Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Towards FAIR Energy Research Software

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften
der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels

Doktor der Ingenieurswissenschaften (Dr.-Ing.)

angenommene Dissertation

von Herrn

Stephan Alexander Ferenz

geboren am 23.01.1995 in Hannover

1. Gutachterin:	Prof. Dr.-Ing. Astrid Nieße Department für Informatik Carl von Ossietzky Universität Oldenburg
2. Gutachter:	Prof. Dr. Wilhelm Hasselbring Institut für Informatik Christian-Albrechts-Universität zu Kiel
Tag der Disputation:	15. Juni 2026

Abstract

The rapid transition to carbon-neutral energy systems demands interdisciplinary and increasingly complex energy research. Energy Research Software (ERS) has become indispensable for analyzing and developing energy systems. However, its growing use comes with long-standing challenges, including low software quality, missing documentation, and limited testing. These factors lead to poor reproducibility and low reusability, which in turn hinder the reliability and effectiveness of energy research.

The FAIR4RS principles (Findable, Accessible, Interoperable, Reusable for Research Software) provide a framework for improving the reusability of research software, but their adoption in the energy domain is still at the beginning. Therefore, this thesis outlines an ecosystem for FAIR ERS consisting of different services and standards that can support researchers to make their ERS FAIR. Furthermore, the thesis investigates three fundamental components of this ecosystem in depth, resulting in three main contributions:

First, ten domain-specific recommendations for engineering ERS were derived from a literature review and two interactive workshops. The resulting guidance addresses documentation, modular design, testing, version control, and licensing, thereby providing practical advice for improving the quality and FAIRness of ERS.

Second, a new metadata schema for ERS, called *ERSmeta*, was designed to improve findability and reusability. Requirements were gathered from a qualitative interview-based study and a review of research software literature. The resulting element set was aligned with established schemas and ontologies, enriched with value vocabularies, and formalized into a metadata schema. An evaluation involving energy researchers indicated that most metadata elements are useful and relevant, while the overall number of elements introduced a complexity that requires further research.

Third, the Software Metadata Extraction and Curation Software (SMECS) was developed to help researchers to create metadata by harvesting existing information from repositories and supporting manual curation through a web interface. Usability tests confirmed that SMECS is functional, yet highlighted opportunities for extending extraction of information, simplifying the user interface, and expanding the export capabilities.

Altogether, the three artifacts are an important step towards an ecosystem for FAIR ERS that will enable efficient digitalized energy research required for the energy transition.

Zusammenfassung

Die notwendige Transformation zu klimaneutralen Energiesystemen erfordert eine interdisziplinäre und zunehmend komplexere Energieforschung. Dabei ist Energieforschungssoftware zu einem unverzichtbaren Werkzeug für die Analyse und Entwicklung von Energiesystemen geworden. Ihre zunehmende Nutzung bringt jedoch auch zentrale Probleme mit sich: Geringe Softwarequalität, fehlende Dokumentation und unzureichende Tests führen zu eingeschränkter Reproduzierbarkeit und niedriger Wiederverwendbarkeit und beeinträchtigen damit die Effektivität und Zuverlässigkeit der Energieforschung.

Die FAIR4RS-Prinzipien (Findable, Accessible, Interoperable, Reusable for Research Software) bieten einen wichtigen Rahmen für die Verbesserung der Wiederverwendbarkeit von Forschungssoftware, doch ihre Anwendung in der Energieforschung steht erst am Anfang. Um ihre Einführung zu fördern, skizziert die vorliegende Dissertation ein Ökosystem für FAIRe Energieforschungssoftware, das aus verschiedenen Diensten und Standards besteht und mit diesen Forschende dabei unterstützt, ihre Energieforschungssoftware FAIR zu gestalten. Ausgehend von dieser Grundlage untersucht die Arbeit detailliert drei zentrale Bausteine dieses Ökosystems, die die Hauptbeiträge dieser Arbeit darstellen:

Als erster Baustein wurden auf Basis einer Literaturrecherche und zweier interaktiver Workshops zehn zentrale Empfehlungen für die Entwicklung von Energieforschungssoftware zusammengestellt. Die Empfehlungen adressieren Dokumentation, modulare Architektur, Testsstrategien, Versionskontrolle und Lizenzierung und geben damit konkrete Ratschläge zur Verbesserung der Qualität und FAIRness von Energieforschungssoftware.

Außerdem wurde das Metadatenschema *ERSmeta* entwickelt, um die Auffindbarkeit und Interoperabilität von Energieforschungssoftware zu erhöhen. Dafür wurden Anforderungen aus qualitativen Interviews mit Energieforschenden und einer Auswertung der Literatur zu Forschungssoftware erhoben. Das resultierende Metadatenschema orientiert sich an existierenden Standards und Ontologien, wurde um Vokabularien für die verschiedenen Elemente ergänzt und abschließend formalisiert. In einer Evaluation mit Energieforschenden wurden die meisten Metadatenelemente als nützlich und relevant eingeschätzt. Gleichzeitig stellte sich die große Anzahl an Metadatenelementen als eine starke Komplexität heraus, die weitere Verbesserung erfordert.

Als dritter Baustein wurde das Tool SMECS entwickelt, um Forschende bei der Erstellung

von Metadaten zu unterstützen. Dazu extrahiert das Tool vorhandene Informationen aus Softwarerepositorien (GitHub, GitLab) und ermöglicht über eine webbasierte Oberfläche manuelle Kuratierung sowie den Export der Metadaten. Nutzungstests bestätigten die Funktionsfähigkeit von SMECS, aber zeigten auch Verbesserungsbedarf bei der Extraktion von Informationen, der Benutzeroberfläche und der Exportfunktion.

Insgesamt bilden die drei Artefakte einen wichtigen Teil des Ökosystems für FAIRe Energieforschungssoftware, das effiziente digitalisierte Energieforschung ermöglicht, die für die Energiewende notwendig ist.

Acknowledgements / Danksagung

Diese Dissertation entstand während meiner Tätigkeit an der Leibniz Universität Hannover, der Carl von Ossietzky Universität Oldenburg und dem OFFIS – Institut für Informatik und wurde vom niedersächsischen Ministerium für Wissenschaft und Kultur im Rahmen des Zukunftslabors Energie (ZLE, 11-76251-13-3/19–ZN3488) sowie von der Deutschen Forschungsgemeinschaft (DFG) im Rahmen der Nationalen Forschungsdateninfrastruktur für interdisziplinäre Energiesystemforschung (NFDI4Energy, 501865131) gefördert. Neben dieser Förderung wäre diese Arbeit in den vergangenen sechs Jahren nicht ohne die Unterstützung einer Reihe von Menschen möglich gewesen, die mich auf meinem Weg begleiteten und denen ich an dieser Stelle besonders danken möchte.

An erster Stelle gilt mein besonderer Dank meiner Doktormutter, Prof. Dr.-Ing. Astrid Nieße, die mir eine sehr wichtige Mentorin war und mich kontinuierlich förderte. Insbesondere unterstützte sie mich unermüdlich dabei, genügend Zeit und Fokus für meine Promotion zu finden und der Verlockungen anderer spannender Aufgaben zu widerstehen. Ihre Anregungen und ihre Expertise waren entscheidend für diese Arbeit.

Zudem danke ich Prof. Dr. Willi Hasselbring dafür, dass er die Rolle des Zweitprüfers übernahm. Aus unseren Begegnungen nahm ich stets neue Perspektiven und Ideen mit. Außerdem danke ich Prof. Dr.-Ing. Jürgen Sauer für die Übernahme des Prüfungsvorsitzes.

Mein Dank gilt auch Dr. Oliver Werth, der mich über den gesamten Promotionsweg in verschiedenen Rollen begleitete – erst als Projektkollege im ZLE und später als Gruppenleiter im OFFIS. Mit seiner Sicht als Wirtschaftsinformatiker bereicherte er meine Arbeit und diskutierte mit mir an vielen Stellen die notwendige Methodik.

Das Team im NFDI4Energy Coordination Office zeigte stets Verständnis, wenn ich mir mal wieder Zeit für meine Diss nehmen musste - ich danke euch dafür, dass ihr mir den Rücken in vielen Situationen freihieltet! Den guten Einstieg in die Koordinationsrolle verdanke ich vor allem auch dem NFDI Managementkreis, der mir den Start als Koordinator von NFDI4Energy erheblich erleichterte.

Viele der im Kontext dieser Dissertation entstandenen Veröffentlichungen waren Teamerfolge und insbesondere die fruchtbaren Diskussionen in diesen Teams ermöglichten diese Arbeit. Ich danke allen Co-Autor*innen für die gute Zusammenarbeit, inbesondere Emilie Frost, Rico Schrage, Thomas Wolgast, Alexandro Steinert, Jonathan Brandt, und Patrick

Kuckertz. A special thanks goes to Aida Jafarbigloo who was the important developer of SMECS - first as a student, later as a colleague!

Während der letzten Jahre war ich Teil von mehreren Arbeitsgruppen an drei Einrichtungen, die verschiedene wichtige Aspekte beitrugen:

Meine Kolleg*innen an der Universität Oldenburg waren stets mein fester Bezugspunkt in den letzten Jahren – vielen Dank! Insbesondere danke ich Thomas Wolgast und Paul Hendrik Tiemann, die mit mir den Weg nach Oldenburg auf sich nahmen – gemeinsam ist der Einstieg in neue Strukturen auf jeden Fall einfacher! Paul Hendrik Tiemann und Maria Albers nahmen mich in Oldenburg die meiste Zeit auf. Vielen Dank für eure immer währende Gastfreundschaft und die vielen schönen Abende!

Meine Kolleg*innen am OFFIS in den Gruppen SEA, COM, (DAI und darüber hinaus) ließen mich immer am OFFIS-Leben teilnehmen trotz meiner seltenen Zeit vor Ort. Vielen Dank für eure stete Offenheit!

Die meiste Zeit meiner Promotion verbrachte ich am Lehrstuhl für Elektrische Energiespeichersysteme (IfES-EES) der Leibniz Universität Hannover. Mein großer Dank gebührt Prof. Dr.-Ing. Richard Hanke-Rauschenbach, der dies mit ermöglichte und mich stets unterstützte! Die Zeit am IfES brachte viele lustige und schöne Momente – vor allem wegen der vielen Kolleg*innen, die mich so gut aufnahmen und durch ihre kreative Art und den guten Austausch stets den Arbeitsalltag bereicherten! Besonderer Dank gilt meinen Bürokolleg*innen: Hannah Hengstler, Levin Matz und Lena Bühre, die geduldig meine vielen Meetings ertrugen und stets offen für kurze Gespräche zwischendurch waren!

Einige dieser Kolleg*innen unterstützten mich auch durch ein kritisches Lesen dieser Arbeit. Ein großer Dank gebührt Lena Bühre, Hannah Hengstler, Aida Jafarbigloo, Levin Matz, Bernhard Miller, Rico Schrage, Alexandro Steinert, Paul Hendrik Tiemann und Janis Woelke für ihre wertvollen Kommentare und Anmerkungen. Zudem waren Emilie Frost und Malin Radtke, die zeitgleich den finalen Dissertationssprint hinlegten, besonders kritische Leserinnen meiner Arbeit. Vielen Dank für eure Anmerkungen und den guten Austausch!

Neben der Zeit im Büro war aber auch die Unterstützung abseits der Arbeit essenziell, um diese Disseration abzuschließen. Ich danke meinen Eltern, die mir mein Studium ermöglicht haben und mich über die letzten Jahre unterstützten. Zuletzt bin ich Philine zutiefst dankbar für ihre immer währende Unterstützung – Danke Philine, dass du immer verständnisvoll warst und mich dazu ermutigt hast, meine Zwischenerfolge zu feiern und stets mit Freude an die gesamte Arbeit zu gehen!

Hannover, den 20. Juni 2026

Stephan Ferenz

Contents

Acronyms

AI Artificial Intelligence. 20, 21, 114, 115

API Application Programming Interface. 25, 26, 32, 49, 68, 69, 82, 83, 86, 87, 89, 107, 112, 137

CFF Citation File Format. 48, 57, 58, 59, 73, 74, 77, 83, 86, 89, 107

CI/CD Continuous Integration/Continuous Delivery. 45, 48, 85

CMR Core Metadata Requirement. 55, 56, 57, 60, 61, 106, 145

CRediT Contributor Role Taxonomy. 117, 118, 119, 121, 122, 123, 124, 128, 129, 145

DCAT Data Catalog Vocabulary. 39, 58

DFG German Research Foundation (Deutsche Forschungsgemeinschaft). 22, 23

DLR German Aerospace Center (Deutsches Luft- und Raumfahrtzentrum). 22, 42

DOI Digital Object Identifier. 31, 83

EOSC European Open Science Cloud. 21, 23, 28

ERS Energy Research Software. iii, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 55, 56, 59, 60, 61, 63, 64, 65, 68, 70, 71, 73, 74, 76, 78, 80, 81, 97, 98, 99, 101, 102, 103, 105, 107, 109, 110, 111, 112, 113, 114, 115, 116, 117, 119, 120, 121, 125, 128, 131, 143, 144, 145

FAIR Findable, Accessible, Interoperable, Reusable. iii, v, vi, 2, 6, 7, 8, 9, 10, 11, 13, 21, 22, 23, 24, 26, 27, 28, 30, 42, 48, 53, 55, 63, 71, 81, 97, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 120, 126, 127, 143, 144

FAIR4RS Findable, Accessible, Interoperable, Reusable for Research Software. iii, v, 2, 6, 7, 9, 10, 11, 13, 17, 21, 23, 24, 25, 26, 27, 28, 30, 39, 55, 56, 61, 64, 71, 72, 108, 109, 116, 145

HERMES Helmholtz Rich Metadata Software Publication. 85, 86, 87, 89, 91, 95, 106, 107, 108, 112, 115

HPC High Performance Computing. 17, 18, 20, 70, 111

JSON JavaScript Object Notation. 32, 36, 37, 82, 84, 85, 89, 90, 94, 107

JSON-LD JavaScript Object Notation for Linked Data. 32, 34, 36, 37, 58, 61, 76, 77, 84, 86, 90, 92, 108, 111, 112

LOD Linked Open Data. 31, 34

Me4MAP Method for the development of Metadata Application Profiles. 60, 72, 74

NFDI German National Research Data Infrastructure (Nationale Forschungsdateninfrastruktur). 23, 28, 99

NFDI4Energy German National Research Data Infrastructure for Interdisciplinary Energy System Research (Nationale Forschungsdateninfrastruktur für die interdisziplinäre Energiesystemforschung). 99, 115

OEMetadata Open Energy Metadata. 73, 74, 87

OEO Open Energy Ontology. 76, 80, 126

OEP Open Energy Platform. 2, 57, 59, 60, 73, 74, 77

openmod open energy modeling initiative. 57, 59, 60, 73, 99

ORCID Open Researcher and Contributor ID. 31, 82, 83

ORKG Open Research Knowledge Graph. 34, 82, 84, 85, 108, 115

PID Persistent Identifier. 2, 6, 17, 26, 27, 32, 43, 48

RDA Research Data Alliance. 22, 24, 27

RDF Resource Description Framework. 31, 32, 33, 34, 36, 37, 38, 39, 82, 84, 86

ReSA Research Software Alliance. 19, 22, 24

RQ Research Question. 2, 10, 11, 12, 41, 55, 81, 97, 109

RSE Research Software Engineering. 8, 9, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 27, 28, 39, 41, 42, 43, 45, 46, 50, 51, 52, 61, 64, 71, 72, 82, 106, 109, 110, 111, 115, 119, 127, 132, 133, 143, 145

SHACL Shapes Constraint Language. 37, 61, 76, 77, 108, 111, 114

SMECS Software Metadata Extraction and Curation Software. iii, v, vi, 12, 81, 87, 88, 89, 90, 91, 93, 94, 95, 97, 98, 99, 101, 102, 105, 106, 107, 108, 111, 112, 113, 114, 115, 116, 120, 128, 129, 130, 143, 144

SMP Software Management Plan. 9, 10, 16, 17, 27, 50, 106, 114, 116

SoMEF Software Metadata Extraction Framework. 59, 85, 86, 87, 107, 115

SR Software Requirement. 82, 83, 84, 85, 87, 89, 90, 107, 108, 145

SUS System Usability Scale. 92, 93, 95, 98, 99, 101, 102, 105, 108, 112, 143, 144

UML Unified Modeling Language. 44, 62, 70, 88

URI Uniform Resource Identifier. 31, 32, 33, 35, 36, 37, 38, 56, 58, 59, 75

URL Uniform Resource Locator. 75, 87, 88, 89, 91, 136

W3C World Wide Web Consortium. 31, 37, 39

1 Introduction

A rapid transition to carbon-neutral energy systems is fundamental to tackle the climate crisis and is a key component of an 1.5°C-consistent pathway, according to the Intergovernmental Panel on Climate Change (IPCC)[1] [183]. This development includes the integration of multiple new technologies into energy systems to enable a high share of distributed renewable power generation and of electrified end-use [183]. Partially, these new technologies are enabled by increased digitalization, which leads to complex, interconnected, digitalized energy systems. The transformation of energy systems can only be successful when supported by trustworthy research. For example, analyzing multi-technology interactions in the energy domain becomes increasingly important to better understand the mutual influence of these technologies, e.g., to reduce the risk of error propagation [159].

A fundamental vehicle for energy research is self-developed research software as commercial tools are not flexible enough for complex research. This research software serves multiple purposes, such as visualizing processes and values, for example, power quality [37], the simulation of specific devices like fuel cells [223], the (co-)simulation of smart grids [200], or the analysis of transition pathways [156]. Therefore, Energy Research Software (ERS) contributes significantly to energy research and serves as the foundation for most research outcomes in this domain. Thus, there is a direct link between the quality of ERS and the quality of results in energy research.

In general, research software faces multiple challenges. For example, missing documentation (e.g., insufficient instructions on how to use the software or on the underlying assumptions) often hinders the reproducibility of research, which is one of the key elements of science [109]. Furthermore, building on existing research software is often impossible because suitable tools are either poorly documented or simply undiscoverable. Additionally, building on existing research software often carries the risk of including unmaintained software [8]. Overall, these issues have led to parallel developments of similar tools. For example, 63 software frameworks exist for optimization-based energy system modeling, many of which have highly overlapping features [118]. Often, new tools are developed without reusing existing ones. Thereby, the research process is slowed down because a lot

[1] https://www.ipcc.ch/, last accessed 2026-03-03

of time is spent on (re)developing research software instead of doing research.

In 2022, Chue Hong et al. [39] introduced the FAIR4RS principles (Findable, Accessible, Interoperable, Reusable for Research Software) to address these challenges. They adapted the FAIR principles (Findable, Accessible, Interoperable, Reusable) from Wilkinson et al. [220] which mainly focused on research data. The FAIR4RS principles provide high-level guidance on how research software should be treated to enable high reusability and support good scientific practice. Each principle contains multiple subprinciples, including diverse aspects such as metadata, interfaces, registering software, and versioning [39].

Generally, adopting the FAIR principles is a complex process [23]. For FAIR4RS, many supporting resources are still missing, such as services to provide Persistent Identifiers (PIDs), software registries, metrics, and others [15]. While some of these required resources are domain-agnostic, domain-specific efforts are also needed, e.g., for metadata standards [39].

Within the energy domain, the adoption of the FAIR4RS principles is in its early stages and only few supporting resources have been developed so far, e.g., the framework factsheets of the Open Energy Platform (OEP)[2]. Hence, the focus of this thesis is on the next relevant steps to enable the adoption of the FAIR4RS principles in energy research. This leads to the overall research goal: **Identify process steps and define resources to enable FAIR energy research software.**

To approach this research goal, Section 1.1 defines and characterizes ERS. Afterwards Section 1.2 presents the current state of FAIRness of ERS and a vision which resources can improve it. Based on this conceptualization, three detailed Research Questions (RQs) are developed and outlined in Section 1.3. The resulting structure of this thesis is presented in Section 1.4.

Parts of this chapter are already published in [77, 84, 89] (See Appendix A for detailed descriptions of the author's contributions to these publications).

1.1 Energy Research Software

Definition Energy Research Software (ERS) is the central object of this thesis. The term combines two concepts that are first clarified separately: *research software* and *energy research*.

The definition of the term *research software* has been the subject of an extensive debate. Gruenpeter et al. [102] summarized this debate and proposed an exclusive definition that

[2] https://openenergyplatform.org/factsheets/frameworks/, last accessed 2026-03-03

focuses on all software created for or during scientific investigations:

> "Research software includes source code files, algorithms, scripts, computational workflows and other executables that were created during the research process or for a research purpose." [102]

While *energy research* is an often used term, it is not well defined in the literature [187]. It is generally understood to comprise investigations on technologies for energy supply and/or end use [187]. This includes especially technologies that enable conversion and transmission of energy. Nowadays, energy research is often defined in a mission-oriented way, aiming at sustainability, net-zero emissions, and/or the energy transition, for example in the 8$^{\text{th}}$ German Energy Research Program [1].

By bringing these components together, ERS is defined as:

Definition 1 *Energy Research Software (ERS) is software that was created during energy research and/or especially to support energy research. Energy research aims to understand, analyze, improve, and/or design energy systems or components especially for energy systems.*

Like all research software, ERS spans a wide spectrum of technical complexity. At the simplest end are single-purpose scripts and small tools, e.g., written in Python. More elaborate projects can consist of libraries, modular frameworks, or full-scale software suites that integrate many different components and which might come with a user interface. Energy researchers often refer to ERS as models or frameworks.

Regarding functionality, ERS can visualize measurements or simulation results, analyze data from laboratory experiments or field deployments, and/or generate synthetic data for testing or scenario analysis. ERS can also implement mathematical models of individual energy components (e.g., heat pumps, batteries, or converters) or of entire energy system configurations. It can support the analysis of specific designs, e.g., including design optimizations, or control strategies which may also be implemented as prototypes. Additionally, ERS can indirectly support energy research. Regarding other software, ERS can manage other ERS and/or enable the orchestration of other ERS, like co-simulation frameworks. Regarding data, ERS can facilitate the organization and management of research data in energy research. Regarding hardware, ERS can enable hardware-in-the-loop experiments.

By fulfilling any of these purposes, ERS enables energy researchers to analyze current energy systems or their components and to explore new solutions for energy systems, thereby constituting an essential component of modern energy research.

