

# Toward Research Software Categories

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*

Anna-Lena Lamprecht, *University of Potsdam, Potsdam, 14476, Germany*

Jan Linxweiler, *TU Braunschweig, Braunschweig, 38106, Germany*

Bernhard Rumpe, *Software Engineering, RWTH Aachen University, 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, developer-based, and maturity-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.*

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 [2], [3].

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 scientific research purposes.

In this paper, we propose a multi-dimensional categorization of research software, along the dimensions of roles, developers, and maturity. 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, developer-based, and maturity-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 [4] [5]. This *prescriptive* 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 or with a clear research intent should be considered “software in research” and not “research software.”

A *descriptive* definition of research software could

instead include all the software used in research, as for instance done in [6] for analyzing GitHub repositories. While such a descriptive definition may be useful in analyzing research processes, and therefore may be useful for RSE research [7], the prescriptive definition defines a clearer focus for the work presented here, and enables a better disambiguation of properties specific to research software. The Research Data Alliance also adopted the prescriptive distinction between *research software* and *software in research* [4], as we do. Figure 1 shows the resulting segmentation of software.



**FIGURE 1.** Segmentation of all software, research software, and software in research. In the present paper, we further categorize the orange box, i.e., 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 [8], [9];
- › in RSE Research [7], 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 categories 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 meanwhile constitutes a new paradigm of scientific inquiry next to theory and experiment [10] 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 [11]. 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*, which 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, or biological processes in spatio-temporal contexts.
- 2) *Proof-of-Concept Software* in science and engineering research.

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-categorical relations	What are the relations between different categories?

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

3) *Research Infrastructure Software*, such as research data and software management systems.

The assignment of a 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) [12]. In different contexts, a software may also be in multiple categories at the same time. As an example, the Monticore [13] framework for the development of domain specific languages initially was a proof-of-concept for researching implementation methods for developing domain specific languages. Meanwhile, Monticore is mainly used for developing specific domain specific languages, thus it turned into a research infrastructure. Still, some original research on developing domain specific languages is done in the Monticore context.

Category 1) software for modeling, simulation, and data analytics is quite large. We further refine this category 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) Integrative analysis (data assimilation and decision analysis)
- 1.4) Scientific visualization

Category 2) for proof-of-concept software is used in structural sciences (mathematics and computer science) and in engineering sciences (software, electrical, mechanical, and civil engineering). Examples for proof-of-concept research software are a software for flexible energy management of vehicle fleets within a harbor terminal that was developed and evaluated in the FRESH research project [14], and the Kieker monitoring framework that has been employed in various software engineering research projects [15].

Category 3) for research infrastructure software is also quite large. We further refine this category 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 (survey software, sensor-based data collection, etc.).
- 3.3) Pipelines, Workflows, and frameworks for composing software components.
- 3.4) Libraries, for instance for high performance computing.
- 3.5) Laboratory Notebooks.
- 3.6) Data Management.
- 3.7) Software Management.
- 3.8) Collaboration software.

These categories have varying quality requirements. 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 2, left, shows our resulting role-based categorization.

