research Software Lifecycle

The European Open Science Cloud (EOSC) aims to create a virtual environment for sharing and accessing research data across borders and scientific disciplines. The SubGroup 1 “On the Software Lifecycle” of the EOSC Task Force “Infrastructure for quality research software” provides a categorization for software in the research lifecycle [9]:

1) Individual creating research software for own use (e.g. a PhD student).
2) A research team creating an application or workflow for use within the team.
3) A team / community developing (possibly broadly applicable) open source research software.
4) A team or community creating a research service.

This categorization is covered by our developer-based categorization.

Computational research in the earth system sciences

Döll et al. [10] provide recommendations for sustainable research software for high-quality computational research in the Earth System Sciences, and categorize this research software as follows:

› Simulation of Earth system processes by Earth system models.
› Design, processing and analysis of Earth observation and lab experiment data.

› Integrative analysis of simulation models, large data bases, and stakeholder knowledge.

These categories correspond to our role-based categories 1.1), 1.2), and 1.4), respectively.

Categorizing the Software Stack

Another dimension is the research software stack, from non-scientific infrastructure, scientific infrastructure, discipline-specific software, up to project-specific software [11]. This dimension could be the basis for another branch in our multi-dimensional categorization.

Qualitative Evaluation

As a pre-review study, we conducted a multi-dimensional categorization of selected research software examples, to check whether we can categorize selected research software in multiple dimensions. The selection is mainly based on in-depth knowledge of the respective research software by the authors, such that we are able to confidently categorize these research software examples, in particular the readiness level. In Table 3, we categorize the *Hexatomic* framework for multi-layer linguistic annotation of corpora (https://corpus-tools.org/hexatomic/), the *Kieker* observability and monitoring framework (https://kieker-monitoring.net/), the *MontiCore* framework for the development of software languages (http://monticore.github.io/), the Prospective Monitoring and Management App - *PIA* (https://info-pia.de/), and a quantum optics control software. Due to length limits, we refer to the extended version of this magazine article for further details [6].

Our qualitative evaluation shows that it is possible to categorize different research software along multiple categories. In particular, it shows that our categorization is applicable to research software independently of a single dimension: we successfully categorized software at different maturity levels, developed by different actors, and disseminated through different means. We expect that our categorization can significantly contribute to categorizing research software. It increases coverage over existing approaches to categorization by adding the dissemination category and integrating:

- role-based categorization [4], [10] in our role categories;
- maturity-based categorization [7], [8] in our readiness categories;
- lifecycle-based categorization [9] in our developer categories;

In our evaluation, example research software has been categorized with 1–5 roles. This shows a high precision to cover different roles research software can take in different contexts, while manifesting that research software roles are not exclusive. While *PIA*, for example, serves a single purpose within a single context, *Hexatomic* can be used for different subtasks in different data-centric application contexts. As infrastructure software that can be used to integrate tools into a pipeline, Hexatomic combines research-related tasks such as data generation and integration with research tasks such as data editing and analysis. Simultaneously, it is technology research software whose target system is an existing ecosystem of software tools for linguistic research. The Hexatomic example reveals a property of research software that is central to our argument, i.e., that different contexts and perspectives put software into different roles, which makes a multi-dimensional categorization necessary.

As future work, we intend to conduct more in-depth quantitative research into our categorization to assess and improve its granularity and precision. Based on this, we intend to analyze relations and correlations between categorical dimensions. We also plan to widen the corpus of categorized research software by asking more members of the RSE community to categorize their own research software. In particular, the assessment of the readiness levels requires a profound knowledge of the software and its use. To quantitatively evaluate our categorization scheme, we intend to apply more systematic and replicable research via a systematic literature review of published research software [12].

Additional research into providing methodological guidance for researchers to consistently replicate our classifications, especially for more subjective aspects like technology readiness levels, could offer clear decision criteria and documentation protocols to support the application of our framework.

Conclusion

We categorize research software along various dimensions, contributing to fostering effective development, recognition, and utilization of research software within the research community. One essential use case of this categorization is its incorporation into forthcoming guidelines for research software development. As we classify research software, we enable tailoring guidelines to specific classes, offering developers a structured framework that aligns with each category's unique requirements and challenges. The multi-dimensional categorization of selected research software examples stimulated the refinement and strengthening of our categorization.

Moreover, the categorization is intended to be a

Software	Role	Readiness	Developer	Dissemination
Hexatomic	1.2 Data Analytics 1.4 Integrative Analysis 2.2 Software Related 3.2 Data Collection and Generation 3.3 Pipelines and Tools	TRL 4	Local Research Group	Open Source
Kieker	1.3 Software Analytics 2.2 Software Related	TRL 4, TRL 5, TRL 6	Community	Open Source
MontiCore	1.1 Modeling and simulation 2.2 Software Related 3.3 Pipelines and Tools	TRL 4 – TRL 8	Community	Open Source, Software as a Service
PIA	3.2 Data Collection and Generation	TRL 9	Contractor	Open Source
Quantum Optics Control Software	2.2 Software Related 3.1 Control and Monitoring Software 3.2 Data Collection and Generation	TRL 9	Project Group	Closed Source

TABLE 3. Exemplary multi-dimensional categorization of research software, further details in [6].

valuable tool for stakeholders, especially research software engineers and their group, chair, department, or institute leaders. The categorization may provide these individuals with a better understanding of the software they are developing, offering insights into its nature, purpose, and potential impact. This knowledge is essential for informed decision-making, adequate resource allocation, and strategic planning within research institutions.