Characteristics ERS has some typical characteristics which are summarized in Figure 1.1. Some of these characteristics are generally typical for research software:

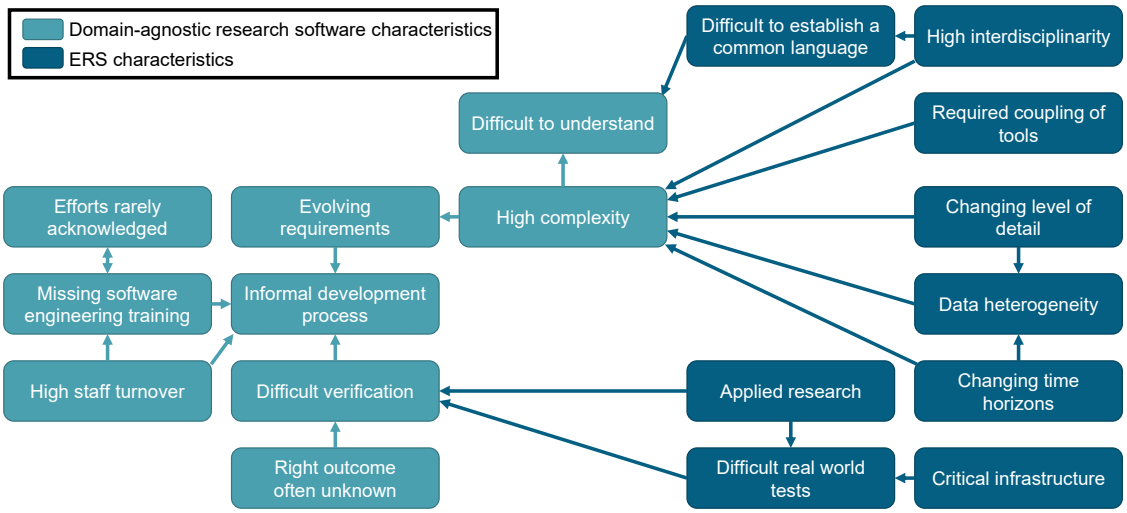


Figure 1.1: Overview of characteristics of research software (and its development) and of ERS. The arrows show how the different characteristics influence each other.

Efforts rarely acknowledged In many cases, research software often does not possess any value for the researchers beyond its specific use case [134]. It is often highly specific and created to solve a particular problem or set of problems [70]. Therefore, it is usually not designed for reuse and has a short lifecycle.

Missing software engineering training Most researchers are not explicitly trained in software engineering [59, 134]. Therefore, good practices from software engineering are often not applied in research software development.

High staff turnover The developers of research software are often not reachable later since they leave science directly or a few years after their degrees (PhD students and early postdoctoral researchers) [134].

Evolving requirements One of the most significant characteristics of research software is that the software requirements are mostly not known beforehand, but rather evolve during development as research software is used to study unknown phenomena and the knowledge on the topic evolves over time [70, 134].

Informal development process As a result of the evolving requirements, flexibility is essential, and iterative development is normal [134]. This leads to the development of research software often being done in an informal way [116].

Difficult verification The exact right outcome of an experiment is often not known which is a fundamental aspect of scientific research [134]. Therefore, verification against the real-world is an additional challenge for research software.

High complexity Regardless of the documentation, research software is often complex and difficult to comprehend, even for researchers from the same domain, due to the inherent complexity of the solved problems [134].

ERS shares these characteristics of research software, while some general aspects can be considered more critical in energy research. Additionally, ERS also has some special characteristics:

High interdisciplinarity Energy research is a highly interdisciplinary field with researchers having diverse backgrounds ranging from social science over engineering to different natural sciences and mathematics. As a broad range of domains are involved, it is hard to establish a common language. Therefore, it is difficult to achieve a mutual view of the problems that must be solved as a team. This issue influences the development of ERS as well as its capability of being understood by different interested researchers like project partners.

Required coupling of tools As all research software, ERS is diverse and complex. As huge infrastructure systems are under analysis, different types of simulation need to be coupled to analyze different aspects together [200, 215].

Data heterogeneity ERS often needs to support many different analysis methods and varying detail levels (e.g., transient vs. steady-state). Also, ERS often considers diverse time horizons, diverse spatial resolutions, and complex optimization problems [51, 67] which may lead to high performance requirements. To cover these different levels, a lot of diverse and heterogeneous data are used for most simulations and ERS often produces data different in structure, type, and size, which needs to be managed [226].

Applied research Energy research is an applied research field. Therefore, cooperation and joint projects with industry are widespread. This also has implications on the engineering of the jointly created software projects, e.g., with respect to open source or use of commercial software.

Critical infrastructure Some energy research also includes field tests which require highly reliable research software since energy systems are critical infrastructures.

Overall, energy research is highly diverse. As a consequence, it often takes a lot of time to identify if existing software meets the requirements of a planned research or has a different focus (especially regarding reliability, time horizons, and level of detail). This comes together with a highly interdisciplinary field where different disciplinary backgrounds lead to additional complications in communication. Concluding, the most essential aspects of ERS are the people involved and working on different detail levels, spatial resolutions, and time resolutions in a diverse and highly simulation-driven research area.

1.2 FAIR Energy Research Software

Current state After describing the characteristics of ERS, this section focuses on the current state of FAIRness of ERS. The FAIR4RS principles list general requirements how FAIR research software should look like. For this purpose, each principle contains multiple subprinciples which can be checked individually. The subprinciples especially cover topics like metadata describing the research software, archiving research software, registering research software, and more. A detailed overview of the FAIR4RS principles is given in Section 2.2.

As far as known, there exists no published systematic analysis of FAIRness of ERS to date. However, Wieners presented an analysis of the FAIRness of three exemplary ERS in his bachelor thesis [219]. The author investigated the frameworks mosaik, pandapower, and HOMER Pro. The co-simulation framework mosaik [200] is available open source and is developed and maintained by OFFIS[3], a research institute based in Oldenburg. The tool pandapower [209] can perform power flow calculation in power grids and can also compute optimal power flows. It is also available open source and is developed and maintained by the Fraunhofer Institute for Energy Economics and Energy System Technology (Fraunhofer IEE)[4] in Kassel. HOMER Pro[5] is a software to simulate microgrids, which was originally developed by the National Renewable Energy Laboratory (NREL)[6] but is now commercialized and closed source.

Wieners performed a detailed analysis of all three tools regarding the FAIR4RS principles[7]. The results of Wieners are summarized in Figure 1.2. Generally, none of the software fulfilled all FAIR4RS principles. The major reported issue was the limited scope regarding metadata. For HOMER Pro, metadata were not available at all. The existing entries in software directories for the other frameworks were partly outdated and also limited due to incompleteness of the metadata and the limitations of the underlying metadata approaches (this aspect is further described in Section 4.1). All three frameworks missed PIDs. Partly the user documentation of the software needed improvements to simplify reusability. [219]

As the three selected ERS were each maintained by established research institutions, they are atypical for the wider ERS landscape. Thus, it is assumed, that the FAIRness of smaller software is even worse. Overall, there is a substantial potential for improving the FAIRness of ERS.

[3] https://www.offis.de/en/index.html, last accessed 2026-03-03
[4] https://www.iee.fraunhofer.de/en.html, last accessed 2026-03-03
[5] https://www.homerenergy.com/products/pro/index.html, last accessed 2026-03-03
[6] which is named National Laboratory of the Rockies (NRL) today; https://www.nlr.gov/, last accessed 2026-03-03
[7] Since his work predated the official release of the FAIR4RS principles, he mainly used the preliminary principles as defined by Katz et al. [143].

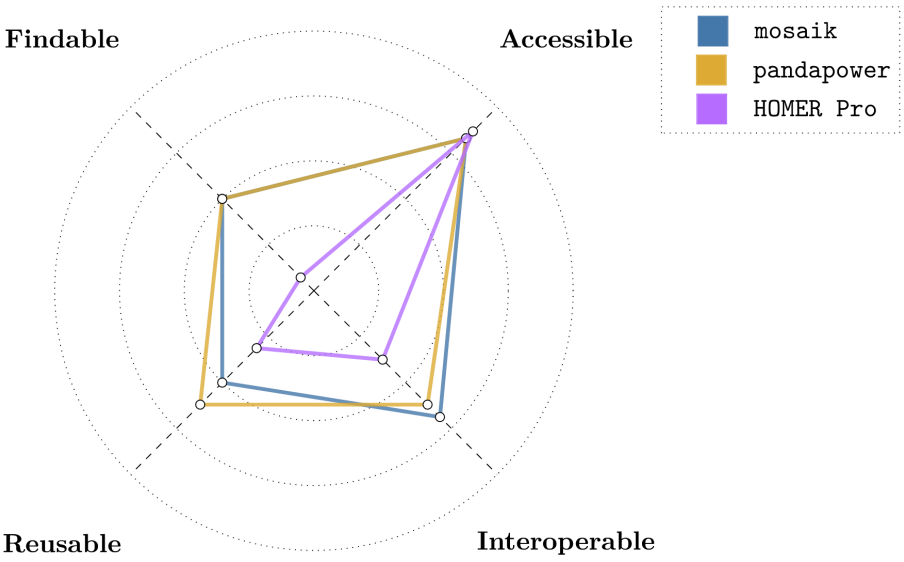


Figure 1.2: FAIRness evaluation of the ERS mosaik, pandapower, and HOMER Pro from [219]. The highest scores are on the outside.

Vision To make ERS FAIR, distinct activities are required by researchers in many phases of a typical research project cycle which ranges from analyzing existing solutions over developing software to ensuring the FAIRness of the software. This new way of handling ERS can be supported by a group of dedicated resources for each phase within the research cycle which simplify the FAIR handling of ERS and, therefore, become key enablers for FAIR ERS. Together, these resources form an ecosystem for FAIR ERS. In order to identify the required resources, this section presents a vision as an ideal process of how ERS could be managed in a FAIR way in future research projects.

The vision aligns the FAIR4RS principles with a typical research project cycle, illustrating where each resource can be used. The typical cycle of a research project is divided into five distinct phases: (1) starting with the analysis of existing research software, (2) planning for new software developments, (3) actual software development, (4) activities to make software findable, and, finally, (5) ensuring the software fulfills FAIR4RS principles to allow reuse. Each phase poses unique requirements and challenges that should be addressed while adhering to the FAIR4RS principles. Therefore, relevant services, standards, and recommendations are proposed to support the different activities in each phase. While the vision targets FAIR ERS, it includes domain-agnostic examples and examples from other domains to showcase how these resources can be helpful for researchers. The entire cycle is shown in Figure 1.3.

In the first phase (1), researchers will begin with an analysis of their relevant field

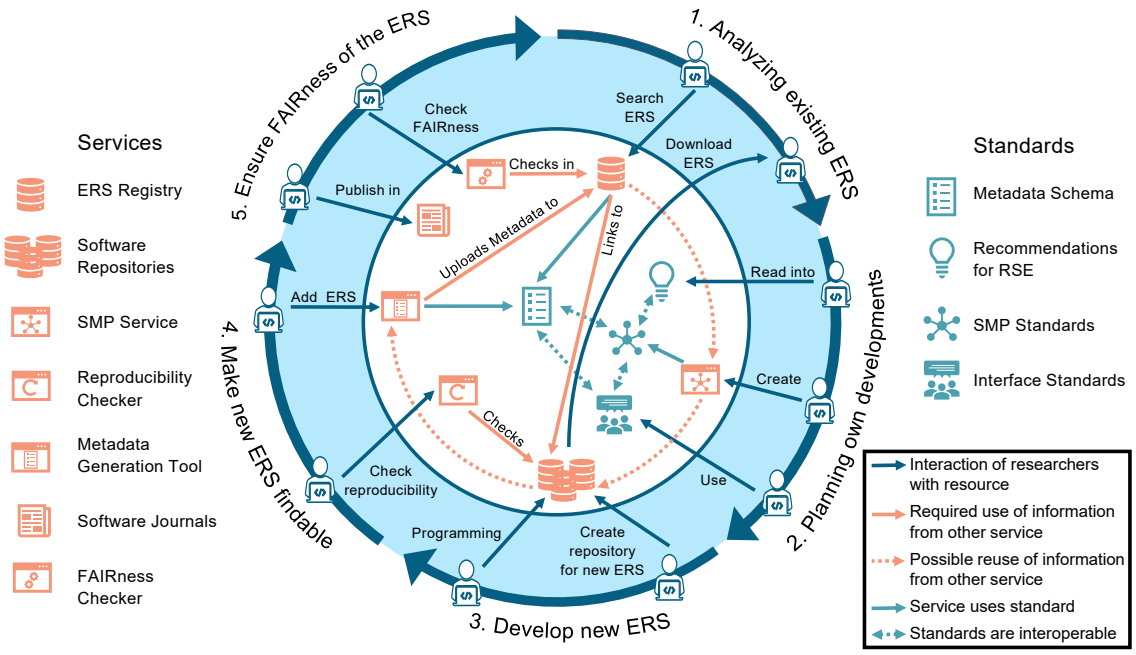


Figure 1.3: Vision of an ecosystem enabling FAIR ERS in different phases of a typical lifecycle of a research project.

by searching for ERS using an energy-specific software registry. Universally utilized and, therefore, complete software registries allow software discoverability and improve research transparency [95]. They only include software metadata while software repositories also store the source code [95]. An example for an established software registry is bio.tools[8] [127] in the life science domain.

The envisioned software registry includes metadata about different ERS like their title or their license. Providing metadata is an important key to enable good findability and reusability [39]. The metadata will follow a schema which standardizes the metadata properties and makes the metadata easier to compare and machine-readable. The metadata schema will be a domain-relevant community standard [39] that is interoperable with other schemas [95]. For example bio.tools uses the specific schema *biotoolsschema*[9] [128]. When the researchers find a fitting software in the registry, the registry links to the software repository (e.g., on GitHub[10]). The researchers will download the software and try it.

If the existing software does not fulfill all requirements, additional development will be planned in the second phase (2). Since researchers are often not trained in Research Software Engineering (RSE), they can get an overview of the most important RSE aspects

[8] https://bio.tools/, last accessed 2026-03-03
[9] Within this thesis metadata schemas and ontologies are written in *italic*.
[10] https://github.com/, last accessed 2026-03-03

by reading recommendations on RSE in their domain. Recommendations are an important instrument to successfully integrate the FAIR4RS principles into the routine practice of software design and use in research projects. Examples for relevant recommendations papers already exist in life science, e.g., by List et al. [155] who recommend how to deal with challenges like reproducibility, open source development, and other.

Based on the recommendations, the researchers will plan their specific development. Therefore, they will first write a Software Management Plan (SMP) which will help to plan and structure the development [4]. The SMP should be based on an energy-specific SMP standard. The researchers will use a specific service supporting them in writing the SMP. The envisioned service allows some automation, as proposed by Alves et al. [4], e.g., by reusing software metadata as a starting point. Therefore, the SMP will be based on the same metadata schema allowing high interoperability.

When planning their new software, the researchers will also consider and plan to use energy-specific interface standards, especially for data exchange. This will ensure interoperability with other ERS as emphasized by the first interoperability principles of FAIR4RS [39].

In the development phase (3), the researchers will first create a new repository, e.g., on GitHub or GitLab[11] to enable version control for their development. Ideally, the SMP service will support the creation of the new repository and will allow to reuse the information from the SMP. By sharing their developments early, the researchers will allow others to collaborate. Afterwards, they will develop their software to their needs.

In the next phase (4), the researchers will publish their research based on the software after finalizing it. Therefore, they will check the reproducibility of their work. For this step, they will use a tool which simplifies the check for reproducibility, a reproducibility checker. Also, they want to make their developed software findable. Therefore, they will create relevant comprehensive and standardized metadata to add the software to the registry. To simplify this process, they will be supported by a metadata generation tool which extracts as many information as possible from the software repository and other available sources. The tool will ensure that the metadata follow the metadata schema. It also allows the researchers to review the metadata and to add additional information.

In the last phase (5), the researchers want to increase the visibility and FAIRness of the software. Therefore, the researchers will publish their software in a software journal. Software journals offer a legitimate publication medium to ensure peer-review of the software components [198]. The software is normally presented in short articles which can be properly cited [198]. An example for a software journal is the Journal of Open Source Software (JOSS) [198].

[11] https://about.gitlab.com/, last accessed 2026-03-03

Finally, to see if further improvements can be made, the researchers will check the FAIRness of their software with a FAIRness checker. The service will work with the link to the entry in the software registry and will verify to which extent the software fulfills the FAIR4RS principles. A first approach for such a service is FAIRsoft which works on the metadata of a software [160].

For new research projects, the researchers will start the cycle again. Most steps stay relevant in every iteration, while some steps such as reading the recommendations get less relevant. Based on this vision of an ecosystem for FAIR ERS, the next section details the RQs addressing relevant resources of the ecosystem.

1.3 Research Questions

Based on the research goal and the conceptualization in Section 1.1 and Section 1.2, three detailed RQs are formulated for this thesis. These RQs focus on understanding three specific resources of the conceptual ecosystem presented in Section 1.2 more in detail: the recommendations for better engineering of ERS, the metadata schema, and the metadata generation tool.

These three resources are selected due to their high relevance for FAIR ERS. The recommendations for engineering of ERS are explored because they allow practical guidance for energy researchers to improve their current practice towards FAIR ERS. Thereby, the recommendations lay an important foundation for cultural change in the energy domain. Developing the recommendations allows to gain a more detailed understanding of the characteristics of ERS. Furthermore, the metadata schema is examined since metadata are a key element of FAIR research software. The schema serves as the foundation for different services, such as the registry which enables finding research software in the first phase of the research cycle (1) and for the SMPs service which is important for the planning phase (2). Additionally, a metadata generation tool is investigated because it plays a crucial role in enabling FAIR ERS by lowering the entry barrier to create high-quality metadata. It supports researchers in the fourth phase (4) and allows to simply adding ERS to a registry.

Targeting these resources, the following three RQs are formulated:

RQ1: Which recommendations can help energy researchers to develop more reusable ERS?

The first RQ focuses on the recommendations for good engineering of ERS. The aim is to identify the most important recommendations that can help energy researchers to develop more reusable ERS. For this goal, it is relevant to balance the different, sometimes

contradictory requirements for recommendations, such as being understandable for a broad audience, covering all relevant topics, being practicable, and serving as a starting point for readers who want to learn more about the topics.

RQ2: Which metadata are suited to improve the FAIRness of ERS and how can they be structured within a metadata schema?

The second RQ concentrates on the metadata elements required for improved FAIRness of ERS. It aims to identify the most important metadata elements that are necessary for achieving FAIR ERS. Also, energy researchers should be able to create the metadata with a reasonable amount of effort. Therefore, it is important to understand the level of information required to get good search results, to choose a ERS for reuse, and to achieve FAIR ERS. Meanwhile the elements should be accepted by researchers so they will provide the specified information their ERS. The metadata elements should allow maximal interoperability with existing standards for research software. To address this RQ, it is also relevant to investigate to what extent controlled vocabularies, such as ontologies, can be used to get harmonized metadata for ERS.

RQ3: How can a tool support the creation of metadata for ERS?

The third RQ focuses on ways to support the creation of metadata for ERS through a tool. This includes understanding which existing information can be reused to create high-quality metadata. Also, the RQ includes analyzing how a user interface can be designed to best support researchers in creating metadata. To allow broad use, the approach should achieve interoperability with domain-agnostic research software metadata while also supporting domain-specific metadata schemas.

1.4 Approach and Structure of the Thesis

This section presents the thesis structure to address all RQs. Figure 1.4 shows a summary of the structure together with its relation to the RQs, the developed research artifacts, and the related publications. A detailed overview of the scientific dissemination of this thesis including resulting publications is provided in Appendix A.

First, Chapter 2 introduces general foundations for this thesis. It provides an overview of RSE and introduces the FAIR4RS principles in detail. The chapter also outlines general aspects of metadata, the semantic web, and metadata schemas.

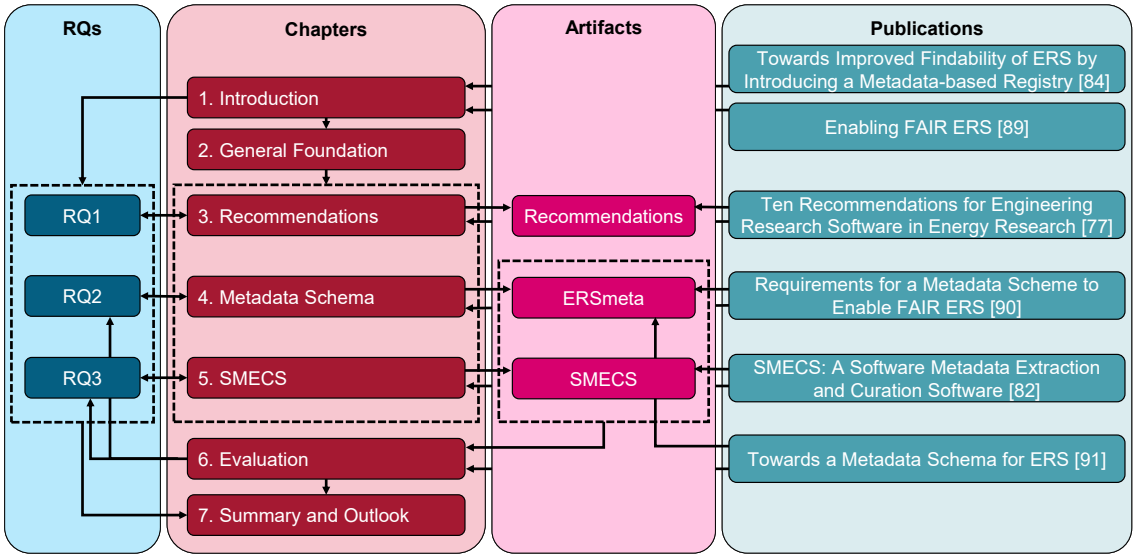


Figure 1.4: Overview of the overall structure of the thesis including RQs, resulting artifacts, and related publications.

Afterwards, Chapter 3 focuses on RQ1 regarding the recommendations for the development of ERS. The recommendations are published in [77].