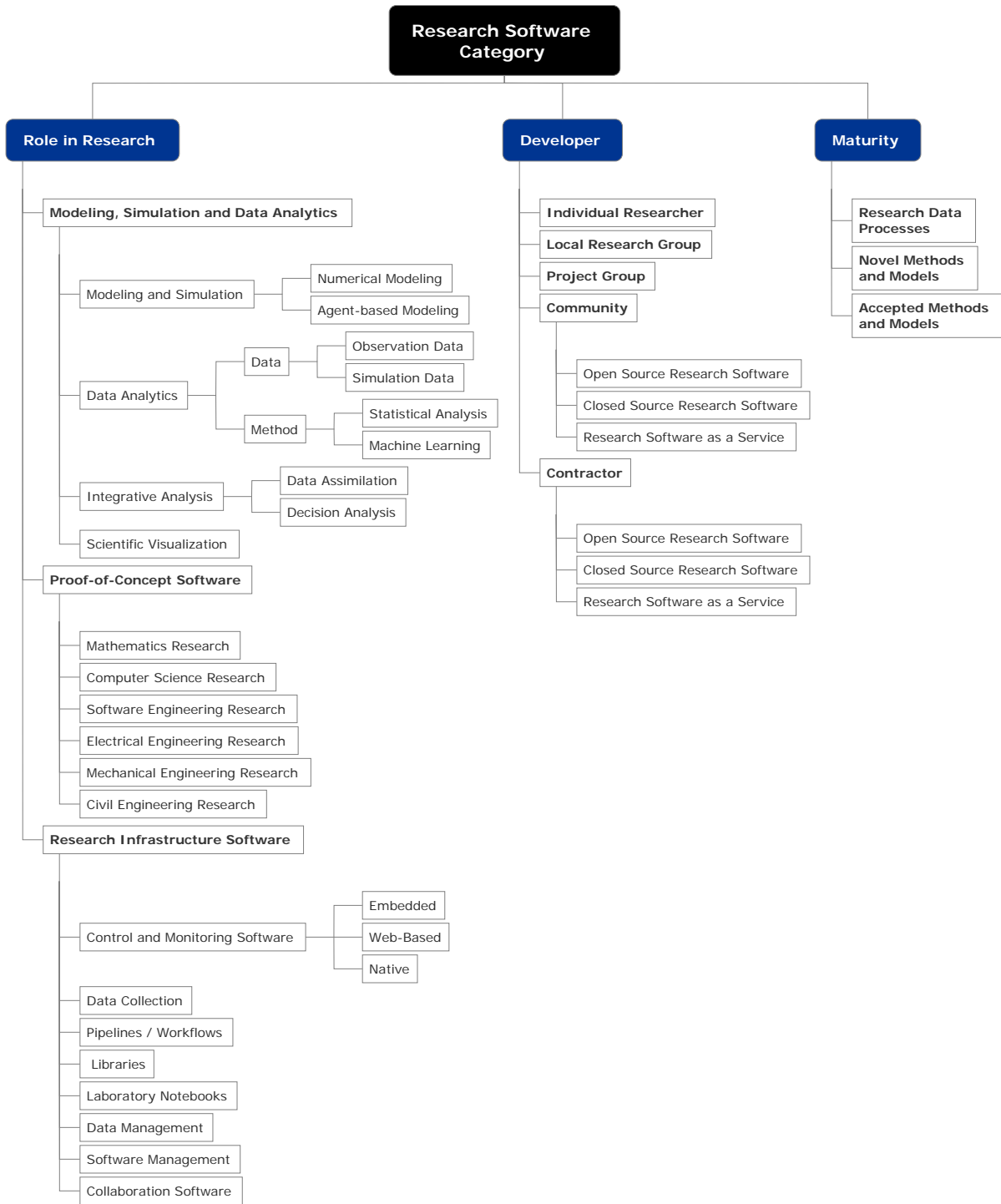
Table 2 characterizes our multi-dimensional categorization in terms of the template in Table 1. The developer-based and maturity-based categorizations are introduced in the following two sections, before we discuss some related categorizations.

## 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 or PostDoc.
- 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 the researchers).

A community or contractor may develop the software open-source, closed-source, or it may provide



**FIGURE 2.** Our multi-dimensional categorization of research software, along the dimensions of roles, developers, and maturity.

Criterion	Explanation
Scope	This categorization covers the dimensions of roles, developers, and maturity.
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 [7].
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-categorical relations	Individual research software may change its category within one or more dimensions.

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

research software as an online service. An example for a contractor-developed research software is PIA, the Prospective Monitoring and Management App, which has been developed by professional software companies as open-source software, on behalf of the Helmholtz Institute for Infection Research to conduct observational epidemiological studies by facilitating longitudinal data collection and cohort management [16].

Figure 2, middle, shows our resulting developer-based categorization.

### Maturity-Based Categorization of Research Software

Concerning a maturity-based categorization of research software, we adopt the ARDC approach [17]:

- 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.