Recognition for research software engineers is another outcome we anticipate from categorizing research software. By delineating different types of software and acknowledging the diverse skill sets required for their development and maintenance, our categorization aims to contribute to elevating the status of research software engineers. We hope this recognition motivates individuals and fosters a culture that values and appreciates the crucial role played by software in advancing research efforts.

Categorizations may also help assess external software when considering its use. We envision that it contributes to a standardized framework for evaluating software's relevance, applicability, and quality, facilitating informed decisions in adopting tools from different sources.

The categorization may become particularly valuable in allocating project-based or permanent funding. It can help researchers and developers clearly articulate their software's significance in a funding proposal. We envision this classification providing a framework that helps researchers and funding agencies.

Additionally, the categorization may help to emphasize which software is critical, highlighting the importance of its maintenance and continued development for its continued functionality. By highlighting this importance, we seek to contribute to an enhanced awareness of the ongoing support and resources required to ensure the longevity and sustainability of research software.

In the realm of Research Software Engineering (RSE) research [3], we hope that the categorization provides a framework for classifying research objects, supporting software corpus analyses, and enhancing our understanding of the different types of research software and their properties. This structured approach may aid in organizing and interpreting the vast landscape of research software, contributing to advancements in RSE methodologies and practices.

REFERENCES

1. A. Johanson and W. Hasselbring, “Software engineering for computational science: Past, present, future,” *Computing in Science & Engineering*, vol. 20, no. 2, pp. 90–109, Mar. 2018. doi: 10.1109/MCSE.2018.021651343
2. M. Gruenpeter, D. S. Katz, A.-L. Lamprecht, T. Honeyman, D. Garijo, A. Struck *et al.*, “Defining Research Software: A controversial discussion,” *Zenodo*, Sep. 2021. doi: 10.5281/zenodo.5504016
3. M. Felderer, M. Goedicke, L. Grunske, W. Hasselbring, A.-L. Lamprecht, and B. Rumpe, “Investigating research software engineering: Toward RSE Research,” *Communications of the ACM*, vol. 68, no. 2, Feb. 2025. doi: https://doi.org/10.1145/3685265
4. R. van Nieuwpoort and D. S. Katz, “Defining the

roles of research software (Version 2)," *Upstream*, Jul. 2024. doi: 10.54900/xdh2x-kj281

5. A. D. Rose, M. Buna, C. Strazza, N. Olivieri, T. Stevens, L. Peeters, and D. Tawil-Jamault, "Technology readiness level: guidance principles for renewable energy technologies," *European Commission, Directorate General for Research and Innovation*, 2017. doi: 10.2777/577767
6. W. Hasselbring, S. Druskat, J. Bernoth, P. Betker, M. Felderer, S. Ferenz, B. Hermann, A.-L. Lamprecht, J. Linxweiler, A. Prat, B. Rumpe, K. Schoening-Stierand, and S. Yang, "Multi-dimensional categorization of research software with examples," *Zenodo*, 2025. doi: 10.5281/zenodo.14082553
7. Australian Research Data Commons, "A National Agenda for Research Software," *Zenodo*, Mar. 2022. doi: 10.5281/zenodo.6378082
8. T. Schlauch, M. Meinel, and C. Haupt, "DLR Software Engineering Guidelines," *Zenodo*, Aug. 2018. doi: 10.5281/zenodo.1344612
9. G. Courbebaisse, B. Flemisch, K. Graf, U. Konrad, J. Maassen, and R. Ritz, "Research software lifecycle," *Zenodo*, Sep. 2023. doi: 10.5281/zenodo.8324828
10. P. Döll, M. Sester, U. Feuerhake, H. Frahm, B. Fritzsch, D. C. Hezel *et al.*, "Sustainable research software for high-quality computational research in the Earth system sciences: Recommendations for universities, funders and the scientific community in Germany," *FID GEO*, Feb. 2023. doi: 10.23689/fidgeo-5805
11. K. Hinsen, "Dealing With Software Collapse," *Computing in Science Engineering*, vol. 21, no. 3, pp. 104–108, May 2019. doi: 10.1109/MCSE.2019.2900945
12. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Systematic literature studies," in *Experimentation in Software Engineering*, 2nd ed. Springer, 2024, pp. 51–63. doi: 10.1007/978-3-662-69306-3_4

Wilhelm Hasselbring is a professor of software engineering at Kiel University, Germany, and an adjunct professor at the University of Southampton, UK. His research interests include software engineering, distributed systems, and open science. Hasselbring received a PhD in computer science from the University of Dortmund. He is a member of the ACM, IEEE Computer Society, and the German Association for Computer Science, at which he is vice chair of the special interest group on research software engineering. Contact him at hasselbring@email.uni-kiel.de.