Next, Chapter 4 addresses RQ2 regarding the metadata elements. Therefore, a new metadata schema for ERS was developed which is named *ERSmeta*. An important source for requirements for this schema was an interview-based requirements analysis published in [90]. The final metadata schema was included in a paper that is currently under review [91].

Then, Chapter 5 focuses on RQ3 regarding the support for the creation of metadata for ERS. The resulting artifact, the Software Metadata Extraction and Curation Software (SMECS), was published in [82].

To extend the work on the second and third RQs, *ERSmeta* was integrated in SMECS for further investigation. This part was included in a paper that is currently under review [91]. Chapter 6 presents this investigation with respect to RQ2 and RQ3.

Finally, Chapter 7 summarizes the results, draws a conclusion, and gives an outlook to future work.

2 General Foundations

Within this chapter, the general foundations for this thesis are introduced, as summarized in Figure 2.1. First, Section 2.1 outlines RSE as both a practical field and a research topic. The different aspects of RSE influence how FAIR research software can be achieved and supported. Afterwards, Section 2.2 describes the FAIR and FAIR4RS principles and summarizes the current state of their adoption. For the three artifacts in this thesis, specific technologies are relevant and, therefore, introduced in the subsequent three sections of this chapter. First, Section 2.3 provides an introduction to metadata, historically influenced by librarianship. In recent years, concepts from the semantic web have been applied to metadata, enabling numerous new applications. Consequently, Section 2.4 gives a brief overview of the concepts of the semantic web. Based on metadata and semantic web technologies, Section 2.5 introduces metadata schemas and provides common examples.

2.1 Research Software Engineering

RSE, also referred to as computational science [109], constitutes a central foundation for this thesis and is a relatively new field [70]. To present relevant aspects of RSE in this section, the field is separated into multiple clusters as shown in Figure 2.2 together with relevant

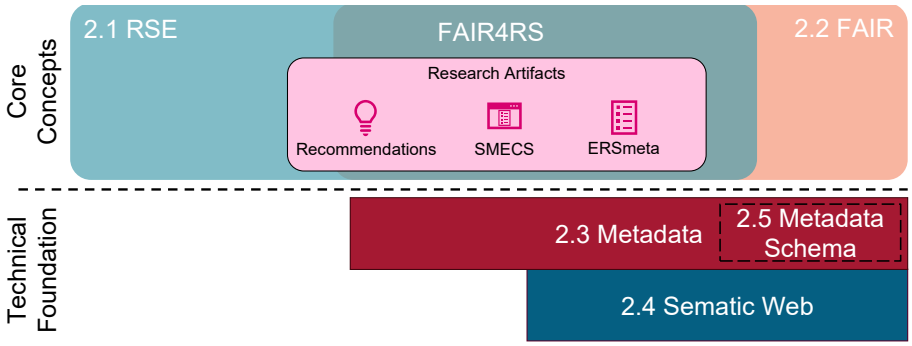


Figure 2.1: Overview of the general foundations as the chapter's sections, together with the placement of the thesis's research artifacts, illustrating how they relate to the core concepts.

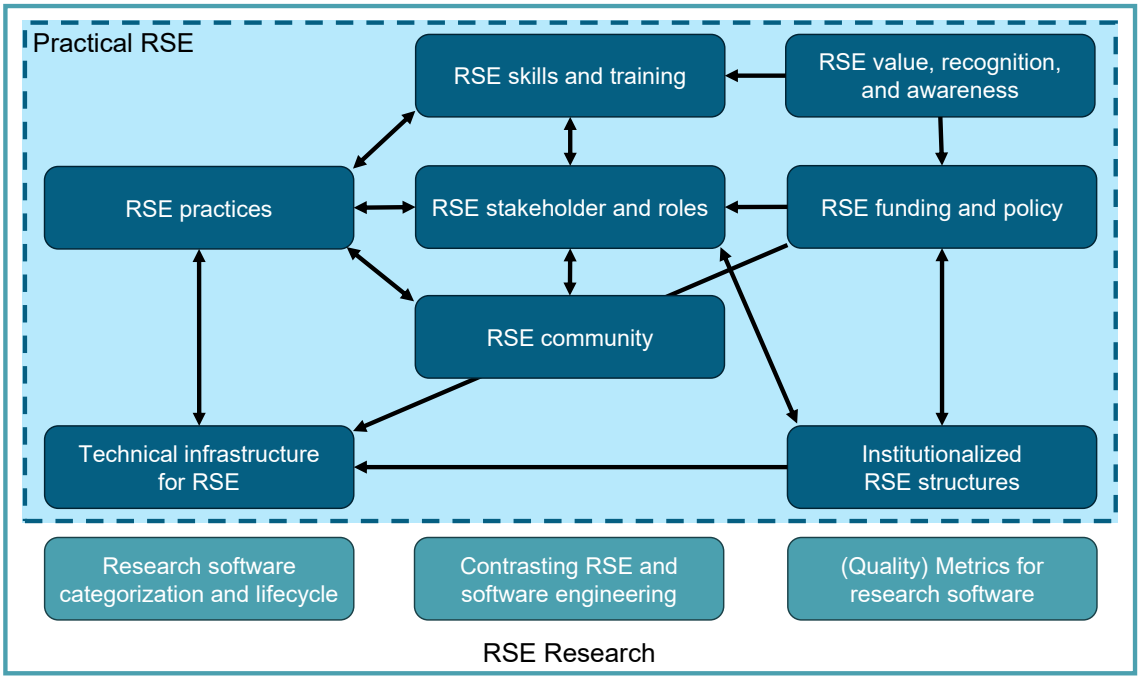


Figure 2.2: Overview of the RSE clusters and their interactions, inspired by Grunske et al. [103].

interactions between the clusters. The clustering takes into account that RSE is often approached from two complementary perspectives: practitioners who develop and maintain research software on a daily basis and researchers who study the underlying processes, methods, and organizational aspects of software development. For the second perspective, Felderer et al. [70] introduced the term RSE research. RSE research does not only cover research software itself but also its entire lifecycle, emphasizing process improvement, reproducibility, and sustainability [70]. On the practical side, RSE initiatives are often mission-driven, aiming to enable a more supportive environment for the development of research software [8].

As practical RSE and RSE research partly overlap and address similar aspects of RSE, they are not suited as clusters for this overview but form an additional dimension. Table 2.1 gives an overview how the different clusters relate to RSE research and practical RSE. Regarding suited clustering approaches for this section, Cohen et al. [40] identified four pillars: software development, community, training, and policy. Meanwhile, Grunske et al. [103] differentiated seven clusters: RSE practice, RSE training, RSE infrastructure, RSE community, RSE career pathways, RSE lobbying, and RSE research. The overview in this section as shown in Figure 2.2 and Table 2.1 extends the approach of Grunske et al. [103]. As

Table 2.1: Overview of RSE clusters and their characterization as RSE research and/or parctical RSE.

✓:	cluster is a relevant field of interest for the respective category
(✓):	cluster is partly relevant for the respective category
✗:	cluster is not relevant for the respective category

Clusters	**RSE Research**	**Practical RSE**
Research software categorization and lifecycle	✓	(✓)
Contrasting RSE and software engineering	✓	(✓)
Technical infrastructure for RSE	Design Science based development	Practical usage
RSE stakeholders and roles	(✓)	✓
Institutionalized RSE structures	(✓)	✓
RSE community	✗	✓
RSE skills and training	Analyzing required competencies	Practical training courses and curricula
RSE practices	Characterizing	Summarizing best practices
(Quality) Metrics for research software	✓	(✓)
RSE value, recognition, and awareness	(✓)	✓
RSE funding and policy	(✓)	✓

already outlined, RSE research is not a distinct cluster but may cover many RSE aspects to different extends. Yet, there are specific topics mostly associated with RSE research which are added as additional clusters: comparison of RSE with traditional software engineering, research software categorization and lifecycle, and the definition of quality metrics for research software. The RSE infrastructure cluster can be further divided into technical infrastructure (e.g., tools, platforms, and repositories) and institutionalized structures (e.g., organizational units and support services). Likewise, RSE lobbying encompasses funding and policy as well as the broader recognition of the value of research software. While these clusters are used to give a more detailed overview in the following, many clusters overlap to a certain extent.

Research software categorization and lifecycle The systematic categorization of research software and definitions of its lifecycle constitute a core focus of RSE research by providing a conceptual foundation needed for subsequent investigations across all other RSE clusters. Hasselbring et al. [111] summarized existing approaches for the categorization of research software and introduced a multidimensional categorization that captures aspects such as the software's role in research, technology-readiness level, developer profile, and dissemination strategy. In particular, their role-based dimension distinguishes three clear roles a software can take in research: modeling, simulation, and data analytics software; technology research software (prototype tools that demonstrate specific functionalities for emerging systems, further elaborated in [112]); and research infrastructure software (systems that support research activities, e.g., data management platforms or orchestration frameworks). This role centric view enables a software artifact to take multiple roles simultaneously, thereby reflecting the complex ways in which research software is employed.

A specific way of categorizing research software is a lifecycle perspective. Courbebaisse et al. [43] defined a lifecycle model and especially took Software Management Plan (SMP) templates into account. Their model consists of six phases: initialization, planning, development, publication, deployment, and community feedback. Yehudi et al. [224] proposed a different framework by incorporating different existing software cycles. Their framework addresses periods of reduced activity and potential re-activation, thus acknowledging that research software may transition to a dormant state before being revived for new purposes.

Contrasting RSE and software engineering Software engineering research is a classical research field in computer science, yet it had not covered the specific challenges of RSE for a long time [70]. Consequently, the two fields evolved largely in isolation [64]. This created an interesting opportunity for RSE research to examine how far classical software engineering knowledge could be transferred to RSE and where approaches need to be modified [70]. A major obstacle to this transfer is the divergent terminology used by RSE practitioners who often employ key concepts differently from the definitions established in traditional software engineering literature [146]. From a practical perspective, integrating established software engineering practice into RSE workflows remains a relevant topic [59]. This is especially relevant for requirements engineering methods [59]. Such integration can lead to better productivity and sustainability of research software development [64].

Technical infrastructure for RSE Effective development, use, dissemination, and long-term maintenance of research software require dedicated infrastructure that is largely composed of specialized software components offered as services. The infrastructure needs technical standards like metadata schemas, enabling interoperability, discoverability, and

Table 2.2: Overview of typical technical infrastructure for research software.

Types of Software for Infrastructure	**Explanation**
Repository	Software storing the source code of the software. The most typical types are: (1) version control systems which store different versions and are used for development, (2) software archives which archive a specific version of the software and often provide Persistent Identifiers (PIDs).
Registry	Software only storing metadata of research software.
Quality checker	Software checking the quality of a provided software, e.g., the FAIRness, often working on metadata.
Reproducibility checker	Software checking the reproducibility of a certain result.
Metadata tooling	Software supporting the creation and/or management of metadata (a detailed overview is provided in Chapter 5).
Software Management Plan (SMP) tooling	Software supporting the creation and use of SMPs.
Workflow managers	Software allowing to run specific tools in a row.
Orchestration software	Software allowing to couple different software to run in parallel together, e.g., for co-simulation.
High Performance Computing (HPC) tooling	Software supporting HPC like job schedulers and others.

compliance to the FAIR4RS principles. The conceptual framework presented in Section 1.2 already outlines a suite of services that are important for research software. Table 2.2 extends this overview to present relevant software types necessary as infrastructure for research software.

From an RSE research standpoint, the creation and evolution of such infrastructure can be pursued through systematic scientific approaches, such as design science research or engineering-focused investigations [112]. The requirements that shape this infrastructure are derived from the intrinsic characteristics of research software, e.g., its rapid development cycles and its environment. This makes the topic itself an additional interesting area for RSE research. From a practical RSE perspective, the availability of robust, well-engineered infrastructure is indispensable to develop, share, and sustain high-quality research software, thereby supporting the broader goals of the FAIR4RS principles.

RSE stakeholders and roles Anzt et al. [8] identified multiple relevant actors in RSE ecosystems: domain researchers, research software engineers, research leaders, scientific libraries, technical infrastructure units (e.g., for HPC), research performing organizations (e.g., universities and non-university organizations such as the Helmholtz, Max-Planck, Leibniz, and Fraunhofer societies in Germany), and funding agencies. A clear understanding of these actors is relevant for RSE research. Within RSE research, empirical work may analyze concrete roles in research software development, e.g., Jung et al. [138] for a research software project in ocean system modeling or Anderson et al. [6] for multiple GitHub repositories of research software projects.

The role of a research software engineer is of special interest in RSE. Baxter et al. [22] defined a research software engineer as a professional who combines software development expertise with extended knowledge of at least one research domain. Consequently, a spectrum of research software engineers exists, ranging from domain scientists who code and are primarily evaluated on their scientific output to dedicated research software engineers whose career advancement depends on the quality of their software [22]. When research software engineers primarily support researchers they take over software stewardship as an analogy to data stewardship [218]. The backgrounds of research software engineers are diverse and shape distinct career trajectories [42]. Many come from a domain science background while others have training in computer science and/or software engineering experience from industry [42]. Within their organizations, research software engineers may hold academic positions at any rank or serve as research support staff [218].

Institutionalized RSE structures The organization of RSE activities within research performing institutions varies considerably and most of the literature focuses on practical implementations rather than on systematic research of optimal structuring which would be part of RSE research. In many cases, the activities are embedded in existing units, either decentralized in departmental laboratories or research groups or centralized in IT services, libraries, or scientific computing centers. Additionally, they may be gathered in dedicated central RSE units, e.g., at Heidelberg University[1] [148] . Kempf et al. [148] summarized nine functional modules that can be covered by an RSE unit and analyzed to which extent four existing units cover these modules.

While institutional RSE structures are a way to create attractive, long-term career pathways for research software engineers [148], there are also challenges for these structures: (1) limited knowledge about how to design effective RSE support structures [8], (2) strong heterogeneity among research communities that leads to differing requirements for RSE

[1] https://www.ssc.uni-heidelberg.de/en, last accessed 2026-03-03

services [8], and (3) insufficient recognition of research software as a scholarly output, which hampers investment in dedicated staff [19].

Nationally funded structures may complement institutional efforts by aligning with country wide research funding mechanisms [141]. The United Kingdom's Software Sustainability Institute (SSI)[2] [44] is a successful example for such a model. The SSI is a partnership of four universities (Edinburgh, Manchester, Oxford, and Southampton) that provides consultancy, community events, fellowship programs, coordinated training (e.g., through the Carpentries[3] who offer modular training material and coordinate trainings), and policy support to foster cultural change [44, 145]. Similar initiatives are emerging in other countries, e.g., in the United States and Australia [141]. In Germany, the FutuRSI project[4] aims to establish a German research software institute that addresses the diverse German research landscape, expertise gaps, visibility of career paths, and the reuse of research software [157]. While such national entities are often limited by project-based funding and resource constraints, they provide a coordination and advocacy.

RSE communities Beyond formal institutions, RSE communities play a relevant role in knowledge exchange and professional development. Cohen et al. [40] identified community as one of the four pillars of RSE, encompassing networking, technical seminars, and collaboration. Numerous national and regional organizations exist across the United Kingdom, the Netherlands, Belgium, Germany, Denmark, the Nordics, Australia, New Zealand, and the United States [59]. The Research Software Alliance (ReSA)[5] provides a global platform for coordination. Katz et al. [145] offered a comprehensive overview of additional community initiatives and their activities. Together, institutional units, national institutes, and community networks constitute a multi-level ecosystem that supports the development, sustainability, and recognition of research software and RSE.

RSE skills and training Incomplete expertise is a main challenge for the development of sustainable research software since many research software engineers and researchers who write code are largely self-taught and lack formal training [103]. Especially for researchers, insufficient time for training is a major barrier [31]. Also, early access to targeted training materials is especially valuable for novice research software engineers [42].

From an RSE research perspective, identifying precise RSE competencies and skill requirements of research software engineers or researchers constitutes a relevant research

[2] https://www.software.ac.uk/, last accessed 2026-03-03
[3] https://carpentries.org/, last accessed 2026-03-03
[4] https://www.futursi.de/, last accessed 2026-03-03
[5] https://www.researchsoft.org/, last accessed 2026-03-03

topic which is complicated by the diversity of envisioned roles and tasks as previously introduced. For example, the skill set required for a dedicated research software engineer does not map directly onto that of a researcher [99]. Also, the required skills are evolving through the increased use of Artificial Intelligence (AI) for software development. Cohen et al. [40] identified a tiered approach that includes basic and advanced skill development, up-skilling of existing research software engineers, and support for newcomers. Goth et al. [99] proposed a comprehensive competency framework clustered into communication, research, and software skills, while Bernoth et al. [24] reported workshop outcomes that highlighted competencies such as publishing, archiving, and citing research software as well as developing and using research software.

While there are no accredited RSE training programs [42], there already exist a variety of training formats for research software competencies. The Carpentries[3] provide introductory workshops on programming, data handling, and HPC [40]. CodeRefinery[6] offers more advanced technical lessons and best-practice guidance [40]. The Digital Research Academy[7] delivers distributed training on software management topics. Additionally, some institutions run in-house courses, often coordinated by central RSE units [40]. In addition, formal academic pathways are emerging, e.g., the development of a master-level curriculum for RSE in natural sciences [54].

RSE practices A substantial body of literature addresses concrete RSE practices. From a practical RSE perspective, many publications summarize best practice recommendations often presented in the form of "ten rules" papers, for example [155, 178, 189]. Some of these focus on a specific research domain whereas others concentrate on particular aspects of RSE. Typical topics include (i) software architecture and modularization [26, 58, 227], (ii) development process (incl. documentation) [11, 114, 123, 126, 153], (iii) testing and quality assurance [63, 163, 173], and (iv) publication and citation of research software [3, 20, 139, 142, 164, 197, 199]. Legal considerations, such as intellectual-property rights and software licensing, are also relevant [8]. A more detailed discussion of existing RSE recommendations is presented in Section 3.1. Additionally, there are introductory books synthesizing essential practices [14] and institutional policies, like [190], that often codify important practices by articulating the expectations of research performing organizations.

From a research standpoint, RSE practices constitute a relevant field of inquiry which overlaps with the topics research software lifecycle and RSE roles. Interesting aspects include (i) organizing and modeling the RSE development process, (ii) characterizing research software creation and evolution [66, 107, 134, 138], (iii) studying software discovery

[6] https://coderefinery.org/, last accessed 2026-03-03
[7] https://digital-research.academy/, last accessed 2026-03-03

and reuse patterns [121, 188, 202, 207], and (iv) analyzing emerging trends in practical RSE, such as the integration of AI [69].

(Quality) Metrics for research software The primary purpose of metrics for research software is to assess research software quality [65], e.g., to understand how the quality compares to other software and how it can be improved. Thereby, the metrics can serve as actionable guidelines, especially when accompanied by concrete recommendations for improvement, and provide additional benefits for different stakeholder groups in practical RSE which are discussed in detail by Castell et al. [32]. From an RSE research perspective, the identification of precise, measurable quality indicators constitutes an active research topic and enables further research, e.g., how certain activities influence the software quality.

The FAIR4RS principles are an important part of research software metrics and are discussed in detail in Section 2.2. Their introduction started discussions about how to make these largely qualitative principles measurable. Within the context of the European Open Science Cloud (EOSC)[8], Chue Hong et al. [38] proposed a set of 17 metrics to quantify the FAIRness of research software. Meanwhile, several FAIRness checking tools (e.g., FAIRsoft [160] and FAIRSECO [53]) embed these existing metrics while also extending them by new ones based on observable software properties, e.g., Deekshitha et al. [52] introduced a Research Software Maturity Model (RSMM) that additionally incorporates sustainability, community engagement, and adaptability. At the institutional level, the Helmholtz Association adopted quality indicators derived from the FAIR4RS principles and extended them with metrics for scientific quality and technical competence [137].

Also, analyzing the adaptation of metrics from traditional software engineering approaches in RSE forms an interesting field of RSE research. An empirical study by Eisty et al. [65] revealed a gap between awareness and usage of traditional software metrics. Although research software engineers are familiar with traditional software metrics, they seldom apply them in practice [65].

RSE value, recognition, and awareness Missing recognition and awareness of research software constitute one of the most pressing challenges for RSE [8]. A growing body of publications seeks to elevate the visibility of research software and the awareness for its importance [55, 117, 119, 129]. A recurring difficulty is the measurement of research software impact [8]. Consequently, the development of robust metrics and impact analysis methods for research software remains a central topic.

[8] https://eosc.eu/eosc-about, last accessed 2026-03-03

RSE funding and policy Insufficient funding and a lack of incentives for high-quality research software remain among the most pressing obstacles for RSE identified by Anzt et al. [8]. Criteria of funding agencies are a powerful lever for shaping researchers' behavior but current schemes often fail to reward sustainable software development [131]. The Amsterdam Declaration on Funding Research Software Sustainability (adore.software[9]) brings together a range of funding organizations, including the German Research Foundation (DFG)[10], which agreed to adopt recommendations in following four areas: (i) research software practice, (ii) research software ecosystems, (iii) research software personnel, and (iv) research software ethics [5]. The effectiveness of such policies is closely tied to the broader recognition and awareness of research software and RSE within the scientific community [40]. From a research perspective, analyzing funding mechanisms is an active RSE research topic, e.g., [130, 132, 144]. The findings point to the need for more coherent, long-term funding strategies that explicitly address the full software lifecycle and the social dimensions of RSE [132].

Institutional policies constitute another essential instrument to enable researchers and research software engineers to allocate sufficient time to develop robust research software and to receive appropriate incentives. Also, these policies can facilitate the recognition of software contributions in hiring, promotion, and tenure decisions [179]. Therefore, Barker et al. [19] argue that comprehensive RSE policies are needed across research institutions. A joint working group from Research Data Alliance (RDA)[11] and ReSA [16] compiled a list of relevant resources for research organizations. Their analysis stressed that the involvement of all relevant stakeholder groups, like researchers, research software engineers, funders, and administrative units, is a key determinant of successful policy adoption [16]. In Germany, the German Aerospace Center (DLR)[12] developed their own institutional recommendations [190] while the Helmholtz Association[13] provided concrete recommendations for implementing RSE policies at the institutional level [182]. The German Informatics Society (GI)[14] and the German RSE association[15] released a generic RSE policy which should serve as model for German research institutions [48]. Together, funding mechanisms and institutional policies form a dual framework that can drive the cultural and structural changes required for sustainable, FAIR research software.