Figure 2, right, shows the resulting maturity-based categorization.

This categorization system is not yet complete. To illustrate the gap between existing research categories, we characterize them in the following section, based on the characteristics from Table 1. Adding more dimen-

sions and refining the dimensions is subject to future work.

### 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, and b) may be used to compare and assess our categorization. We characterize these categories via the previously introduced template in the appendix (supplement).

#### Role-Based Categorization

Van Nieuwpoort and Katz [11] 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, but they suggest a different set of categories.

#### Maturity-Based Categorization

In their National Agenda for Research Software [17], 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 categorization of research software that we adopted for our maturity dimension.

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 [18][19]:

- › 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 maturity 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 [20]:

- 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. [21] 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 1.3), 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 [22]. This dimension could be the basis for another branch in our multi-dimensional categorization.

## 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.

Moreover, the categorization is intended to be a 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, facilitat-





ing 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 [7], 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.

We propose a multi-dimensional categorization of research software, along the dimensions of roles, developers, and maturity. The various dimensions of the categorization are not completely independent of each other. Looking at the dependency between the dimension and identifying constraints on combinations of the dimensions is the subject of future work. Additional dimensions could be the reuse scenarios (such as single-use/single-purpose, extensibility, reusability), the users (such as scientists, humans as research subjects, and citizens), the research software stack [22], and the criticality (for instance, mission-critical software). Such extensions and refinements are subject to future work.

## Appendix

### Characterization of Related Research Software Categories

The related research software categories are characterized in terms of the template in [Table 1](#).

[Table 3](#) characterizes the role-based categorization by van Nieuwpoort and Katz [11].

[Table 4](#) characterizes the ARDC categorization.

[Table 5](#) characterizes the institutional guideline application class categorization.

[Table 6](#) characterizes the EOSC research software lifecycle categorization.

[Table 7](#) characterizes the categories in computational research in the Earth system sciences.

[Table 8](#) characterizes the software stack categorization [22].

Criterion	Explanation
Scope	Role-based categorization.
Purpose	Funding organizations joined forces to explore how they could effectively contribute to making research software sustainable.
Context	International workshop in 2022 on the future of research software, organized by the Research Software Alliance (ReSA) and the Netherlands eScience Center.
Properties	The roles for research software are defined from the point of view of a researcher, with the goal of making this understandable for funders and policymakers.
Consequences for Creation	Depending on its role 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-categorical relations	Individual research software may change its role or take multiple roles.

**TABLE 3.** Characteristics of the role-based categorization by van Nieuwpoort and Katz [11].

Criterion	Explanation
Scope	The categorization in [17] supports a discussion about recognition of software in research, with the aim to increase this recognition.
Purpose	The categorization aims to describe the purpose of the software it categorizes as capturing applied or widely accepted research ideas, methodology, and models, or demonstrating new ones.
Context	The categorization has been produced in the context of ARDC's research software policy.
Properties	The properties of the categories represent different challenges faced by software that fall in the respective category.
Consequences for Creation	Depending on its category, software is expected to meet different requirements. While analysis code should be FAIR [4], prototype tools should exhibit a "high quality", and research software infrastructure must be created for sustainability, which is realized through safeguarding its long-term maintenance.
Consequences for Use	Software use is featured only implicitly in the categorization. We expect that software under the different categories are expected to be used differently: Analysis tools are used for specific research tasks, and are more likely to have a small scope, e.g., are applied only to answer a specific research question. Prototype tools are used to test the methodological hypotheses they implement, but may also be used experimentally to answer specific research questions.
Inter-categorical relations	The categories are related through <i>evolution</i> and <i>transitive value</i> . One category evolves from another, e.g., analysis code may evolve into a prototype tool, that in turn evolves into research software infrastructure.

**TABLE 4.** Characteristics of ARDC's research software categorization.



Criterion	Explanation
Scope	Guidelines for software engineering at an academic institution.
Purpose	Identify suitable quality requirements.
Context	Institutional policy and practice.
Properties	Criticality, institutional risk, projected use, development timeline, distribution, commercial exploitation.
Consequences for Creation	Increasingly employ established software engineering methods.
Consequences for Use	Increased (critical) use by increasingly large community.
Inter-categorical relations	Transitive requirements, legal requirements.