Stephan Druskat is a software engineering researcher at the German Aerospace Center (DLR), and a Fellow of the Software Sustainability Institute (UK). He co-founded, and served on the inaugural board of, de-RSE – Society for Research Software. His research interests include research software sustainability, software citation and publication, and empirical research software engineering. He is a member of the Society for Research Software Engineering, the German Society for Research Software, and the German Association for Computer Science, where he co-founded the special interest group on research software engineering. Contact him at stephan.druskat@dlr.de.

Jan Bernoth is a researcher at the University of Potsdam, affiliated with the Chair of Complex Multimedia Application Architectures. His main research focus is on designing an architecture of the infrastructure to support Research Data and Software Management within a National Research Data Infrastructure Germany consortium NFDIxCS. Additionally, he investigates in his PhD thesis the creation of a research environment aimed at investigating analytical dashboards for science communication. Contact him at jan.bernoth@uni-potsdam.de.

Philine Betker is a researcher at the Helmholtz Centre for Infection Research, Brunswick, Germany. She is funded as part of a subproject of NAKO by the Federal Ministry of Education and Research (BMBF, project funding reference number 01ER2301/12), and the Helmholtz Association, with additional financial support by the participating universities and the institutes of the Leibniz Association. Contact her at philine.betker@helmholtz-hzi.de.

Michael Felderer is the director of the Institute of Software Technology at German Aerospace Center (DLR) and a full professor in the Department of Computer Science at the University of Cologne. His research interests include software quality and security, software engineering for AI, quantum computing and digital twin technologies as well as empirical and research software engineering. Contact him at michael.felderer@dlr.de.

Stephan Ferenz is a senior researcher at the Carl von Ossietzky Universität Oldenburg, Germany. His research interests are research data management, research software engineering, research software metadata. He holds a Master's degree in Electrical Engineering from the Leibniz Universität Hannover, Germany. Stephan Ferenz is a member of the German Association for Computer Science. Contact him at stephan.ferenz@uol.de.

Ben Hermann is a professor for secure software engineering at TU Dortmund University. His research is focused on static program analysis and metascience. Especially, his work on research artifact evaluation (i.e. peer review) have caught much attention. He received his PhD in computer science from TU Darmstadt. Contact him at ben.hermann@cs.tu-dortmund.de.

Anna-Lena Lamprecht is professor of software engineering at the University of Potsdam, Germany. She focuses on research software engineering (RSE) and is particularly known for her work on the FAIR Principles for Research Software (FAIR4RS) and on the automated composition of scientific workflows. Lamprecht is chair of the new special interest group on RSE that has recently been installed as a joint endeavour of the German Association for Computer Science and the German Association for Research Software Engineering. Contact her at anna-lena.lamprecht@uni-potsdam.de.

Jan Linxweiler is the general manager of the Center for Mechanics, Uncertainty and Simulation in Engineering (MUSEN) at TU Braunschweig and head of IT and Research Services at the University library. His research interests include Research Software Engineering, High Performance Computing, Research Data Management, and Open Science. Linxweiler holds a PhD in Engineering and is a founding member of the German RSE association of which he recently became chairman of the board. Contact him at j.linxweiler@tu-braunschweig.de.

Arnau Prat is a researcher at the German Aerospace Center (DLR). He leads the Payload Software research group. His current research interests include model-driven software engineering for space systems and the use of novel programming languages for safety-critical systems. He is currently involved in several space missions, ranging from payloads on board the ISS to sounding rockets or satellites. Contact him at arnau.pratisala@dlr.de.

Bernhard Rumpe is the Software Engineering chair at RWTH Aachen University and Editor-In-Chief of the SoSyM Journal. His main interests are rigorous and practical software and system development methods based on adequate modeling techniques. This includes agile development methods and model-engineering based on UML/SysML-like notations and domain specific languages. He also helps to apply modeling, e.g., to human brain simulation, contract digitalization, production automation, or cloud. He is author books of several like "Agile Modeling with the UML" and "Engineering Modeling Languages: Turning Domain Knowledge into Tools". Contact him at rumpe@se-rwth.de.

Katrin Schoening-Stierand is a senior digital consultant at the Hub for Computing and Data Science (HCDS) at the University of Hamburg. Her focus is on fostering interdisciplinary collaboration between different fields of science. Her research interests are informatics in the natural sciences and Research Software Engineering. She is involved in teaching and offers a course on Research Software Engineering. Contact her at katrin.schoening-stierand@uni-hamburg.de.

Shinhyung Yang is a postdoctoral researcher in the Software Engineering group at Kiel University with the SustainKieker project, focusing on the sustainability of research software. His research interests include performance engineering of cloud-native applications with a special focus on Java virtual machines and native applications in distributed and parallel architecture. He received a PhD degree from Yonsei University in South Korea, and his research is supported by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation), grant no. 528713834. Contact him at shinhyung.yang@email.uni-kiel.de.