[9] https://adore.software/, last accessed 2026-03-03
[10] https://www.dfg.de/en, last accessed 2026-03-03
[11] https://www.rd-alliance.org/, last accessed 2026-03-03
[12] https://www.dlr.de/en, last accessed 2026-03-03
[13] https://www.helmholtz.de/en/, last accessed 2026-03-03
[14] https://gi.de/en/, last accessed 2026-03-03
[15] https://de-rse.org/en/index.html, last accessed 2026-03-03

2.2 FAIR and FAIR4RS Principles

General FAIR principles The FAIR principles were introduced in 2016 as a set of high-level, domain-agnostic guidelines for the management of research data to enhance their reuse [220]. They emerged from a workshop that brought together a broad spectrum of stakeholders including academic researchers, publishers, and representatives of the private sector to define a common vision for data stewardship [220]. The overarching goal of the principles is to improve the quality of research data so that it can be reused on its own or in combination with other datasets for secondary data analysis and data-driven science [220].

FAIR consists of four interrelated principles: Findability, Accessibility, Interoperability, and Reusability. Each of the principles is further refined by up to four subprinciples that specify rules for research data and, in particular, its metadata [220]. The emphasis on machine-actionable metadata enables automated discovery and integration of data, and, thereby, supports both human knowledge discovery and computational workflows [220]. The principles do not require the data themselves to be openly available but mandate that their metadata be openly accessible while allowing access restrictions on the underlying data when appropriate [220]. The principles are technology-agnostic, so they do not suggest any specific technology for their implementation [220]. By providing a clear set of objectives for research data management, the principles support good scientific practice and help to align research activities with the expectations of funding agencies and publishers.

Since its publication, the original FAIR article was cited more than 20.500 times[16]. The European Union incorporated the principles into its action plan for open science and supports their implementation through the EOSC[8] [56]. In Germany, the DFG[10] has embedded the FAIR principles into its guidelines for safeguarding good research practice [105] and the federal and state governments have adopted FAIR as a guiding framework for the German National Research Data Infrastructure (NFDI)[17] [29].

Towards FAIR4RS While the original FAIR principles were formulated for research data, the authors already noted that the principles should also apply to "algorithms, tools, and workflows that led to that data" [220]. Because research software can be regarded as a form of research data [109], a lively discussion has emerged on how the FAIR principles can be mapped to software artifacts.

Examining the applicability of FAIR to research software,Hermann et al. [115] emphasized the central role of metadata and pointed out that versioning is an important aspect of software that must be addressed. Hasselbring et al. [109] highlighted the need for research

[16] Google Scholar citation count, last accessed 2026-03-29
[17] www.nfdi.de/?lang=en, last accessed 2026-03-03

software to be both FAIR and open. They also observed that code quality is not covered by the original principles while it is often a main barrier for reuse. Their work offered a first overview of software-specific reformulations of the FAIR principles. Lamprecht et al. [151] summarized insights from multiple workshops and also presented a draft of FAIR principles for research software. They stressed multiple fundamental differences between data and software: software is executable, frequently depends on other software packages, and requires explicit version control. The authors also questioned whether FAIR software must be openly licensed and argued that software quality aspects, particularly sustainability and maintainability, should be incorporated into the FAIR framework. Their draft retained the overall structure of the original principles but introduced detailed modifications for interoperability and reusability [151].

To achieve a common definition, a FAIR4RS working group was established under the RDA, the ReSA, and FORCE11[18] [140]. The group organized several sub-working groups that addressed distinct topics such as the precise definition of research software [102], the broader conceptualization of FAIRness for software [152], and the formulation of the principles. Katz et al. [143] contributed a formulation of the principles that treats software as a living, complex object emphasizing its executability and creative provenance. Their version largely mirrors the data-centric wording but replaces *data* with *software* and introduces specific changes to the interoperability and reusability subprinciples. They especially noted that FAIR alone is insufficient for full computational reproducibility [143]. The collaborative effort of the working group resulted in a final set of FAIR4RS principles published by Chue Hong et al. on Zenodo [39] and subsequently described by Barker et al. in the journal *Scientific Data* [18]. The proposed FAIR4RS principles provide a domain-independent, technology-agnostic framework that extends the original FAIR principles to encompass research software, explicitly addressing versioning, software dependencies, quality, and sustainability while preserving the core goals of findability, accessibility, interoperability, and reusability.

FAIR4RS principles The following description of the FAIR4RS principles is based on their publication by Chue Hong et al. [39]. All FAIR4RS principles and their subprinciples are shown in Table 2.3. Generally, it should be acknowledged that the FAIR4RS principles are meant to be aspirational and, therefore, do not always include practical guidance [39]. Similar to the FAIR principles, the FAIR4RS principles also cover the accompanying metadata [39].

The findability principle should ensure that the software and its metadata can be dis-

[18] https://force11.org/, last accessed 2026-03-03

Table 2.3: FAIR4RS principles as defined in [39].

Findable: The software, and its associated metadata, should be easy to find for both humans and machines.	F1. Software is assigned a globally unique and persistent identifier.	F1.1. Different components of the software must be assigned distinct identifiers representing different levels of granularity. F1.2. Different versions of the same software must be assigned distinct identifiers.
	F2. Software is described with rich metadata.	
	F3. Metadata clearly and explicitly include the identifier of the software they describe.	
	F4. Metadata are FAIR and are searchable and indexable.	
Accessible: The software, and its metadata, must be retrievable via standardized protocols.	A1. Software is retrievable by its identifier using a standardized communications protocol.	A1.1. The protocol is open, free, and universally implementable. A1.2. The protocol allows for an authentication and authorization procedure, where necessary.
	A2. Metadata are accessible, even when the software is no longer available.	
Interoperable: The software interoperates with other software through exchanging data and/or metadata, and/or through interaction via Application Programming Interfaces (APIs).	I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards. I2. Software includes qualified references to other objects.	
Reusable: The software is both usable (it can be executed) and reusable (it can be understood, modified, built upon, or incorporated into other software).	R1. Software is described with a plurality of accurate and relevant attributes.	R1.1. Software must have a clear and accessible license. R1.2. Software is associated with detailed provenance.
	R2. Software includes qualified references to other software.	
	R3. Software meets domain-relevant community standards.	

covered by machines and humans. Therefore, the software packages and the metadata need PIDs. As previously mentioned, one specialty of software is the availability of different versions. Therefore, each version must receive its own identifier and the relationships among the versions must be expressed in the metadata, e.g., *newer-than* or *derived-from*. Large software systems may also need distinct identifiers for sub-modules, libraries, or plugins. To support machine-driven discovery, the metadata describing the software must be machine-readable. They should conform community-specific metadata standards and employ controlled vocabularies. The metadata record itself must be FAIR as well. Therefore, it should link to the PID of the software and be published in or harvested by a software catalog or registry. [39]

The accessibility principle concerns the ability to retrieve both the software artifact and its metadata via standard, open communication protocols such as HTTP/HTTPS or FTP. These protocols must be free of licensing restrictions to enable unrestricted technical access. Similar to the FAIR principles, the FAIR4RS principles acknowledge that there are cases where software can not be open due to intellectual-property, security, or privacy constraints. In these cases, the protocol must support authenticated access and the metadata must remain openly available even if the software itself is restricted. This ensures that users can at least discover the existence of the software and understand the conditions under which it may be obtained. [39]

The interoperability principle demands that the research software is capable to interact with other software through well-defined APIs, e.g., for exchanging data. These interfaces should follow domain-relevant standards and be described using formal specifications and controlled vocabularies for data types, formats, and communication patterns. The software's metadata must explicitly reference any external objects required for operation, such as reference datasets, configuration files, or dependent libraries. Where possible, those objects should themselves satisfy the FAIR principles. [39]

The reusability principle encompasses two complementary aspects. First, the software must be usable meaning executable by third parties. This implies that sufficient runtime artifacts (e.g., binaries, containers, or build scripts) are provided. Second, the software should be reusable, meaning understandable and modifiable, so that it can be extended or incorporated into other projects. To achieve both aspects, the metadata must contain a machine-readable license specifying the conditions under which the software is allowed to be used, modified, and redistributed, as well as provenance information that records the origin, authorship, and development history of the software. The software should reference all its dependencies ideally with PIDs. Finally, the software should meet domain-relevant standards with respect to documentation, license, coding conventions, and more. [39]

Reception and adoption for the FAIR4RS principles The uptake of the FAIR4RS principles is tightly intertwined with activities in the RSE community as introduced in Section 2.1. When the principles were first published, Chue Hong et al. [39] already identified several obstacles that need to be addressed, e.g., domain-specific metadata schemas and vocabularies existed only for a limited number of research areas. Likewise, there was no common answer to the question which PID should be used for software artifacts and their versions. Therefore, the authors called for additional efforts before the principles can be widely applied [39].

Since their introduction, the two papers defining the FAIR4RS principles [18, 39] have been cited more than 600 times[19], indicating a growing interest. In 2024, Barker et al. [17] presented a comprehensive report on the state of the adoption. Their analysis covered five interrelated topics: policy, incentives, community, training, and infrastructure. They concluded that progress is needed in all of them to achieve the cultural change required to fully adopt the principles.

Some policy initiatives already exist at both national and institutional levels, e.g., national policy documents in the Netherlands explicitly reference the FAIR4RS principles and tie them to the creation of SMPs [17]. Overall, incentive structures are still scarce. Jensen and Katz [130] examined the awareness of funding agencies which are generally less familiar with FAIR4RS principles than with the original FAIR principles.

Community engagement is fostered through several mechanisms. The RDA hosts a software source code interest group that discusses FAIR4RS related topics and the RSE communities promote the principles as part of their best practice repertoire (see Section 2.1). Also, training programs increasingly embed FAIR4RS concepts, e.g., the Netherlands eScience Center[20] includes the principles in its curricula for early-career researchers. These educational efforts are essential for translating the abstract principles into concrete actions [17]. Practical guidance for researchers and research software engineers often emerges in domain-specific contexts. Patel et al. [172] offered actionable recommendations for biomedical software developers, illustrating how the abstract principles can be translated into concrete development workflows. Such guidance overlaps with the broader effort to provide recommendations on RSE.

Additionally, the FAIR4RS principles increased the attention for infrastructure development for research software which is partly described in Section 2.1. At the European level, several large-scale initiatives are dedicated to making research software FAIR, e.g., the European Virtual Institute for Research Software Excellence (EVERSE)[21], the FAIR-

[19] Google Scholar citation count, last accessed 2026-03-27
[20] https://www.esciencecenter.nl/, last accessed 2026-03-03
[21] https://everse.software/, last accessed 2026-03-03

IMPACT project[22], and the FAIRCORE4EOSC[23]. They are funded in the context of the EOSC and aim to provide tools, metadata standards, and sustainable infrastructure for FAIR4RS. In Germany, the NFDI started to address research software as a relevant digital research object with working groups on RSE [106] and research software metadata [34]. The advances regarding metadata standards are detailed in Section 4.1. Although, Hounsri and Garijo reported that adoption of metadata standards still remains low across European Open Science Clusters [120].

2.3 Metadata

Section 2.2 highlights the important role of metadata in achieving FAIR research software. Therefore, this section introduces the fundamental concepts of metadata.

The systematic description of objects in libraries and museums, as known today, can be traced back to the 19th century, when they began to use catalogs governed by explicit rules for recording information about physical items [225, p. 14]. With the upcoming of electronic computing, these catalog systems were digitized [225, pp. 14-19]. The emergence of the internet subsequently enabled the interlinking of disparate catalogs, creating a clear need for shared metadata standards which were subsequently developed and refined [225, pp. 14-19]. Today, metadata are not only used to describe tangible artifacts but also to characterize digital resources such as research data and software.

Thereby, metadata serve multiple purposes: they describe resources, enable their systematic organization and discovery based on controlled criteria, support aggregation of similar resources, facilitate information exchange, provide digital identification, and supply essential information for long-term archiving [225, p. 11].

Fundamental principles Metadata contain multiple statements which each consist of a single property-value pair that characterizes a specific *resource*[24] [225, p. 3]. For example, the statement illustrated in Table 2.4 makes it explicit that the programming language of the software mosaik [200] is Python. The resource being described is also called an entity. The property appears in the literature under various names, like attribute, field, tag, or *element* [162, p. 13]. When multiple metadata statements are assembled they constitute a description [225, p. 7] or a *metadata record* [162, p. 6] of the resource. Once the metadata record is formalized and encoded, it is referred to as metadata [225, p. 7] (more information

[22] https://fair-impact.eu/wp4-metadata-and-ontologies, last accessed 2026-03-03
[23] https://faircore4eosc.eu/eosc-core-components/eosc-software-heritage-mirror, last accessed 2026-03-03
[24] The terms further used in this thesis are *emphasized* in this paragraph.

Table 2.4: Example metadata statements for mosaik [200].

Resource	Property	Value
mosaik	programmingLanguage	Python
mosaik	developmentStatus	Active

on encoding metadata is provided in Section 2.5). A set of elements forms an element set which is a fundamental building block of any metadata schema or standard [225, pp. 12-13]. Metadata schemas are introduced in Section 2.5.

Metadata categories Metadata can be divided into several categories that correspond to different purposes. Table 2.5 provides an overview of typical categories and their overlap. Zeng and Qin [225, pp. 19-22] identified five primary categories: administrative, descriptive, preservation, technical, and use metadata. Administrative metadata contain information needed for managing a resource, such as access requirements, copyright statements, licensing details, and technical data required for decoding or handling the item. Descriptive metadata provide the elements that enable users to locate and understand a resource. They may include basic bibliographic elements (author, title, keywords) as well as richer content-level descriptors like about the context (e.g., author affiliation, geographic location) of the resource. Preservation metadata are mainly relevant for metadata of physical objects, e.g., for documenting their condition. Technical metadata include information how a system

Table 2.5: Metadata categories based on Zeng and Qin [225, pp. 19-22].

Category	Example Elements	Overlap
Administrative	Rights, location information	Provenance, technical, preservation, meta-metadata
Descriptive	Keywords, authors, title	Provenance
Preservation	Physical condition	Administrative
Provenance	Creation history, rights information	Administrative, descriptive
Technical	Software requirements, authentication	Administrative
Use	Use record	
Meta-metadata	License of the metadata, author of the metadata	Administrative
Structural	Part of	

works, e.g., on hardware and software requirements. Use metadata focus on the previous use of the resource, e.g., how a resource is cited, downloaded, reviewed, or otherwise reused. [225, pp. 19-22]

In practice these categories intersect. An additional category is provenance metadata which capture the creation history, versioning, and rights information of a resource, thereby supporting reproducibility and trust [225, p. 21]. Meta-metadata describe the metadata themselves, such as the standard employed, the license governing the metadata, or the creator of the metadata record [225, p. 21].

Metadata services Metadata services support the usage and/or creation of metadata. They often rely on metadata schemas which are described in detail in Section 2.5. Metadata can be embedded directly with the digital resource itself or in dedicated databases forming a metadata service [225, pp. 256-271]. Zeng and Qin [225, pp. 256-271] distinguished three main types of metadata services:

Metadata registries Metadata registries contain information about existing metadata schemas including their element sets and further details. They enable users to discover and reuse domain-specific standards, thereby avoiding redundant schema development [225, pp. 256-271].

Metadata repositories Metadata repositories store the actual metadata records that describe concrete resources, e.g., research software packages. The metadata repositories act as registries for those resources. Metadata records from multiple repositories can be harvested and aggregated through the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [225, pp. 256-271]. For FAIR artifacts, the metadata records need to be openly accessible.

Development and production services A range of tooling facilitate the creation, design, and conversion of metadata, including utilities for schema development, automated metadata extraction, and metadata harvesting [225, pp. 256-271]

Many of these services depend on technical standards to ensure interoperability and machine-readability across heterogeneous platforms. They often use the semantic web technologies [225, pp. 272-273].

2.4 Semantic Web

The FAIR4RS principles explicitly require that research software be discoverable not only by humans but also by machines [39]. Achieving this level of machine findability depends on semantic web technologies that provide the foundation to expose identifiers, provenance,

dependencies, and licensing in a computable way.

The original vision of the semantic web is "to make the web more accessible to computers" [7, p. 1]. By publishing metadata in machine-readable formats, the content of the internet becomes understandable and processable by automated tools [7, p. 1]. Because semantic web technologies enable systematic linking of metadata and thus between different entities, they improve data quality, reduce the effort required for maintenance of metadata [225, p. 381], reveal previously hidden connections [2, p. 108], and allow automated agents to interpret, integrate, and reuse the information.

Design principles The idea of the semantic web is to extend the connectivity of the web by linking data items to one another. When such linked data are published as open, machine-readable resources they form Linked Open Data (LOD). The design of the semantic web rests on three fundamental principles [7, p. 2]: (i) structured and semi-structured data must be offered in standardized formats; (ii) the individual data elements that compose a dataset should be exposed as addressable units; and (iii) the meaning (semantics) of the data must be formalized in a way that machines can interpret. In this context metadata do not merely provide human-readable records but consist of statements that articulate the meaning of each part, also for machines [162, p. 7]. Thereby, metadata become essential building blocks that give the semantic web its "meaning" [7, p. 9]. Three core technologies form the foundation of the semantic web: Uniform Resource Identifier (URI), labeled graphs, and ontologies [7, p. 4].

URIs According to Berners-Lee's guidelines for URIs[25], a URI should serve as a stable name for a thing, be dereferenceable via HTTP, return useful metadata when accessed, and be linkable to other URIs so that additional resources can be discovered. Thereby, URIs serve as globally unique names for resources which can denote any identifiable thing. In research practice, common URIs include the Open Researcher and Contributor ID (ORCID) as identifier for researchers which conveys trustworthy information such as affiliation [2, p. 141], and the Digital Object Identifier (DOI) for digital resources, e.g., scholarly outputs [2, p. 141].

Labeled graphs An important foundation of the semantic web is the graph-centric Resource Description Framework (RDF) that is a data model standardized by the World Wide Web Consortium (W3C)[26]. It represents knowledge as a collection of statements which consist of resources and properties [7, p. 26]. A resource denotes any identifiable

[25] https://www.w3.org/DesignIssues/LinkedData.html, last accessed 2026-03-03
[26] https://www.w3.org/, last accessed 2026-03-03

```
<http://mosaik.offis.de> <http://schema.org/programmingLanguage>
    <http://www.wikidata.org/entity/Q31205855>.
```

Listing 2.1: Small RDF example in turtle where `http://www.wikidata.org/entity/Q31205855` is Python 3 (last accessed 2026-03-03)

```
@prefix schema: <http://schema.org/> .
@prefix wd: <http://www.wikidata.org/entity/> .

<http://mosaik.offis.de> schema:name "mosaik"^^schema:string ;
    schema:programmingLanguage wd:Q31205855.
```

Listing 2.2: Extended RDF example in turtle

thing such as a specific research software like mosaik and is uniquely addressed by a URI [7, p. 26]. Properties are special resources that also have a URI and express relationships to other resources, e.g., the element `programmingLanguage` links a software resource to the language it is written in [113, p. 37].

In RDF terminology a statement (also named triple) consists of a subject (the described resource), a predicate (the property), and an object, which may be either another resource or a literal like a string or a number [7, p. 27]. Listing 2.1 shows an example for such a statement: “mosaik – programmingLanguage – Python 3”. The resource mosaik is represented by a URI. The programming language “Python 3” can be represented as a literal or as shown in the example as another resource by referencing an external PID (in this case a Wikidata entry[27]). By combing multiple connected triples, RDF forms a directed labeled graph [7, p. 29] in which nodes represent entities and edges encode their relationship. The graph can be distributed across the web so that only a subset of statements is locally available while other parts reside elsewhere [7, p. 29].

RDF triples may be expressed in a variety of syntaxes. A widely used easy-readable syntax is Turtle [7, p. 31] which is used for the example in Listing 2.1. In Turtle, literals may be typed like the name mosaik in Listing 2.2 which also shows that common prefixes can be declared to abbreviate repeated vocabularies [7, p. 31–32]. Turtle also permits compact grouping of statements that share the same subject, which enhances readability [7, p. 31].

Alternative syntaxes for RDF include RDFa, RDF/XML, Notation3 (N3), and JavaScript Object Notation for Linked Data (JSON-LD). The latter is favored by programmers and web developers for its seamless integration with JSON-based APIs [162, p. 338]. Regardless

[27] `http://www.wikidata.org/entity/Q31205855`, last accessed 2026-03-03

of the chosen syntax, the underlying RDF graph remains the same, providing a flexible, machine-readable foundation for linking resources, exposing their semantics, and enabling the construction of large-scale graphs.

Ontologies An ontology is a semantic model that allows to formalize abstract concepts in a domain and the logical relationships among them [7, p. 11]. Thereby, it represents a shared understanding of a part of the reality [7, p. 11]. Potentially, there are multiple ontologies for every domain [7, p. 194]. By providing controlled, machine-readable vocabularies, ontologies make it simpler to reason over RDF graphs, to integrate heterogeneous data sources, and to ensure that metadata convey a consistent meaning across different communities. Therefore, ontologies can be used within metadata schemas as introduced in Section 2.5 to semantically describe a resource or to form value vocabularies for certain properties [162, p. 352].

An ontology consists of classes (the allowed types of entities), properties (the relations between those entities), and axioms that constrain how classes and properties may be combined (e.g., subclass, domain-range restrictions, disjointness, value restrictions) [7, p. 10] [225, pp. 61-64]. By assigning a URI and a definition to each class, ontologies resolve the problems of homonymy (the same word with different meanings) and synonymy (different words with the same meaning) through disambiguation and equivalence relationships [162, p. 168].

Ontologies are expressed in standardized web languages such as RDF Schema or the Web Ontology Language (OWL) which both extend RDF. Both use URIs and support reasoning over the defined structures [7, p. 69]. RDF Schema provides basic hierarchical constructs (e.g., `rdfs:subClassOf`, `rdfs:subPropertyOf`) and domain-range specifications. OWL (especially OWL2) extends RDF schema, e.g., by adding richer logical features such as cardinality constraints and class equivalence, enabling more expressive axioms [2, p. 113].

Two principal kinds of ontologies can be distinguished. Reference ontologies are broad and aim to capture general, domain-wide semantics. Application ontologies are smaller and tailored to the specific needs of a particular use case such as a metadata vocabulary [225, p. 61].