**TABLE 5.** Characteristics of institutional guideline application classes.

Criterion	Explanation
Scope	Developer- and stakeholder-based categorization.
Purpose	Achieve a common understanding of the current processes in research software engineering, particularly the research software lifecycle.
Context	SubGroup 1 “On the Software Lifecycle” of the EOSC Task Force “Infrastructure for quality research software”.
Properties	Different levels of adopting software engineering practice, different publication requirements and usage scenarios, different stakeholders
Consequences for Creation	Depending on its developer category, software is expected to meet different quality requirements and follow different development processes.
Consequences for Use	Increasing maturity and support for reproducibility
Inter-categorical relations	Not specified

**TABLE 6.** Characteristics of the EOSC research software lifecycle categorization.

Criterion	Explanation
Scope	Recommendations for universities, funders, and the scientific community.
Purpose	Safeguard the quality and efficiency of computational research in Earth System Sciences and make research results that have been generated by research software reproducible.
Context	Ideas of a DFG round table meeting on sustainable research software for high-quality computational research in the Earth System Sciences.
Properties	Research software developed in the Earth System Sciences is characterized by the complexity of the underlying models, multifaceted dependencies, the multi-modality of the data, and the size of the data, which can impose specific hardware and software requirements.
Consequences for Creation	Depending on its role category, software is expected to meet different quality requirements and follow different development processes.
Consequences for Use	Dependency on the research cycle
Inter-categorical relations	Combination, integration

**TABLE 7.** Characteristics of categories in computational research in the Earth system sciences.

Criterion	Explanation
Scope	Describing principles of software collapse.
Purpose	Identify dependent layers of different (academic) specificity to model threat.
Context	Research software sustainability.
Properties	Domain specificity.
Consequences for Creation	Build on stable lower layers, quickly react to threats, accept agility.
Consequences for Use	Decreasing specificity of application domain from top to bottom.
Inter-categorical relations	Dependency, transitive threats.