Knowledge graphs By combining labeled graphs with ontologies, it is possible to create machine-actionable networks of knowledge. In these networks, all entities have a URI [162, p. 325], ensuring that they can be discovered, described, and interrelated in a consistent, interoperable manner. These labeled graphs are sometimes also called knowledge graphs. The standard query language for such graphs is SPARQL, which is specifically designed to access RDF data [7, p. 69]. Public SPARQL endpoints make existing graphs

accessible on the web.

An example for a labeled graph is Wikidata[28] which provides a large-scale LOD collection of entities such as people, places, and concepts [2, p. 119][225, p. 62]. A further example is the Open Research Knowledge Graph (ORKG)[29] [203] which uses RDF to represent scientific publications, their content, and related metadata [10]. The ORKG allows users to add new properties, suggests existing ones for reuse, and offers domain-specific templates to improve comparability of research outputs [10]. Both examples provide SPARQL endpoints[30] and illustrate how RDF based graphs enable machine-readable, interoperable representations of scholarly information that can be explored and integrated into downstream applications.

2.5 Metadata Schemas

High-quality metadata have to be created in a consistent way [162, p. 12]. A metadata schema provides the formal framework for harmonizing the structure, content, permissible values, and encoding of metadata, thereby making individual descriptions comparable, interoperable, and subject to a minimum quality standard [162, p. 12][225, p. 39]. A metadata schema may also be developed into a standard.

A metadata schema is composed of four tightly coupled aspects:

Structure The metadata structure defines the set of elements that a metadata record may contain [162, p. 15]. These elements constitute the skeleton of the metadata.

Content The data content guidelines contain human-readable guidance for each element: filling instructions, allowed abbreviations, and the distinction between mandatory and optional elements [162, pp. 13-15]. This guidance ensures a uniform usage of the metadata schema.

Value The value specifications restrict the admissible values for certain elements, typically by referencing controlled vocabularies or value encoding schemes [225, p. 25]. By limiting values to a predefined set, the schema reduces ambiguity and improves the consistency of the resulting metadata.

Encoding The encoding specifications specifies the syntactic representation of the metadata, such as XML, RDF/Turtle, JSON-LD, or other formats [162, p. 15]. The chosen encoding determines how the metadata can be exchanged, validated, and harvested by downstream services.

[28] https://www.wikidata.org/wiki/Wikidata:Main_Page, last accessed 2026-03-03
[29] https://orkg.org/, last accessed 2026-03-03
[30] Wikidata: https://query.wikidata.org/, last accessed 2026-03-03, and ORKG: https://orkg.org/sparql/, last accessed 2026-03-03

A metadata schema may integrate all aspects or just specify some of them. It may be organized into modular components which cover a coherent topic or a domain-specific sub-vocabulary [225, p. 63]. Such modularity simplifies the reuse of existing modules, the extension with new terms, and the alignment with community-driven standards, supporting the broader goals of interoperability.

Data structure and content The data structure of a metadata schema consists of elements which form an element set, also referred to as a property vocabulary, metadata vocabulary, or element vocabulary [225, p. 39]. The element set specifies precisely which pieces of information are allowed to describe a particular resource [225, pp. 39-41]. The element set may be expressed as an ontology, providing a formal, machine-readable model of the domain [225, p. 61]. The organization of the elements can follow a flat structure, in which every element is a direct property of the primary resource, or a nested structure, where sub-elements are linked to parent elements according to a defined hierarchy [225, pp. 43-44].

Regardless of the layout, each element carries both structure and content information [225, p. 40]. Table 2.6 shows an example for an element and its relevant information. A typical element includes a clear, machine-readable name that follows a consistent naming convention, a human-readable label, and a unique identifier, usually a URI, that unambiguously distinguishes the element within the vocabulary. The element also has a definition (or description) that articulates its conceptual meaning.

The content information of a metadata schema is normally also organized by elements. Therefore, an element is also accompanied by an obligation flag indicating whether the element is mandatory, recommended, or optional [162, p. 400] and a cardinality specifying

Table 2.6: Example for an element in a metadata schema.

Characteristic	Example
Identifier	`https://schema.org/programmingLanguage`
Label	Programming language
Definition	The computer programming language in which the software is written.
Type	Text
Data value specification	Limited List
Cardinality	n
Obligation	Mandatory

how many times the element may occur in a metadata record [162, p. 401], e.g., a software may have multiple programming languages but only a single name. The data type declares the kind of value the element can hold like string, date, integer, or URI. A data value specification may further restrict the permissible values by referencing controlled vocabularies or value encoding schemes. Together, these characteristics define how metadata records should be populated, validated, and interpreted, ensuring that the records are consistent and usable for different purposes.

Value specifications Value specifications may define the admissible value space for each metadata element [225, p. 26]. Thereby, they constrain what may be recorded for a given element [225, p. 26]. This restriction can be achieved in two complementary ways. On the one hand, a syntax encoding scheme governs the formal representation of simple data types, e.g., dates, ensuring syntactic interoperability across systems. On the other hand, a value encoding scheme addresses more complex values by providing a controlled vocabulary also called value vocabulary. The value vocabulary lists the permissible options and may take the form of a simple list, a synonym ring, a taxonomy, a thesaurus, or a full ontology [162, p. 169]. The latter offers the richest set of semantic relationships among terms. The definitions attached to each term in a controlled vocabulary should be non-redundant, ensuring that every concept is uniquely described [225, p. 192].

Controlled vocabularies improve consistency by enforcing a uniform terminology [225, p. 192]. They serve several interrelated purposes: they guarantee consistent use of concepts, make semantic relationships explicit, support clear labeling and hierarchical browsing, and facilitate reliable retrieval of resources [225, p. 193]. For example, keywords drawn from a controlled vocabulary are better searchable than free-form strings. In RDF-based metadata, literals represent free-form values such as strings, integers, or dates, whereas non-literals are URIs that link to other resources, thereby embedding the element within a broader knowledge graph [225, p. 76]. Existing value vocabularies can be reused, extended, or newly created within a metadata schema, allowing the schema to evolve alongside the domain it describes [225, p. 203].

Encoding a metadata schema The encoding of a metadata schema may turn a human-readable metadata schema into a machine-actionable artifact, enabling automated validation and processing of metadata records [225, p. 238]. Because no single serialization can satisfy all use cases, most schemas are offered in several encodings, each with its own strengths and weaknesses [225, p. 245]. The essential requirement for any encoding is that a machine can automatically check whether a metadata record complies with the schema.

One widely adopted approach is to use JSON-LD together with a JavaScript Object

Notation (JSON) Schema. JSON Schema allows the definition of constraints, like required elements, data types, or pattern restrictions, and supports an automatic validation of these constraints [21]. JSON-LD, a W3C recommendation[31], extends plain JSON with linked data capabilities by attaching a context file that maps JSON keys to URIs in the semantic web [21]. In this way the resulting metadata are both human-readable and machine-actionable while existing JSON tooling can be reused without modification.

A complementary, more expressive option is the Shapes Constraint Language (SHACL), also a W3C recommendation[32]. SHACL is built on top of RDF and can be written in any RDF syntax. While RDF's flexibility can lead to data quality problems, SHACL allows to formulate constraints that make it possible to enforce both structural and semantic rules [169]. It defines shapes that act as templates and describe the expected structure of an RDF graph and the semantic constraints that it must hold, e.g., allowed data types, cardinalities, or limitations to controlled vocabularies [41, 177]. SHACL allows to limit a value to a certain ontology class. Consequently, SHACL is well suited for encoding complex metadata schemas where precise validation of element types and complex semantic value vocabularies are required.

Application profiles Application profiles provide a pragmatic way to reuse existing metadata schemas without having to create an entirely new schema. Thereby, they allow to increase the interoperability of the metadata. Rather than defining a complete element set from scratch, an application profile combines selected modules or individual elements drawn from one or more established schemas, thereby tailoring the metadata to the specific needs of a given domain or use case [225, p. 54]. In the strict sense, an application profile only combines elements from recognized standards [162, p. 394] while a broader interpretation allows the inclusion of new elements when the existing element vocabularies are insufficient [225, p. 54]. Frequently, an application profile also refines a single base schema by adding extra guidance like more detailed definitions, usage recommendations, or additional controlled value vocabulary [225, p. 55]. The development and methodological considerations for constructing application profiles are discussed in detail in Section 4.2.

Crosswalks To improve the interoperability between different metadata schemas, crosswalks can map the elements of one metadata schema to those of another one [225, p. 202]. By establishing a correspondence between elements, a crosswalk makes it possible to translate metadata records from one schema into a different one, which is essential for exchanging metadata across heterogeneous systems. Two principal strategies may be used to construct

[31] https://www.w3.org/TR/json-ld11/, last accessed 2026-03-03
[32] https://www.w3.org/TR/shacl/, last accessed 2026-03-03

a crosswalk. (1) An absolute crosswalk requires an exact one-to-one match. If a source element has no precise counterpart in the target schema, the element is omitted. (2) A relative crosswalk attempts to map every source element, selecting the closest possible match when an exact equivalent does not exist [225, p. 207]. In practice, precise one-to-one correspondences are rare and most crosswalks are directional because the mapping is not invertible [113, p. 178]. This asymmetry can lead to information loss, especially when a single concept in one schema is expressed by multiple elements in the other, when the allowed data types or value vocabularies differ, or when no suitable counterpart exists at all [162, p. 299]. Consequently, while crosswalks are a valuable tool for achieving semantic interoperability, their design must carefully address these limitations to minimize information loss during metadata conversion.

Commonly used metadata schemas *Dublin Core* is one of the most widely adopted metadata standards which was first introduced in 1995 [225, p. 45]. Its development and maintenance are coordinated by the Dublin Core Metadata Initiative (DCMI) [162, p. 65]. The original core consists of fifteen elements that are deliberately generic [225, p. 45]. The schema is flat and every element is optional and repeatable which makes *Dublin Core* suitable as a general-purpose metadata framework [225, p. 45]. Since 2003 the elements are expressed as an RDF vocabulary (the DCMI Metadata Terms) with distinct URIs for each term, allowing metadata to be published as linked data [162, p. 323][225, p. 74]. For each element the standard indicates whether the value must be a literal, a non-literal, or may be either, thereby providing clear guidance on permissible value types [225, p. 78].

Schema.org[33] originated from a joint initiative of multiple search-engine providers (Bing, Yahoo, Google, and Yandex) to establish a common vocabulary that webmasters can use to annotate their web pages in a machine-readable way [162, p. 385][225, p. 65]. By exposing metadata that conforms *Schema.org*, content becomes more discoverable for search engines and other automated agents [225, p. 68]. The development of *Schema.org* follows a collaborative, community-driven model, and the vocabulary continues to grow as new use cases emerge [225, pp. 65-68]. The vocabulary defines a hierarchy of resource types, like `Person`, `CreativeWork`, `Event`, and many others [225, p. 68] where Table 2.7 provides a short overview. For each type, there is a set of properties that can be used to describe the resource [225, p. 68]. Sub-types inherit the properties of their ancestors while adding additional, more specific elements [225, p. 66]. Every element has an expected value type which may be another *Schema.org* type or a primitive data type (e.g., `Text`, `Boolean`, `Date`, `DateTime`, `Number`, `Time`) [225, p. 65]. Together, these features make *Schema.org* a flexible,

[33] `https://schema.org/`, last accessed 2026-03-03

Table 2.7: Overview of resource types and their definitions in *Schema.org*.

Thing	**Definition**
CreativeWork	The most generic kind of creative work, including books, movies, photographs, software programs, etc.
Organization	An organization such as a school, NGO, corporation, club, etc.
Person	A person (alive, dead, undead, or fictional).

widely adopted framework for publishing structured, searchable metadata on the internet.

The *Data Catalog Vocabulary (DCAT)* is a W3C recommendation[34] first published in 2014 [225, p. 446]. It provides a standardized vocabulary for describing datasets and data services, thereby supporting the discovery and reuse of data across organizations. The specification reuses elements from fifteen existing namespaces, including *Dublin Core* [225, p. 446]. Therefore, it can be integrated with a broad range of metadata ecosystems. *DCAT* defines six core classes: `Catalog`, `Resource`, `Dataset`, `Distribution`, `DataService`, and `CatalogRecord` [225, p. 446]. As an RDF vocabulary, *DCAT* can be expressed in any RDF serialization and is linked to the *Schema.org* ecosystem through an explicit mapping [225, p. 448]. In practice, a number of application profiles have been derived from *DCAT* for specific countries and domains [225, p. 449]. These profiles often restrict the allowed value spaces or cardinalities of particular elements to meet local requirements [225, p. 449].

This chapter introduced the fundamentals for metadata (Section 2.3), semantic web (Section 2.4), and metadata schemas (Section 2.5) which are relevant for the investigation of a metadata schema in Chapter 4 and the relevant tooling in Chapter 5. Also, this chapter presented an overview of the current state of RSE and FAIR4RS which are further used for developing RSE recommendations in the energy domain in the next chapter (Chapter 3).

[34] `https://www.w3.org/TR/vocab-dcat-3/`, last accessed 2026-03-03

3 Recommendations for Engineering Energy Research Software

As shown in Section 2.1, a relevant part of RSE literature concentrates on providing recommendations for RSE practices to researchers and research software engineers. Especially published recommendations with a domain focus provide relevant domain-specific support while also fostering cultural change towards better RSE in the related research domain. For energy research, there was no comparable publication. Therefore, RQ1 focuses on recommendations for the development of ERS:

RQ1: Which recommendations can help energy researchers to develop more reusable ERS?

The goal is to identify the most important recommendations that can help energy researchers write more reusable ERS. The recommendations should be understandable for a broad audience, cover all relevant aspects, be practicable, and serve as a starting point for readers who want to learn more about the aspects.

Within this chapter, Section 3.1 first presents a review of related work and especially existing recommendations for general RSE and RSE in different research domains. Afterwards, Section 3.2 outlines the method used to derive recommendations. Then, Section 3.3 gives an overview of the derived recommendations. Subsequently, Section 3.4 shows the analysis of the recommendations based on the evaluation of understandability and on comparing them to the existing recommendations from the related work. Finally, Section 3.5 discusses the results.

Parts of this chapter are already published in [77] (see Appendix A for detailed descriptions of the author's contributions to this publication).

3.1 Related Work

Several existing publications provide recommendations or best practices on RSE. To give an overview, the following types of publications were covered: publications providing recom-

mendations, publications on the general state of RSE, and publications on FAIR principles for research software.

Multiple publications can be found regarding recommendations for research software in general [3, 8, 9, 11, 22, 32, 40, 44, 62, 65, 189, 205]. Additionally, as introduced in Section 2.1, there are also institutional policies for RSE in some institutions, e.g., in the German Aerospace Center (DLR) [190]. Additionally, some works consider best practices for RSE in specific domains, for example in computational biology [155]. Others focus on particular aspects of RSE, e.g., structuring research software [26], specific programming languages such as Python [126], the documentation of software [114, 153], testing research software [63], sharing and reusing research software [171], sustainability of research software [185], or reproducibility of research software [204]. The aspect of open source development is discussed as well [110, 227].

Additionally, there is the large body of publications in software engineering research which is also relevant here, e.g., [135, 216]. Some publications also examine particular aspects in this area that might be applied to developing ERS, such as robust software [208] or code quality [210].

The publications discussing FAIR principles for research software [18, 100, 109, 143], as introduced in Section 2.2, also cover different RSE best practices, e.g., registering research software in a software registry. Also, from publications focusing on quality indicators for research software, as introduced in Section 2.1, like [49], some recommendations for RSE can be derived.

Based on the extensive research, four clusters were identified which are frequently discussed in the literature:

Architecture and software design This cluster includes the structure and reuse of existing software. The literature recommends adopting community standards for input and output data [32], providing a comprehensible, extensible, modular, and exchangeable structure, as well as employing established design patterns, and following consistent programming-style guidelines [36] and iterative processes regarding requirements [134].

Development process This cluster includes the development process itself, including organization, environment, documentation, and code reviews. Specific advice includes the use of version control systems [32], structuring repositories according to community standards [32], performing continuous refactoring and testing [36], and choosing informative file-naming conventions [210].

Testing This cluster addresses test strategies and the variety of test types. Authors advocate automated testing [32, 36], the use of unit, integration, system, and acceptance

tests [36], end-to-end testing [123], systematic test strategy planning, and defining and ensuring test coverages [36].

Publication of software This cluster includes aspects as licenses and the interaction with the community. The literature recommends publishing source code [20] and also developing open source [110]. Reproducibility and reusability of code are emphasized [100], together with the recommendation to provide everything required for reproducibility [204]. Appropriate licensing [32], the assignment of Persistent Identifiers (PIDs) [32], and the provision of well-structured metadata [32] are also highlighted.

Although much literature discusses general topics, none of the existing works derives a unified set of recommendations specifically for RSE in the energy domain. While some recommendations from other domains may be usable for the energy domain, ERS has some unique characteristics as described in Section 1.1, like its high focus on simulations across different expertise fields and its development close to industry. To close this gap, the goal is to develop recommendations for engineering ERS together with energy researchers. The outcome should be a single set of recommendations that covers the most important aspects of RSE in the energy domain. It should incorporate domain-specific requirements and concrete examples, such that the reader can gain a good overview by reading a single set of recommendations. These recommendations should offer a practical starting point for energy researchers to improve their software engineering skills and develop better ERS.

3.2 Method

This section outlines the exploratory approach to identify, organize, and prioritize recommendations. The overall workflow is shown in Figure 3.1 and follows a method-triangulation strategy [30] based on three main sources. Each step is detailed in this section after providing a general overview.

First, four clusters were identified based on the literature as described in Section 3.1 (see Table 3.1). These clusters were used as the foundation for a first exploratory workshop. From this workshop, a first set of recommendations was derived (see Box 3.1) which were the starting point for a second consolidating workshop. The results of both workshops were combined to define the detailed recommendations. Finally, a first draft of the recommendations was tested for understandability, usability, and accessibility leading to the final recommendations.

Generally, the focus is on ERS for modeling, simulation, data analytics, and technology research software [111]. Infrastructure software was deliberately excluded because it is driven by a different set of usability-centric requirements.

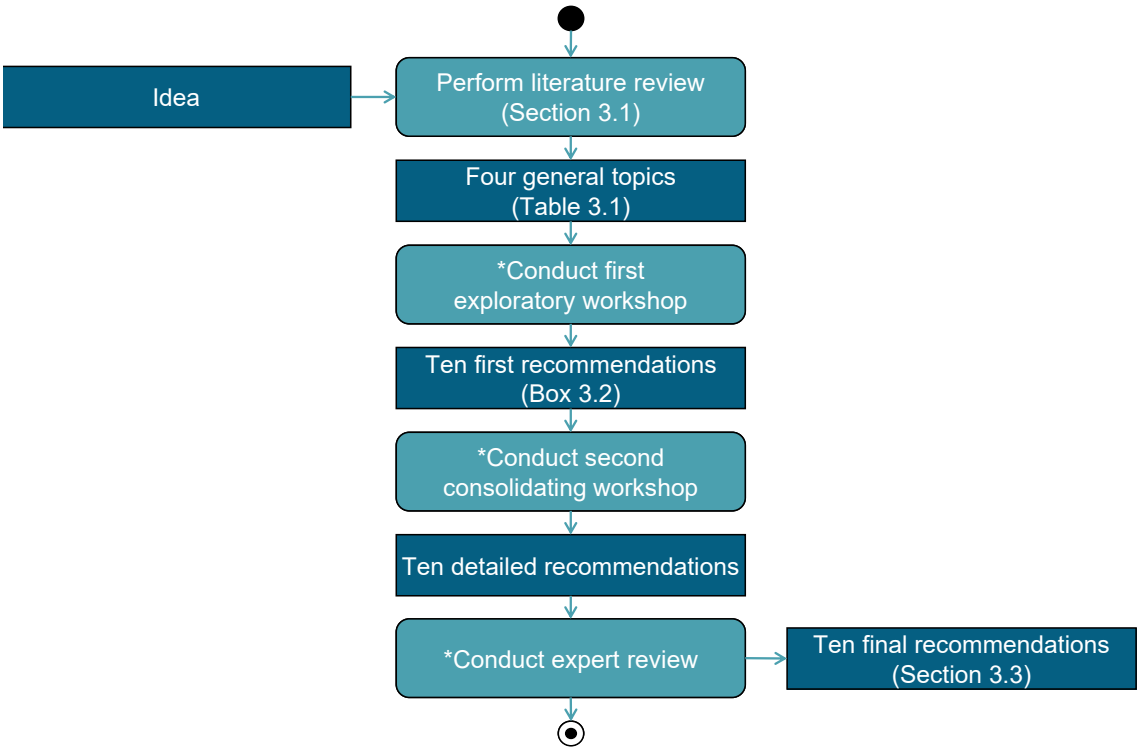


Figure 3.1: Overview of the method to develop the recommendations as Unified Modeling Language (UML) activity diagram. Light-blue round boxes contain activities where expert interactions are marked with *. Dark-blue rectangular boxes contain (interim) artifacts and inputs.

First exploratory workshop The first workshop took place at the Energy Division of OFFIS[1] in July 2024. Eight participants (one professor, five PhD students, and two research staff) representing all energy-related research groups at the institute attended. All of them had a high software engineering focus within their different research groups. The workshop was conducted mainly in English, with occasional informal German exchanges, and all outcomes were recorded directly on whiteboards.

The workshop comprised three phases. An introductory round highlighted the general challenges of developing ERS and invited participants to voice initial ideas without any preconceptions. In the second phase, the participants were divided into random small groups which discussed sequentially all pre-identified clusters. Each cluster was moderated by an expert from the author team of [77]. The groups were provided with the cluster topic and related keywords as shown in Table 3.1. They discussed detailed ideas for relevant

[1]OFFIS - Institute for Information Technology, Energy Division, https://www.offis.de/en/applications/energy.html, last accessed 2026-03-03

Table 3.1: Input for the first workshop: four clusters and related keywords.

Cluster	Related Keywords
Architecture and software design	Interoperability, extendability, reusability
Publication	Licensing, community, reproducibility
Testing	(no keywords provided)
Development process	Documentation, version control, Continuous Integration/Continuous Delivery (CI/CD), code-review

energy-specific aspects of the clusters which were captured on cards. In the final phase, each participant selected four cards they considered most important, thereby revealing the priority areas most relevant for the recommendations. Based on the outcome of the workshop, a first draft of ten recommendations was formulated as shown in Box 3.1.

Second consolidating workshop To broaden the perspective, a second workshop was organized in November 2024. Twenty external participants from thirteen German research institutions attended. The recruiting was conducted in two ways. Representatives of prominent ERS projects were specifically invited. Additionally, members of the Energy Research Centre of Lower Saxony (EFZN)[2] were invited. The workshop was mainly held in English, while many participants discussed in German.

- Test your software based on a test strategy
- Documentation with enough time
- Reuse and extend other software if possible and useful
- Use a project management that takes the size and skills of the involved team members into account
- Consider reuse
- Keep the architecture as simple as possible
- Use version control
- Develop open source and use a license
- Use platforms where software can be easily found
- Explicitly consider the RSE process in your project proposal

Box 3.1: First preliminary set of ten recommendations

[2] https://www.efzn.de/en/, last accessed 2026-03-03

The workshop was structured into four phases. It began with a keynote by a software engineering professor (Anna-Lena Lamprecht) that gave a general overview of different aspects of RSE. The presentation set the stage for the subsequent activities. Afterwards, the invited researchers presented their most relevant ERS in a poster session, allowing the audience to identify and discuss concrete challenges. Then, the participants were divided into four random groups. Each group was asked to discuss all ten draft recommendations as shown in Box 3.1 where only the headings were given to allow free associations. Again, the discussions were moderated by four experts from the author team of [77] to ensure focus and productivity. The groups recorded their associations and suggestions on cards. An empty board was used to capture aspects which have not been discussed before. In the final phase, each participant selected ten cards to prioritize the most pressing challenges and innovative solutions. This voting mechanism helped to identify the key areas relevant for further development of the recommendations. The collective feedback from both workshops was synthesized into the detailed set of recommendations.

Expert review As the target audience of the recommendations is highly heterogeneous (see Section 1.1), an expert review was performed to improve the understandability of the detailed recommendations. Thereby, the clarity and the practical relevance of the formulated recommendations were assessed. Six colleagues (three with a computer-science background[3] and three with an electrical-engineering background[4]) read the detailed recommendations and answered four guiding questions adapted from [176] as summarized in Box 3.2.

Their written feedback (e.g., reformulating specific aspects for better understandability) was used to improve the detailed recommendations to the final recommendations which are presented in Section 3.3.

- Which relevant aspects of energy-domain research software are missing?
- How usable and applicable are the recommendations for your own work?
- Are the recommendations easy to understand?
- Does the level of detail seem appropriate, and why (or why not)?

Box 3.2: Questions for expert review adapted from [176].

[3] from OFFIS - Institute for Information Technology, Energy Division, https://www.offis.de/en/applications/energy.html, last accessed 2026-03-03

[4] from Leibniz Universität Hannover, Institute for Electric Energy Systems (IfES), https://www.ifes.uni-hannover.de/en/, last accessed 2026-03-03

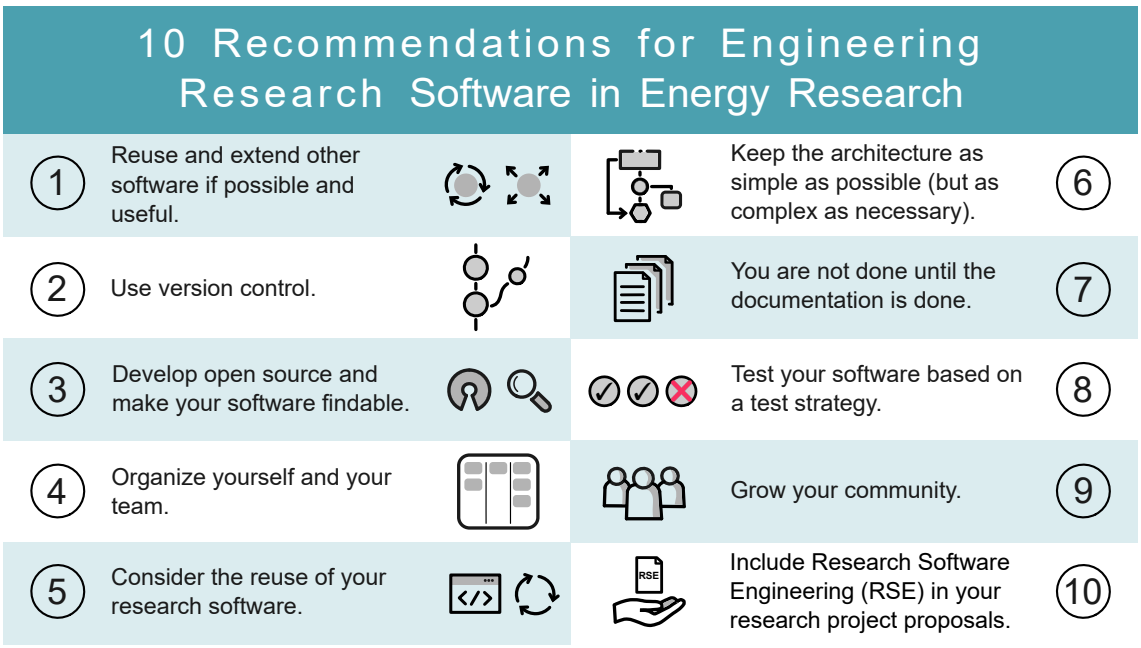


Figure 3.2: Overview of all ten final recommendations.

3.3 Recommendations

Ten recommendations for engineering ERS were derived based on the method described in Section 3.2. Figure 3.2 provides a general overview of the final recommendations. This section describes the general structure and topics of the recommendations. The detailed recommendations for energy researchers are published in [77]. The recommendations were also made available as a moderated GitHub repository[5] where researchers can discuss the recommendations and provide feedback.

Each recommendation consists of two main parts. First, the background on the topic motivates why the recommendation matters in the context of ERS. Afterwards, concrete, actionable advice for the topic is given. This part also covers energy-specific characteristics like working with industry. Many recommendations also include energy-related examples. Although each recommendation offers a practical starting point, readers of the recommendations are also encouraged to explore additional material, such as tutorials to deepen the aspects which are most relevant for them.

The recommendations are ordered by the effort required for their implementation. The first items can be incorporated into everyday workflows with little effort, while the later items typically demand more resources and are especially relevant for larger software

[5] https://github.com/NFDI4Energy/ERSE_Recommendations, last accessed 2026-03-03

projects or for senior researchers who are preparing research project proposals.

The first recommendation (**"Reuse and extend other software if possible and useful."**) addresses reusing ERS for the researcher's project. It recommends to reuse existing software wherever possible instead of building everything from scratch. Thereby, it highlights how existing, well-tested code can reduce required development time, improve software quality, and foster collaboration. The recommendation also acknowledges the bounded exceptions in which new implementations become justified, namely when existing tools are obsolete, of low quality, or locked behind proprietary barriers. The recommendation provides practical energy-specific advice where to find existing ERS. Additionally, it highlights relevant criteria for reuse such as license compatibility and emphasizes collaborating with the developers of the existing ERS.

The second recommendation (**"Use version control."**) focuses on keeping control of different software versions. It motivates the need for version control based on a practical example from energy research and explains typical terms in version control systems like *branch* and *merge request*. Also, the recommendation underlines the importance of meaningful commit messages. It emphasizes the use of git-based platforms like GitHub and GitLab and hints at also using their additional capabilities, like issue tracking and CI/CD.

The third recommendation (**"Develop open source and make your software findable."**) stresses that research software should belong to the public domain of science and motivates open source development. This recommendation includes information on open source licensing that makes the code freely reproducible and reusable. It also discusses how to handle open source in cooperation with industry partners which is often relevant for energy research projects. Additionally, the recommendation provides hints on how to design a repository for better reusability, like using repository templates and providing a *Citation File Format (CFF)* file. It also covers the topic of registering the software in a domain-specific software registry, archiving the software, and minting a PID to make it discoverable and citable. In this way, this recommendation covers multiple aspects relevant for FAIR ERS.

The fourth recommendation (**"Organize yourself and your team."**) addresses the strategic organization of ERS projects and their teams. It stresses that a clear definition of scope, purpose, and intended user base must be established early, because these factors drive all subsequent decisions. It also highlights the importance of assessing the available team, its size, and the members' skill levels, since these determine the appropriate development process. Due to the interdisciplinarity of many energy research projects, the team members often have different backgrounds and experiences. The recommendation acknowledges that no single organizational model fits every project. Instead it recommends to take all the mentioned aspects into account to explicitly organize the development.

The fifth recommendation (**"Consider the reuse of your research software."**) focuses on different aspects to improve the reusability of the software besides being open source. First, this recommendation highlights how the choice of the appropriate abstraction level for the software can simplify reuse. It stresses that software must be neither overly specialized nor needlessly generic. Instead, it is advised to find a sensible balance making the code understandable, easy to adopt, and still flexible enough to address related problems. Additionally, it is recommended to follow community-specific conventions and ideally integrate the code into established open source ecosystems where energy-specific examples like oemof[6] are named. Additionally, the recommendation highlights the importance of a good name and keywords, e.g., based on community-approved ontologies, to increase the visibility of the software.

The sixth recommendation (**"Keep the architecture as simple as possible (but as complex as necessary)."**) stresses that the way components are organized and the design patterns that connect them determine how easily new features can be added, how quickly bugs can be located, and how readily the code can be reused by others. Therefore, the recommendation underlines the strategic role of the software architecture to increase the scientific impact of research software. It is advised to choose a simple, well-defined architecture and carefully document it including the reasoning for architectural decisions. The recommendation highlights the importance of community-adopted interface standards for APIs to keep the system maintainable and interoperable and it lists some of these standards.

The seventh recommendation (**"You are not done until the documentation is done."**) covers the role of documentation as a prerequisite for any ERS to be usable, reproducible, and expandable. It stresses that code without clear explanations is effectively unusable for other researchers and, therefore, contributes little to scientific progress. The recommendation emphasizes the use of different documentation types (code documentation, README file, documentation pages) depending on the size of the ERS. It also links documentation to other good practices such as open source licensing, version control, and automated testing. Finally, the recommendation stresses the need to plan enough resources for documentation, to define minimal documentation requirements, and to use readily available tools for better documentation.

The eighth recommendation (**"Test your software based on a test strategy."**) concerns the role of systematic testing as a foundation for trustworthy ERS. This recommendation stresses that testing is not an optional but a core activity that guarantees code behaving as intended and results remaining reproducible. To motivate the topic, the

[6] https://oemof.org/, last accessed 2026-03-10

recommendation includes energy-specific examples for testing, e.g., ensure that an efficiency is less than one. It advocates moving from ad-hoc, manual sanity checks to a test strategy that is early defined and documented. Regarding automation, the recommendation also provides hints how to embed tests in continuous-integration pipelines and, therefore, enable test automation. To clearly define required testing, different tests types are introduced like unit, integration, and end-to-end tests. Overall, the recommendation advises to allocate enough resources for testing.

The ninth recommendation (**"Grow your community."**) focuses on making the ERS more known and on establishing a user community which is especially relevant for larger ERS. It motivates that these steps are important to increase the scientific impact of the software. The recommendation advises to publish a software paper (in a software journal) and to present the ERS at conferences to increase its visibility where also an example for an energy-specific conference is given. At the same time, the recommendation shows the importance of active community engagement like responding to issues and providing a clear contribution guide to turn passive users into collaborators.

The tenth recommendation (**"Include RSE in your research project proposals."**) stresses the importance of research project proposals to get enough dedicated resources to achieve high-quality ERS and to follow the other recommendations. This recommendation advises to explicitly allocate time and budget for activities such as reusing existing frameworks, maintaining and adapting code, testing, documentation, licensing, and community engagement. When industry partners are involved, it is also advised to include resources for legal support and for negotiating joint licensing arrangements in the proposal. The recommendation also introduces Software Management Plans (SMPs) to systematically describe the software development and to argue for the required funding to develop reusable and sustainable ERS. Additionally, the recommendation advises to also include budget for networking opportunities and training because many researchers have limited RSE expertise and need support to deliver robust high-quality ERS.

3.4 Analysis of Resulting Recommendations

The ten detailed recommendations were evaluated from two complementary perspectives.

First, the understandability of the recommendations was examined by asking a group of colleagues to review them (see Section 3.2). The participants judged the recommendations to be understandable, clear, and applicable to their own research. Most of them liked the level of detail although one participant requested more details, while another requested less details probably both as a result from their own background. Based on their comments a few formulations were refined leading to the final recommendations.

Table 3.2: Comparison of the developed recommendations for developing ERS with existing recommendations.

✓: Includes similar recommendation ✗: Does not include similar recommendation

Final Recommendations	[11]	[123]	[133]	[155]	[168]	[178]	[208]	[221]
1: Reuse and extend other software if possible and useful	✓	✗	✗	✓	✓	✓	✓	✓
2: Use version control	✗	✗	✗	✗	✓	✗	✓	✗
3: Develop open source and make your software findable	✗	✗	✓	✗	✓	✓	✗	✗
4: Organize yourself and your team	✗	✗	✗	✗	✗	✗	✗	✗
5: Consider the reuse of your research software	✗	✗	✗	✓	✓	✗	✗	✗
6: Keep the architecture as simple as possible (but as complex as necessary)	✓	✓	✗	✗	✗	✓	✗	✗
7: You are not done until the documentation is done	✓	✗	✗	✗	✗	✗	✓	✓
8: Test your software based on a test strategy	✗	✗	✗	✓	✓	✗	✗	✗
9: Grow your community	✗	✗	✓	✗	✗	✓	✗	✗
10: Include RSE in your research project proposals	✗	✗	✗	✗	✗	✗	✗	✗

Second, the recommendations were compared with the existing literature. As introduced in Section 3.1, some papers focused only on one specific aspect of RSE like [26, 153, 175, 184, 186, 189]. These papers were excluded from the comparison since they focused on a different level of detail. When comparing with literature, it was analyzed which of the developed recommendations are also covered in existing recommendation papers. Table 3.2 shows an overview of the comparison. A more detailed overview of the comparison is provided in Appendix B. The recommendations concerning the organization of the development team ("Organize yourself and your team") and the inclusion of RSE aspects in research proposals ("Include RSE in your research project proposals") do not appear in any of the surveyed works. Therefore, they constitute novel contributions. The remaining

recommendations overlap with the ones found in literature but differ in the details. Logically, other publications do not include energy-specific examples due to their different focus. Additionally, other works rarely address collaboration with industry and interdisciplinary team structures.

3.5 Limitations and Discussion

As highlighted in Section 3.4, the recommendations align with existing literature but differ in their energy-specific focus, attention to industry collaboration and interdisciplinary teamwork, and the inclusion of RSE in project proposals.

Like all studies, the used method has some limitations that should be considered. First of all, the second workshop was conducted in-person in Germany which may have narrowed the geographic diversity of the perspectives captured. Although this allowed more personal engagement and interactions without technical issues of an online setting. Therefore, it enabled deeper and more fluent discussions. Second, while the participants already comprised an interdisciplinary group, not all possible perspectives may have been captured due to the inherently interdisciplinary nature of energy research. Third, the participants were primarily practical energy researchers with strong programming skills and some of them were trained computer scientists. They provided valuable context-specific practical insights based on their own experience but may have limited awareness of the latest software engineering technologies. Fourth, the diversity of ERS projects means that the recommendations have different relevance depending on the context. Therefore, researchers need to tailor their use to each project's specific requirements and the type of research software.

The resulting recommendations highly depend on existing infrastructure for ERS and research software, like software registries, software journals, and other services, which are introduced in Section 1.2. When further services and standards become available, the recommendations should be adjusted. In this step, further community feedback can also be included which is collected in the GitHub repository of the recommendations[5].

The workshops revealed additional important themes beyond the core recommendations for single researchers. Some participants emphasized the value of regular informal exchanges on RSE within institutions. The topic of training also emerged in the workshops. Many energy researchers lack formal software engineering training, yet skills in version control, testing, clean code, and basic architectural knowledge are essential. These training needs should be addressed at the beginning of projects potentially through internal experts or

external providers such as the Digital Research Academy[7].

Another significant theme that emerged from the workshops was the undervaluation of research software as a scientific contribution which was also pointed out in multiple publications as shown in Section 2.1. Research software is often viewed merely as a tool for generating research results rather than as a valuable contribution in its own. However, well-written and documented software can have substantial scientific impact through direct reusability and extension by other researchers. The scientific community should place greater value on research software development alongside traditional publications.

Most of these challenges are similar for all types of research software, where a general overview was given in Section 2.1. Nevertheless, a domain-specific discussion of these aspects is an important step to foster cultural change towards increased quality of ERS which can improve the quality of energy research. Overall, the recommendations serve as a starting point to better understand ERS and improve its engineering in energy research towards FAIR ERS.

[7] https://digital-research.academy/, last accessed 2026-03-03

4 Metadata Schema for Energy Research Software

Besides practical guidance, achieving FAIR ERS requires consistent and comprehensive descriptions of ERS through rich, standardized metadata. Thus, this chapter addresses RQ2 of this thesis:

> **RQ2:** Which metadata are suited to improve the FAIRness of ERS and how can they be structured within a metadata schema?

As introduced in Section 2.5, the foundation of good metadata lies in a well-designed metadata schema that establishes a shared, structured framework for capturing relevant information in a consistent way. Therefore, this chapter focuses on a metadata schema for ERS. The metadata schema should be both effective in enhancing the findability and interoperability of ERS and practically feasible for energy researchers to use.

To guide the research on the metadata schema, a set of Core Metadata Requirements (CMRs) is derived from the foundational concepts in FAIR research software, metadata management, and metadata schema design, as introduced in Chapter 2. The CMRs are summarized in Table 4.1. From a functional perspective, the metadata schema must capture sufficient energy-specific information (see Section 2.2) to enable robust findability across the diverse landscape of ERS (CMR1). The schema must also incorporate all information required to fulfill the FAIR4RS principles, such as licensing information (CMR2). Meanwhile, the schema should also be as simple as possible to allow widespread adoption and ease of use [225, p. 180]. It is important to ensure that the schema is accessible and usable for energy researchers, with clear guidance, minimal burden, and an ensured quality, e.g., through mandatory elements (CMR3). Another relevant aspect of the metadata schema is interoperability (see Section 2.5), which leads to multiple requirements: (1) The schema should allow for semantic, syntactic, and functional alignment with other metadata standards either by reusing established elements or enabling crosswalks to existing schemas [225, pp. 23-29] (CMR4). (2) To ensure consistency and machine-actionability, many metadata values should be drawn from controlled vocabularies to improve precision [225, pp. 23-29] (CMR5). (3) Furthermore, the schema should be formalized using

Table 4.1: Core Metadata Requirements (CMRs) for a metadata schema for ERS. The order of the requirements does not reflect relevance.

ID	Requirement
CMR1	Capture sufficient energy-specific information.
CMR2	Incorporate all information required for FAIR4RS.
CMR3	Be feasible for energy researchers.
CMR4	Align with existing standards.
CMR5	Use controlled vocabularies.
CMR6	Use widely adopted technologies for its formalization.

widely adopted (semantic web) technologies to ensure compatibility with existing tools and workflows (CMR6).

Based on these CMRs, an analysis of existing approaches is outlined in Section 4.1. Since none of the existing approaches fulfills all requirements, Section 4.2 introduces the methodology to develop a metadata schema for ERS. An important step of the methodology is an analysis of the requirements which is presented in Section 4.3. The subsequent development, described in Section 4.4, iteratively shapes the schema through design, validation, and refinement. Finally, the full metadata schema, called *ERSmeta*, is presented in Section 4.5.

Parts of this chapter are already published in [84, 90] and other parts are included in a paper that is currently under review, available as preprint [91] (see Appendix A for detailed descriptions of the author's contributions to these publications).

4.1 Related Work

This section presents an overview of existing approaches to metadata schemas for research software. While some are explicitly designed as metadata schemas, others are ontologies that can serve as element vocabularies for metadata. The approaches vary significantly in their use of metadata schema capabilities, such as defining mandatory elements, supporting URIs, and integrating ontologies as value vocabularies to ensure semantic consistency and interoperability. As a result, they address the presented CMRs to a different extent. Table 4.2 summarizes the main characteristics of the reviewed approaches.

Table 4.2: Overview of existing metadata approaches for research software.
✓: fulfilled (✓): partly fulfilled ✗: not fulfilled

Types	Specific Approaches	Scope	Aligns with existing approaches (CMR4)	Uses value vocabularies (CMR5)	Formalized through widely adopted technologies (CMR6)
Metadata Schemas	*CodeMeta*[1] [136]	General	✓	✗	✓
	Citation File Format (CFF) [60]	General	✗	✗	✗
	DataDesc [150]	General	✓	✗	(✓)
	biotoolsSchema [128]	Bioinformatics	✗	✓	✓
Ontologies	*OntoSoft* [98]	Geoscience	✗	✗	✓
	Software Description Ontology[2] [96]	General	✓	✓	✓
Not formalized approaches	Catalog of energy co-simulation components [192]	Energy	✗	(✓)	✗
	openmod[3]	Energy	✗	(✓)	✗
	OEP factsheets[4]	Energy	✗	(✓)	✗

Metadata for research software without an energy scope Generally, package registries for different programming languages like PyPi[5] for Python include some metadata for the software packages. Since their underlying metadata schemas are always language specific, they are out of scope for this overview.

CodeMeta[1] [136] is a community-driven metadata standard for research software. It is built on *Schema.org*[6] and draws from existing practices across different software reposito-

[1] https://codemeta.github.io/, last accessed 2026-03-03
[2] https://w3id.org/okn/o/sd, last accessed 2026-03-03
[3] https://wiki.openmod-initiative.org/wiki/Open_Models, last accessed 2026-03-03
[4] https://openenergy-platform.org/factsheets/models/ and https://openenergyplatform.org/factsheets/frameworks/, last accessed 2026-03-03
[5] https://pypi.org/, last accessed 2026-03-03
[6] https://schema.org/, last accessed 2026-03-03

ries. Its development is still ongoing through an active community and public discussions on GitHub. *CodeMeta* is formalized as a JSON-LD Schema and includes a broad range of elements covering technical details (e.g., file size, supported operating systems), administrative metadata (e.g., license, version), and contributor information. While allowing free-text for most elements, it supports the use of URIs for authors, contributors, and licenses, enhancing machine readability. However, *CodeMeta* does not enforce mandatory elements and its domain-specific content is limited to application categories and keywords. *CodeMeta* provides multiple crosswalks, e.g., to the *Data Catalog Vocabulary (DCAT)*.