**TABLE 8.** Characteristics of categorizing the software stack.

## 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](https://doi.org/10.1109/MCSE.2018.021651343)
2. C. Jay, R. Haines, and D. S. Katz, "Software Must be Recognised as an Important Output of Scholarly Research," *International Journal of Digital Curation*, vol. 16, no. 1, p. 6, Apr. 2021. doi: [10.2218/ijdc.v16i1.745](https://doi.org/10.2218/ijdc.v16i1.745)
3. H. Anzt, F. Bach, S. Druskat, F. Löffler, A. Loewe, B. Y. Renard *et al.*, "An environment for sustainable research software in Germany and beyond: Current state, open challenges, and call for action," *F1000Research*, vol. 9, p. 295, Jan. 2021. doi: [10.12688/f1000research.23224.2](https://doi.org/10.12688/f1000research.23224.2)
4. N. P. Chue Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos *et al.*, "FAIR Principles for Research Software (FAIR4RS Principles)," *Research Data Alliance*, May 2022. doi: [10.15497/RDA00068](https://doi.org/10.15497/RDA00068)
5. 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](https://doi.org/10.5281/zenodo.5504016)
6. W. Hasselbring, L. Carr, S. Hettrick, H. Packer, and T. Tiropanis, "Open source research software," *Computer*, vol. 53, no. 8, pp. 84–88, Aug. 2020. doi: [10.1109/mc.2020.2998235](https://doi.org/10.1109/mc.2020.2998235)
7. M. Felderer, M. Goedicke, L. Grunske, W. Hasselbring, A.-L. Lamprecht, and B. Rumpe, "Toward research software engineering research," *Zenodo*, 2023. doi: [10.5281/ZENODO.8020525](https://doi.org/10.5281/ZENODO.8020525)
8. A.-L. Lamprecht, L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. M. D. Pico *et al.*, "Towards FAIR principles for research software," *Data Science*, vol. 3, no. 1, pp. 37–59, Jun. 2020. doi: [10.3233/ds-190026](https://doi.org/10.3233/ds-190026)
9. W. Hasselbring, L. Carr, S. Hettrick, H. Packer, and T. Tiropanis, "From FAIR research data toward FAIR and open research software," *it - Information Technology*, vol. 62, no. 1, pp. 39–47, Feb. 2020. doi: [10.1515/itit-2019-0040](https://doi.org/10.1515/itit-2019-0040)
10. T. Hey, S. Tansley, K. Tolle, and J. Gray, *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Oct. 2009. ISBN 978-0-9825442-0-4. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/>
11. R. van Nieuwpoort and D. S. Katz, "Defining the roles of research software," *Upstream*, Jun. 2023. doi: [10.54900/9akm9y5-5ject5y](https://doi.org/10.54900/9akm9y5-5ject5y)
12. D. S. Katz, "Incentives and frictions in community software projects," *Zenodo*, Jun. 2022. doi: [10.5281/zenodo.6677821](https://doi.org/10.5281/zenodo.6677821)
13. H. Krahn, B. Rumpe, and S. Völkel, "Monticore: a framework for compositional development of domain specific languages," *International Journal on Software Tools for Technology Transfer*, vol. 12, no. 5, 2010. doi: [10.1007/s10009-010-0142-1](https://doi.org/10.1007/s10009-010-0142-1)
14. S. Holly, A. Nieße, M. Tröschel, L. Hammer, C. Franzius, V. Dmitriyev *et al.*, "Flexibility management and provision of balancing services with battery-electric automated guided vehicles in the Hamburg container terminal Altenwerder," *Energy Informatics*, vol. 3, no. 1, p. 26, Oct. 2020. doi: [10.1186/s42162-020-00129-1](https://doi.org/10.1186/s42162-020-00129-1)
15. W. Hasselbring and A. van Hoorn, "Kieker: A monitoring framework for software engineering research," *Software Impacts*, vol. 5, p. 100019, Aug. 2020. doi: [10.1016/j.simpa.2020.100019](https://doi.org/10.1016/j.simpa.2020.100019)
16. J.-K. Heise, R. Dey, M. Emmerich, Y. Kemmling, S. Sistig, G. Krause, and S. Castell, "Putting digital epidemiology into practice: PIA – prospective monitoring and management application," *Informatics in Medicine Unlocked*, vol. 30, p. 100931, 2022. doi: [10.1016/j.imu.2022.100931](https://doi.org/10.1016/j.imu.2022.100931)
17. Australian Research Data Commons, "A National Agenda for Research Software," *Zenodo*, Mar. 2022. doi: [10.5281/zenodo.6378082](https://doi.org/10.5281/zenodo.6378082)
18. T. Schlauch, M. Meinel, and C. Haupt, "DLR Software Engineering Guidelines," *Zenodo*, Aug. 2018. doi: [10.5281/ZENODO.1344612](https://doi.org/10.5281/ZENODO.1344612)
19. O. Bertuch, D. Oliveira, U. Schelhaas, and A. Storm, "Guidelines for the development and distribution of software at Forschungszentrum Jülich," Forschungszentrum Jülich, Tech. Rep., 2022. [Online]. Available: <http://hdl.handle.net/2128/33259>
20. G. Courbebaisse, B. Flemisch, K. Graf, U. Konrad, J. Maassen, and R. Ritz, "Research software lifecycle," *Zenodo*, Sep. 2023. doi: [10.5281/zenodo.8324828](https://doi.org/10.5281/zenodo.8324828)
21. P. Döll, M. Sester, U. Feuerhake, H. Frahm, B. Fritzsche, 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](https://doi.org/10.23689/fidgeo-5805)
22. K. Hinszen, "Dealing With Software Collapse," *Computing in Science Engineering*, vol. 21, no. 3, pp. 104–108, May 2019. doi: [10.1109/MCSE.2019.2900945](https://doi.org/10.1109/MCSE.2019.2900945)