The *Citation File Format (CFF)* [60] is another approach which aims to enable proper citation of research software. While it captures essential information such as authorship, version, and license, it is not a full metadata schema but rather a lightweight format for citation metadata. As such, it lacks structured definitions for its metadata elements and does not include domain-specific or technical details, limiting its utility beyond citation.

Kuckertz et al. [150] introduced *DataDesc* as metadata schema that focuses on interfaces of research software. This schema reuses multiple elements from *CodeMeta* and allows for a detailed description of research software interfaces. It does not cover detailed domain-specific aspects besides keywords and description.

In contrast, domain-specific approaches often allow providing richer metadata. Ison et al. [128] introduced *biotoolsXSD* as a metadata schema in life sciences, developed for the registry bio.tools[7] [127]. It is a formally defined XML schema with 55 metadata elements of which ten are mandatory. *biotoolsXSD* leverages the *EDAM* ontology[8] as a controlled vocabulary for key elements such as function, input, and output, ensuring consistency and interoperability. The schema also includes technical metadata like programming language, license, and operating system. However, the use of ontologies or URIs is not required for these. The schema was shaped through community workshops, reflecting a participatory design process [128].

Gil et al. [97, 98] introduced *OntoSoft* as an ontology for research software, initially tailored to geosciences. For its development, they performed informal surveys [97] without providing more details on their method. *OntoSoft* is organized in six thematic categories: identify, understand, execute, do research, get support, and update. Thereby, it covers both technical and descriptive aspects such as dependencies, contributors, and execution environment. While *OntoSoft* allows free-text input for many elements to ease adoption, this flexibility reduces the semantic interoperability of the created metadata. The creators of *OntoSoft* also developed a registry[9].

[7] http://bio.tools, last accessed 2026-03-03
[8] https://edamontology.org/, last accessed 2026-03-03
[9] https://www.ontosoft.org/portal/, last accessed 2026-03-03

Further advancing semantic integration, Garijo et al. [96] expanded *OntoSoft* by creating the *Software Description Ontology*[2]. Their development process was guided by competency questions, ensuring alignment with real needs. The *Software Description Ontology* aligns with *CodeMeta* and incorporates the *Scientific Variables Ontology*[10] to describe input and output data, particularly relevant for scientific models. It enables rich linking to external knowledge graphs such as Wikidata[11] and supports publishing metadata in an open knowledge graph. Kelley and Garijo [147] presented a tool for metadata extraction that uses the *Software Description Ontology* (SoMEF).

Overall, the existing approaches reflect a spectrum of design philosophies: from lightweight, simple formats like *CFF* over larger approaches like *CodeMeta*, *biotoolsXSD*, and *OntoSoft* to rich, ontology-driven frameworks such as the *Software Description Ontology*. While many approaches support URIs and semantic linking, only a subset enforce controlled vocabularies or mandatory elements, impacting consistency and machine-processability. Domain-specific schemas like *biotoolsXSD* demonstrate the value of integrating domain ontologies to capture meaningful, reusable metadata. However, none of these approaches fully addresses the unique needs of ERS, particularly in terms of energy-specific elements.

Metadata for research software in the energy domain There exist only a few approaches for metadata of research software in the energy domain. Schwarz and Lehnhoff [192] presented a way to catalog energy-related simulation components by utilizing a semantic media wiki[12] to collect information on simulators. They used the Functional Mock-up Interface (FMI) to describe simulation interfaces. Although the elements of their catalog could serve as a metadata schema, they lack formalization and detailed description, limiting their utility for widespread adoption and integration.

The open energy modeling initiative (openmod) hosts a wiki[3] that lists numerous energy models, each accompanied by metadata such as license information, links to code repositories, model classifications, and detailed model descriptions. The template for the model entries systematically includes multiple metadata elements. Still, the description is not a formalized metadata schema and controlled vocabularies are neither used for the elements nor for the values.

The Open Energy Platform (OEP) extends this approach by providing factsheets on models and frameworks in energy research[4]. The metadata elements in the factsheets are based on the ones of the openmod wiki and partly mandatory. However, the factsheets also lack a formalization. They only partly employ controlled vocabularies for elements or

[10] https://scientificvariablesontology.org/svo/, last accessed 2026-03-03
[11] https://www.wikidata.org/wiki/Wikidata:Main_Page, last accessed 2026-03-03
[12] https://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki, last accessed 2026-03-03

values but do not use domain ontologies for this purpose, which hampers interoperability and discoverability.

While there are some metadata schemas available in other domains and in general as shown before, a formalized metadata schema specifically designed for ERS is still missing. But, the presented efforts by the OEP, the openmod, and Schwarz and Lehnhoff [192] offer valuable starting points for developing such a schema.

4.2 Overarching Method

As none of the existing approaches fulfills all CMRs, this section outlines the overall method used to develop a metadata schema for ERS. According to CMR4, the metadata schema should align with existing standards. Therefore, it should reuse existing metadata elements from various ontologies and schemas and, thereby, forms an application profile (see Section 2.5). While mainly methods for application profiles were used, this thesis also uses the term metadata schema since it is the more common term.

Several methodologies and summaries have been proposed for developing application profiles, including the Singapore framework [166], the Method for the development of Metadata Application Profiles (Me4MAP) by Curado Malta and Baptista [45, 46, 47], the work of Miller [162, pp. 395-404], and the work of Zeng and Qin [225, pp. 222-223]. Each of them outlines a structured process for creating an application profile, emphasizing the importance of domain analysis, stakeholder involvement, and iterative refinement. However, there is no single, universally accepted methodology for developing application profiles, reflecting the diversity of application domains and the evolving nature of metadata standards. In the following, additional details are provided on these methods.

Me4MAP, developed by Curado Malta and Baptista [45, 46, 47], argues for a collaborative approach, defining four key roles (project manager, system analyst, integrator, and final user) that work together in the development of the application profile. The methodology starts with defining functional requirements which serve as a foundation for a domain model. Afterwards, it includes an environmental scan, which analyzes existing schemas, and the development of an element set, syntax guidelines, and usage guidelines.

Miller [162, pp. 395-404] outlines a six-step approach to develop application profiles. His process includes analyzing context, content, and users; selecting an element set; establishing element specifications; creating controlled vocabularies; developing content guidelines; and documenting the profile.

Zeng and Qin [225, pp. 222-223] present a twelve-step process, divided into three phases: analyzing and understanding the collection, developing the element set, and preparing the specification. This approach covers relevant aspects such as examining resources, drafting

desired elements, designing a domain model, and creating crosswalks.

Key aspects from these different methods are combined to a methodology which was used in this thesis to develop the metadata schema for ERS. Generally this methodology follows the process outlined by Curado Malta and Baptista [45, 46, 47] and is shown in Figure 4.1. First, requirements were collected from two complementary sources, which is discussed in detail in Section 4.3: an interview-based requirements analysis with energy researchers and a review of requirements reported in the research software literature. Together these inputs led to a domain model for the metadata schema which defines and clusters the core properties needed to describe ERS. The development part, outlined in Section 4.4, started with a detailed analysis of elements in existing schemas to complete the domain model. After refining the domain model, a first element set was drafted. The element set was then reviewed by a few domain experts, whose feedback informed further adjustments. To ensure high interoperability, the refined elements were aligned with established metadata schemas and appropriate controlled vocabularies were defined for multiple elements. The preliminary schema was evaluated by applying it to two ERS, allowing to assess its practical applicability and to identify any remaining gaps. Finally, the schema was formalized in both JSON-LD and SHACL representations, and a concluding validation test was performed to confirm its consistency and readiness for use. The final schema is named *ERSmeta*.

4.3 Requirements for a Metadata Schema for Energy Research Software

Developing a comprehensive metadata schema for ERS requires a thorough requirement analysis addressing the three facets identified by Miller [162, p. 395]: context, content, and users. Thereby, the content-oriented CMRs are detailed into more specific requirements.

The analysis began with an interview-based study of energy researchers. Following Castro et al. [33], in-depth interviews are an effective way to capture domain-specific needs. Thereby, CMR1 was specified by focusing on a user's perspective to get a better understanding of the required content. Section 4.3.1 introduces the detailed method used for the study. Afterwards, Section 4.3.2 presents the resulting specific requirements, which led to a first domain model, defining the core properties needed to describe ERS. As highlighted in the Singapore Framework [166], the domain model is a crucial artifact in the development of a metadata schema serving as the foundation for the following steps.

In addition to user insights, Section 4.3.3 derived contextual requirements from the broader RSE literature, as introduced in Section 2.1, and from the FAIR4RS principles, detailing CMR2. This ensured that the schema would not only satisfy the immediate infor-

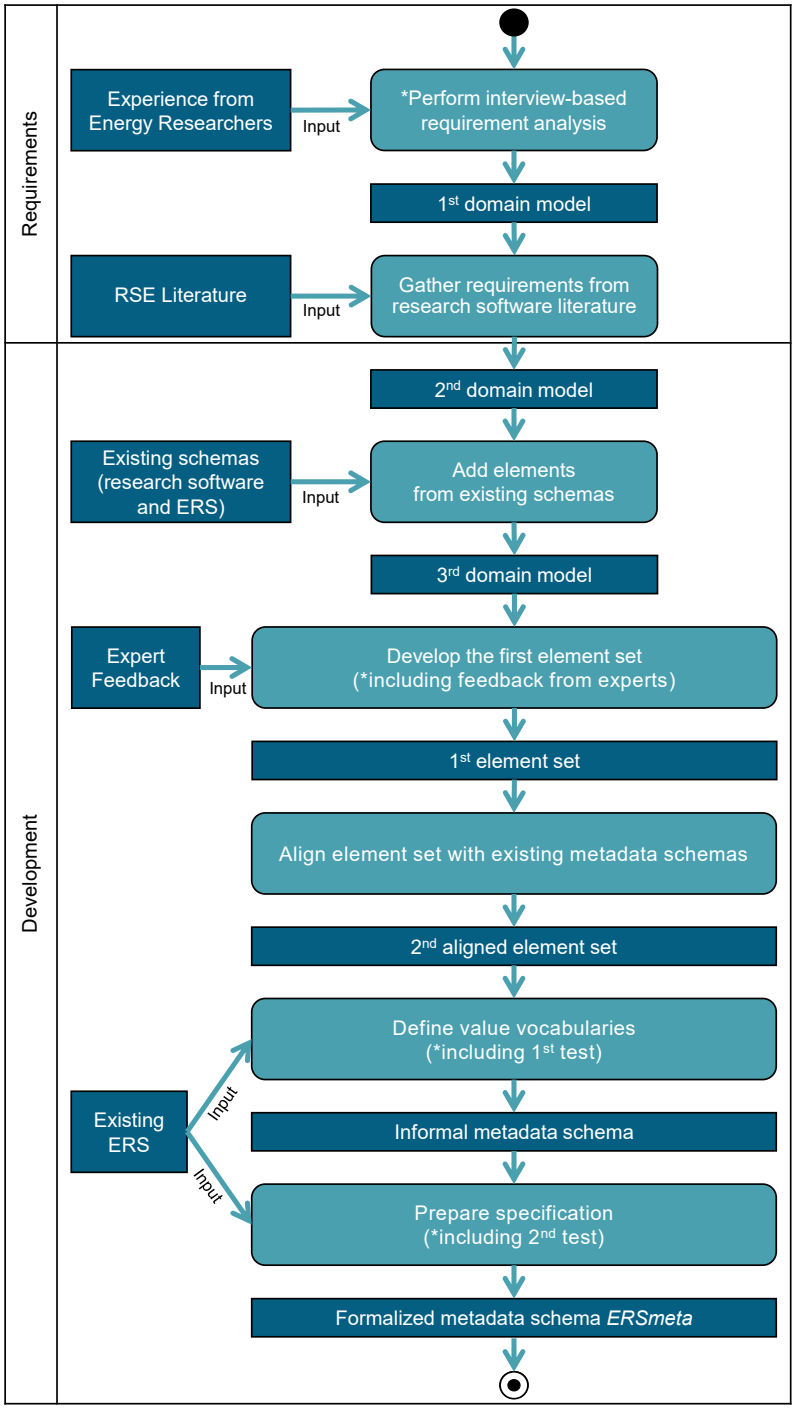


Figure 4.1: Method to develop the metadata schema as UML activity diagram. Light-blue round boxes contain activities where expert interactions are marked with *. Dark-blue rectangular boxes contain (interim) artifacts and inputs.

mation needs of energy researchers but also remains compatible with evolving community standards and organizational practices for FAIR research software. These requirements were added to the domain model, leading to its second version.

4.3.1 Research Design and Method, Data Collection and Analysis

To explore the requirements for ERS metadata, a qualitative approach was employed, involving 32 expert interviews with energy researchers. As research software engineers are yet not established in the energy domain, the interviews focused on researchers who mostly also develop ERS. To ensure a diverse range of perspectives, a wide range of research backgrounds were included, ranging from system-level studies to component-level investigations. Although, some subcommunities may be underrepresented due to the highly interdisciplinary nature of energy research. The selection process aimed to include a balanced representation of both male and female researchers and of different career levels (Ph.D. students, postdoctoral researchers in academia and industry, and professors). By this diversity, the effects of individual biases should be minimized and a comprehensive representation of opinions should be ensured. In terms of career levels, an equal distribution was targeted to avoid response biases, as highlighted by Noble [167]. The interviewees were identified through the researchers' professional networks and invited to participate via email. Figure 4.2 and Table 4.3 provide a detailed overview of the distribution of the interviewees. While the recruitment process aimed to include international researchers, approximately 80 % of the interviewees were based in Germany, which is a limitation of the study as it may have narrowed the geographic diversity of the perspectives captured.

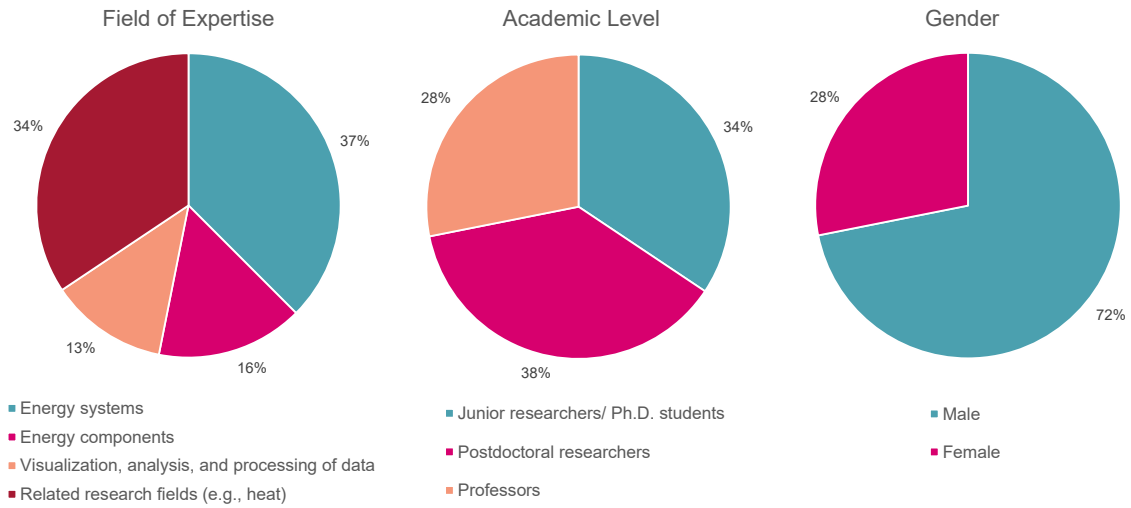


Figure 4.2: Characteristics of interviewees.

Table 4.3: Characteristics of interviewees.

Field of Expertise \ **Academic Level**		Junior researchers/ Ph.D. students	Postdoctoral researchers	Professors
	n	11	12	9
Energy systems	12	I7, I11, I13, I18, I26	I10, I12, I29, I17, I24	I27, I31
Energy components	5	I2, I5	I4, I8, I16	
Visualization, analysis, and processing of data	4	I19, I21, I30		I20
Related research fields (e.g., heat)	11	I1	I3, I22, I23, I32	I6, I9, I14, I15, I25, I28

Additionally, the interviews addressed the needs of energy researchers with respect to the findability and selection of ERS from their perspective. Additional perspectives, e.g., regarding provenance, citation, or long-term preservation, are incorporated through a review of the RSE literature (see Section 4.3.3).

To conduct the interviews along established practices with an appropriate level of reliability [196], an interview guideline was developed as a semi-structured questionnaire based on the FAIR4RS principles and a first analysis of related work as described in Section 4.1. The interview guideline, published as [73], allowed the participants to flexibly express their information needs regarding ERS and was structured into seven parts outlined in Figure 4.3. It began with introductory questions about the interviewees' research and their use of research software (1). The subsequent five sections addressed different aspects of metadata usage for ERS, based on the FAIR4RS principles: findability from a search perspective (2), findability from a selection perspective (3), accessibility (4), interoperability (5), and reusability (6). The guideline ended with concluding questions (7), e.g., the participants' opinions on a research software registry in the energy domain. A pretest interview was conducted with a Ph.D. student to validate the guideline and adjustments were made based on the received feedback.

The interviews took place between August 2022 and January 2023, resulting in a total of 32 expert interviews. They were conducted in either German or English, with some held in-person (n=7) and others online (n=25), and lasting between 20 and 80 minutes. To minimize potential response biases, confidentiality of the interview transcripts was promised, as suggested by Myers and Newman [165]. Theoretical saturation was reached after conducting 32 interviews, as no new themes or concepts emerged, aligning with the

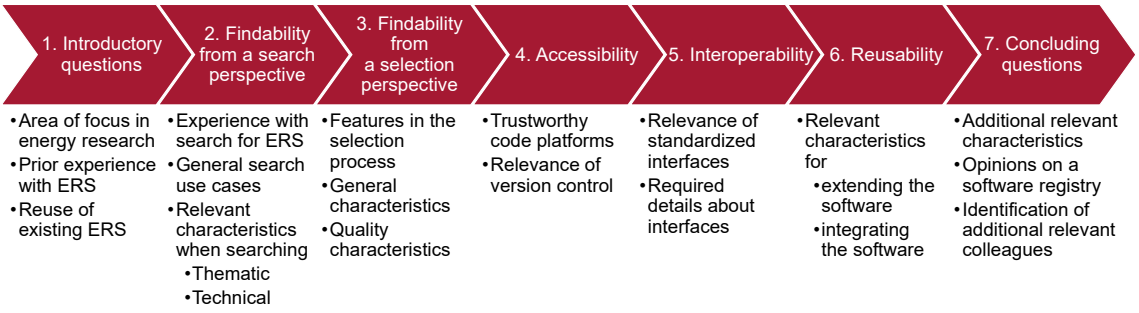


Figure 4.3: Overview of the interview guideline.

principles outlined by Strauss [206].

All interviews were audio-recorded with the consent of the participants. They were transcribed in their original language using whisper[13] [181] and the transcriptions were manually reviewed. The primary data analysis involved qualitative content analysis, using MAXQDA 2022 Analytics Pro[14]. It started with an examination of all available data, identifying common themes and assigning first-order codes based on information areas of ERS, e.g., community. These first-order codes (see Figure 4.4) were then split into particular information needs which were labeled with second-order themes, e.g., contributor. These needs closely resembled potential metadata elements. The derived metadata elements were systematically organized into a first domain model which is outlined in the following section (Section 4.3.2). This domain model serves as a structured framework for understanding the complex information requirements for ERS, derived from the qualitative content analysis. To ensure consistency and completeness, additional elements were introduced to fill existing gaps. For each element, initial definitions were drafted, with reusing existing definitions from *CodeMeta* whenever possible to leverage the established standard.

4.3.2 Detailed Requirements

The domain model provides a structured framework for understanding the information needs of the energy research community and is broadly visualized in Figure 4.4. It is organized around six primary thematic areas (the first-order codes) which serve as the top-level categories: provenance, usage, community, interface, software-specific, and functionalities. These categories are explored in detail in the following part, providing an in-depth examination of each aspect. Thereby, this section presents a general overview of the first domain model, while descriptions for all elements can be found in Appendix C.

[13] https://github.com/openai/whisper, last accessed 2026-03-03
[14] https://www.maxqda.com/de/produkte/maxqda-analytics-pro, last accessed 2026-03-03

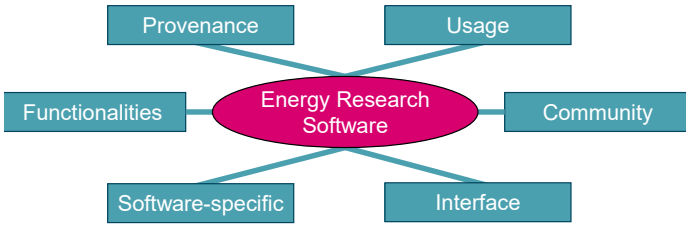


Figure 4.4: First general domain model with the thematic areas. They present the first-order codes of the interview analysis.

Provenance The category provenance focuses on information about the origin of the software. The different identified elements are summarized in Figure 4.5.

The researchers are interested in general background information about the software, such as a publication. Also, they want to know how many people contributed to the software. This number is also seen as an indicator for the quality of the software by the researchers (I27: "So if it's like [...] there's a bunch of people involved in this community, it must be good.").

Regarding the institution behind the software, the interviewees can be divided into two groups. The first group is interested in this information for multiple reasons: if an institution with a high reputation is developing the software, that can be an indicator for a bigger user base (I9: "But maybe if there are researchers at MIT that developed something, they could do a better job in publicizing this and as a result of this, they have a larger user base.") or can be an indicator for quality if the researcher is familiar with the research group behind the software and the typical quality of that group's work. Information about the actor behind a software also allows the researchers to see possible economic interests behind the software. On the other side, a smaller portion of the interviewees is not interested in the information or even expects that they will have an unwanted bias towards software from institutions with better reputations when this information is given.

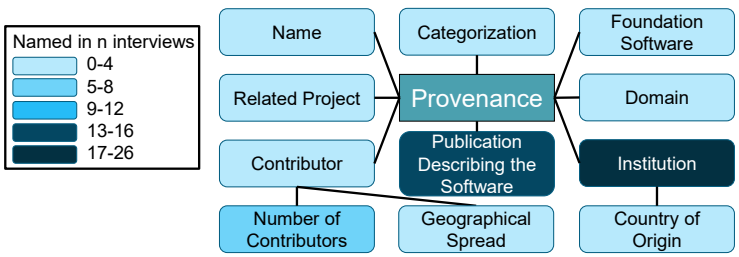


Figure 4.5: Detailed domain model for provenance.

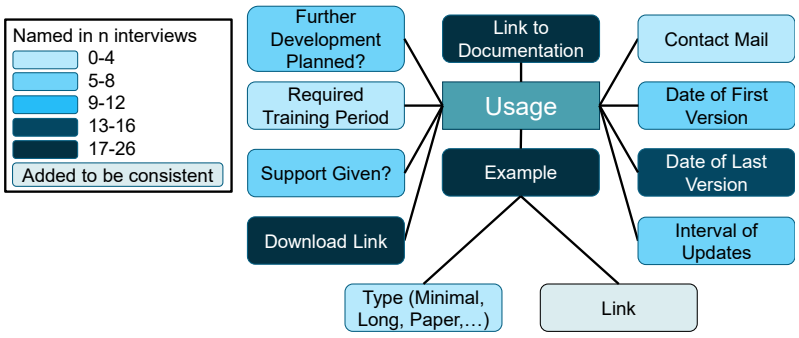


Figure 4.6: Detailed domain model for usage.

Usage This category summarizes information that the researchers want to know to simplify using the software. This includes information on different versions, the given support, and the planned future development of the software. All elements are displayed in Figure 4.6.

Concerning versions, the researchers would like to see information about how often the software is updated (I7: "Which I [...] think is a very important indicator [...] is how long, I say, or how often the code is updated.") and when the last version was published. With this information, they would like to get an impression if problems or bugs they find in the software will be fixed on short notice.

The researchers are especially interested in examples to see if they can directly use the software (I25: "One of the big things that I look for in software is are there examples that are provided with the software"). They want to understand how the software can be used and want to be able to directly run it. Besides examples, they would also like to see a link to the documentation of the software.

Community Different aspects of the engagement with the community and estimating the size of the community are combined in this category. Figure 4.7 gives an overview of all

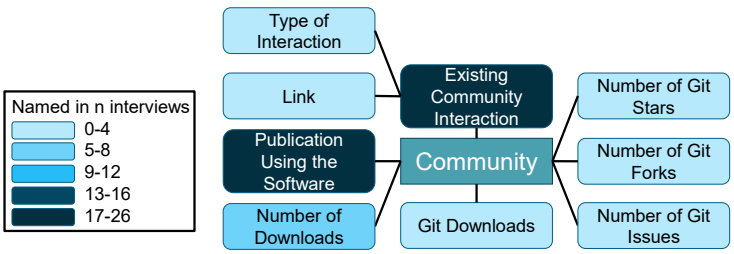


Figure 4.7: Detailed domain model for community.

elements of this category. In general, the researchers would like to know how many users a software has, e.g., by seeing the number of downloads for the software.

The researchers are interested in knowing for which publications the software was already used. On one side, the pure number is interesting for the researchers (I7: "The number of scientific papers that cite the software would actually be super cool.") as it is a good equivalence to paper citations. On the other side, the researchers are looking for examples of what can be done with the software and how it can be done. They expect to get this information from the publications using the software (I9: "So, in many cases, you do not know how to initialize these parameters, and if you see that someone else has used this software in their research, you can also look for, hopefully, that particular section in the paper that explains how this is done, and you can reproduce this and be able to use that software.").

Additionally, the researchers would like to know if there is a community around the software and how it interacts. This interaction can for example include issues on GitHub or GitLab, mailing lists, or community events. Based on this information, the researchers would like to estimate how easy or difficult it is to get support when using the software (I15: "I think a mailing list [...] is very good, because it gives you the opportunity to share your problem with the community of users").

Interface This category summarizes all information needs around the interaction of an ERS with the outside world. This includes information on the available APIs, the types of input the software can read from files (e.g., specified via the command line), or the types of output the software produces (e.g., file types or the command line), as well as links to required or related datasets and software. All related elements are displayed in Figure 4.8. Typically, a software has multiple inputs and outputs. Also, some software can be integrated into other software or a software allows the integration of other software components, e.g., certain models.

The researchers are interested in knowing the interface of the software including its

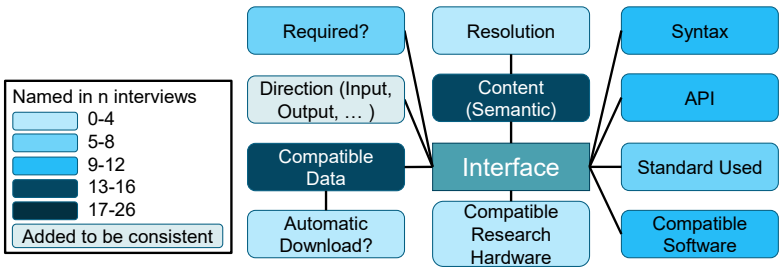


Figure 4.8: Detailed domain model for interface.

syntax, its semantics, a general API description, and the information if the API follows a standard (I21: "But also standardized interfaces to control laboratory devices when it comes to recording measurement data. There are standards, so it would be good if they were at least partially supported."). This information is useful to the researchers to roughly estimate how difficult it is to include the software in their ecosystem of other software and/or laboratory hardware.

Besides the description of the interfaces, the researchers would like to have an overview of compatible data and software (I20: "Not only compatible but what other software can be used together with it?"). Besides better estimating the required integration work, the researchers also hope to find additional relevant software based on this information.

Software-specific In the context of software-specific information, some aspects are already top-level categories by themselves, e.g., usage and interface. The remaining software-specific information can further be separated into quality, licenses, technical requirements, and performance. Figure 4.9 provides an overview of the identified elements.

Concerning the quality of research software, tests are often mentioned but are also controversial. Some researchers state that knowing the number of the tests without knowing the quality of the tests is not useful. (I20: "That depends on how good the tests are. I think the number is not reliable."). On the other hand, interviewees see having tests as a quality indicator for research software (I26: "That's always a good indication that you've at least thought about a few tests. So that's already a plus point.").

For license information, some researchers only require the general information if the

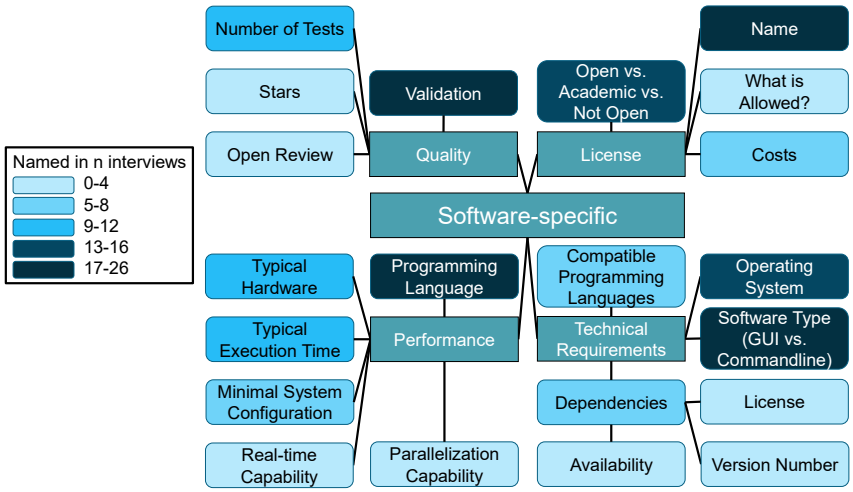


Figure 4.9: Detailed domain model for software-specific.

software is open source while others want to know the exact license. In the context of commercial licenses, the researchers want to know possible license fees (I6: "I don't think I've mentioned license costs yet. That is of course also fundamental. Typically, of course, we don't have huge budgets to procure software licenses in the research projects.").

For the technical requirements, the researchers want to get information about which programming languages the software is compatible with, especially if they have an engineering background (I3: "But if I have to integrate it into my programs myself, then it has to have a Python interface and yes, I think that's my criterion."). Also, they want to know the supported operating systems because being compatible to a specific operating is sometimes a clear requirement for reusing a research software (I27: "I should say, I do have a preference if it doesn't work on a Mac. I don't have a Windows machine. And there's some energy software that doesn't work on Macs like Energy Plus or yeah, CAD software.").

In terms of performance, most researchers see a problem in simply comparing execution times since they highly depend on the used hardware according to the researchers. Nevertheless, an estimated execution time would be helpful for the researchers. Also, they like to know if the software can be run on a laptop or if a HPC cluster is required (I25: "Do I need to deploy this on an HPC cluster [...]?").

Functionalities Within this category, different elements on the functionalities of ERS are clustered. Figure 4.10 shows all elements of this category. The researchers want to get a fast overview of the relevant functionalities of the software to estimate if the software fits their use case and requirements. Therefore, many researchers want to see a graphical abstract of the software, e.g., an UML diagram of the primary functions (I11: "But then a small, for example, abstract or an UML diagram is always nice to see. Or simply a flow diagram, a black-box model, or small boxes that say this goes into a model and this comes out. Simply a small illustration that tries to explain the model to me in fewer words.").

To estimate if the software fits the problem, the researchers would like to see for which use case the software was originally developed and which use cases are currently supported (I15: "Why was this software created? Which problem can now be solved with this tool, which was not reasonably solvable before?").

In the energy domain, location and time are important factors. Therefore, the researchers want to know the geographical and temporal scope as well as the possible geographical and temporal resolutions of the software. Based on this information, the researchers can decide if the software fits their requirements (I18: "This also makes it easier to evaluate, whether this is a similar flight altitude to the one you are looking for."). When looking at certain components, the spatial dimensions which they support are also relevant. Concerning

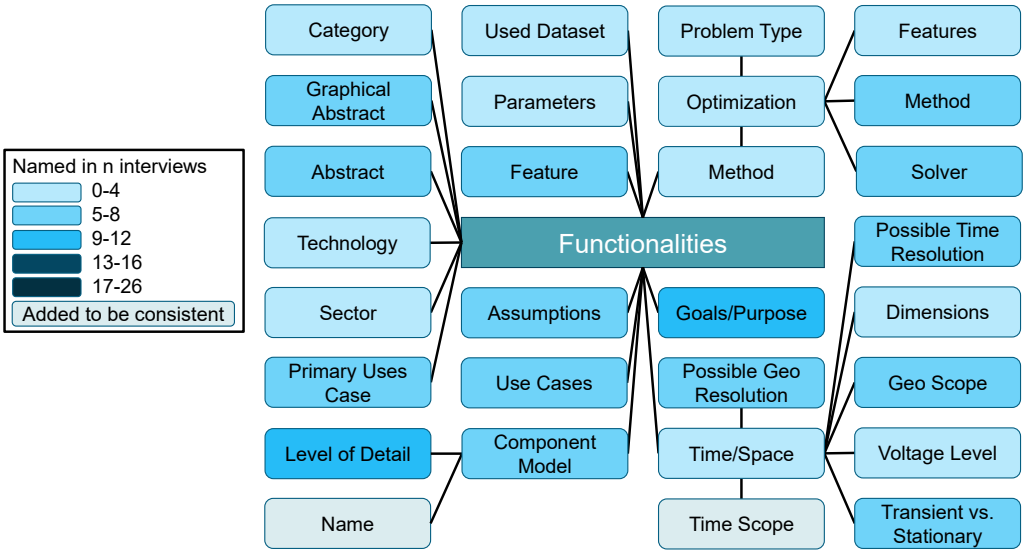


Figure 4.10: Detailed domain model for functionalities.

energy grids, the information if transient effects are covered and which voltage levels are included are also of interest (I17: “In thermal systems, we often do not know before we use libraries, whether transient behavior can be covered or whether capacitive behavior can be included.”).

The researchers also want to know which sectors and technologies are covered and modeled. They often look for models of specific components in energy systems that they need in their research (I1: “Are models available for pumps? Are models available for storage tanks?”). Also, the level of detail of the components is relevant. The models need to include enough details to cover the relevant effects but should not be too detailed because that slows down the overall simulation (I18: “If this is now modeled in more detail, then it probably includes more physical effects, which run on time scales that are not relevant for us and would then make the simulation slower.”).

To see if a software fits, the researchers would also like a quick overview of the assumptions of the model implemented in the software. Since many ERS deal with optimization, the researchers also want detailed information on the optimization, e.g., regarding the solver or the method.

4.3.3 Requirements from Research Software Literature

To ensure that the envisioned metadata schema not only meets the requirements of energy researchers but also contributes to the development of FAIR research software, an analysis of the FAIR4RS principles [39] (see Section 2.2) and additional RSE literature was conducted

(see Section 2.1). This analysis aimed to identify additional specific requirements for the proposed metadata schema.

A relevant work to this regard are the Research Software Metadata (RSMD) guidelines introduced by Gruenpeter et al. [101]. The RSMD guidelines outline a set of relevant requirements for research software metadata based on the FAIR4RS principles. They provide recommendations for reference and identification, including tracking versions, using intrinsic and extrinsic identifiers, and selecting the appropriate granularity level for identification. In addition, the guidelines suggest adding descriptive metadata, such as names and descriptions, and citing related resources, while avoiding duplication of information. They also provide advice on credit and attribution, including adding information about authors and contributors, specifying citation preferences, and identifying owners. The guidelines also cover reuse and legal aspects, such as determining rights holders, adding embargo dates, and licenses, as well as verifying license compatibility and including file headers with license information. Moreover, the guidelines address re-executability, recommending the description of dependencies, operating systems, hardware requirements, and required data as well as the provision of build instructions and user documentation.

These guidelines and additional requirements from RSE literature led to the identification of specific elements that were added to the domain model leading to its second version. For example, Lamprecht et al. [151] defines the need of a metadata license which led to the element `metadata license` in the domain model. Overall, additional aspects from broader initiatives in research software metadata and the FAIR4RS principles were incorporated into the domain model.

4.4 Development of the Metadata Schema

This section presents the development of the metadata schema which is based on the second domain model from the requirement analysis. The development systematically followed the steps, which Section 4.2 introduced and Figure 4.1 summarized. It led to the new metadata schema *ERSmeta*.

4.4.1 Elements in Existing Schemas

The development of a metadata schema requires an environmental scan, as introduced in Me4MAP [47]. This process involves a thorough analysis of existing metadata approaches to identify additional relevant elements. The environmental scan serves as a third information source for the domain model, complementing the requirements gathered from users and RSE literature. To this end, an analysis of elements from related metadata schemas was

Table 4.4: Overview of the analyzed metadata schemas.

Scope	Software-focused	Data-focused
General	*CodeMeta* [136], *DataDesc* [150], R Package Metadata[15], Python Package Metadata[16], *CFF* [60], RSD Data Model[17], *Software Description Ontology* [96]	
Engineering		*Metadata4Ing* [125]
Energy	openmod[3], OEP Factsheets[4]	*Open Energy Metadata (OEMetadata)* [122]
Other domains	*ontosoft* [97], *BiotoolsSchema* [128]	

conducted, which are summarized in Table 4.4. Through this analysis, additional elements were identified and incorporated into the domain model filling gaps and broadening its coverage. Thereby, a more robust and inclusive third domain model was created that is able to cover all relevant information on ERS.

4.4.2 Development of the First Element Set

Miller [162, p. 397] suggested that the analysis of context, content, and users, coupled with the determination of functional requirements, should lead to a preliminary set of desired elements. Before defining such element set, the underlying domain model must be prepared. Since the third domain model captures requirements from different sources, it contained redundancies and areas for improvement. To resolve these issues, a consolidation and refinement of the third domain model was undertaken. This involved removing duplicate entries, ensuring that each element was unique. Additionally, the grouping in the thematic areas was improved, creating a more organized and structured framework.

Building on this improved domain model, a first element set was drafted for the metadata schema. To ensure global accessibility and usability, all elements were exclusively expressed in English to facilitate their adoption by the international energy research community. Definitions were added for all elements. When elements were derived from existing schemas, their original definitions were reused whenever possible to maintain consistency, avoid duplication of effort, and leverage established standards. In line with Miller [162, p. 400], each element was further specified by assigning a data type, a cardinality, and an obligation

[15] https://r-pkgs.org/description.html, last accessed 2026-03-11
[16] https://packaging.python.org/en/latest/specifications/core-metadata/, last accessed 2026-03-03
[17] https://research-software-directory.org/documentation/users/adding-software/, last accessed 2026-03-03

level (mandatory, recommended, or optional) based on the information from the interviews and existing approaches. This prioritization provides users with clear guidance on which elements are essential and which may be omitted in particular contexts.

To ensure the quality and relevance of the element set, five experts were invited to review it. They had knowledge in both research software metadata and ERS and worked at different research organizations in Germany. They were provided with the element set including the descriptions, the obligations, and the cardinalities in an Excel sheet and were asked to complete a brief survey to provide their feedback. The primary focus of this feedback process was to identify any missing elements, unclear descriptions, and inconsistencies within the schema, thereby ensuring its completeness and accuracy. The experts assessed the element set as useful and provided minor suggestions for improvements, indicating a high level of satisfaction with the schema's overall structure and content. They provided detailed comments to roughly half of the presented elements. Their feedback was carefully incorporated into the element set, leading to improved descriptions and changes in the obligations. Thereby, the element set was improved.

4.4.3 Alignment with Existing Metadata Schemas

To foster interoperability and enable seamless integration with existing metadata standards, the element set must align with related metadata schemas. This process, referred to as vocabulary alignment in Me4MAP [47], is vital for ensuring that the developed metadata schema can be easily used with other established schemas. Miller [162, p. 397] suggested to consider the most relevant metadata schemas in the target community to achieve optimal interoperability.

The element set was carefully aligned with a range of established metadata schemas

Table 4.5: Overview of the schemas to which the developed element set was aligned.

Scope	Software-focused	Data-focused
General	*CodeMeta* [136], *CFF* [60], *DataDesc* [150], *Software Description Ontology* [96]	
Engineering		*Metadata4Ing* [125]
Energy	OEP Factsheets[4]	*OEMetadata* [122]
Other domains	*ontosoft* [97], *BiotoolsSchema* [128], *Bioschemas ComputationTool*[18]	

[18] https://bioschemas.org/profiles/ComputationalTool/1.0-RELEASE, last accessed 2026-03-03

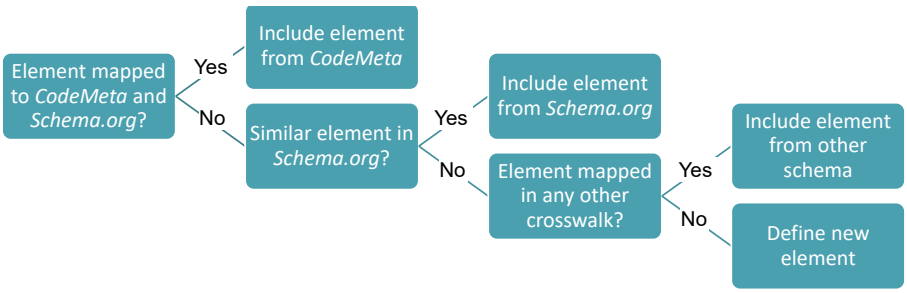


Figure 4.11: Decision tree for element alignment.

summarized in Table 4.5. Therefore, the elements were first mapped to the schemas. Notably, many elements in the element set had multiple corresponding elements across the different schemas and ontologies, highlighting the complexity and richness of the existing metadata landscape. Figure 4.11 shows the method for choosing an element for reuse. For reuse, priority was given to the elements from *Schema.org* because it is the most widely adopted ontology and, therefore, allows to maximize compatibility and consistency. Therefore, elements which are part of *CodeMeta* and *Schema.org* were reused first, e.g., `identifier`. Otherwise, a short check for related elements in *Schema.org* was performed. For example, the element `abstract` is reused from *Schema.org* while it does not exist in *CodeMeta*. If no fitting element was found, the other mappings (Table 4.5) were checked to identify an element for reuse. For each reused element, its original name, description, and URI were adopted, ensuring transparency and traceability. Furthermore, an effort was made to reuse the data types for these elements if they are common data types, like date, Uniform Resource Locator (URL), or text. In some cases, adjustments to the hierarchy of the elements were necessary to better fit to the other elements in the element set.

The alignment process not only ensured that the element set builds upon established standards but also led to the creation of crosswalks to other metadata schemas. These crosswalks allow the mapping and transformation of metadata between different schemas, further enhancing the usability of the element set for different tasks.

4.4.4 Definition of Value Vocabularies

Controlled value vocabularies are essential to ensure the consistency, accuracy, and interoperability of the metadata as introduced in Section 2.5. Establishing them is a relevant step in the development of a metadata schema, as emphasized by Miller [162, p. 402] and Zeng and Qin [225, p. 222]. Consequently, the development of this metadata schema included the identification of value vocabularies for multiple elements to limit their admissible values and, therefore, promoting consistency across the metadata.

The process began by identifying the elements that require controlled vocabularies, followed by the selection of suitable vocabularies from existing ontologies and knowledge graphs. Regarding energy-specific elements, the selection of value vocabularies was informed by the comparative analysis of energy-specific ontologies by Steinert et al. [201], which guided to the use of the *Open Energy Ontology (OEO)* [25] as the primary domain ontology. Regarding domain-agnostic elements, especially knowledge graphs like Wikidata were leveraged, utilizing their established vocabularies to ensure broader interoperability. In cases where suitable existing ontologies or knowledge graphs were not available, simple value vocabularies were constructed from existing taxonomies in literature, expert knowledge, and other relevant sources. For example, literature on specific topics was consulted to inform the development of some value vocabularies, e.g., the work on real-time capabilities by Shin and Ramanathan [195] was used to define the value vocabulary for `realtimeCapable`. This approach ensured that the value vocabularies were comprehensive and relevant to the energy research domain.

Together with the element set, 24 value vocabularies were then incorporated into a metadata schema, which was subsequently tested with two ERS to validate its usability and effectiveness. The testing involved collaborative metadata creation with the developers of the software, providing valuable insights into the schema's practical applicability and identifying areas for improvement. Based on the observation and feedback from this testing with both ERS, the schema was refined.

4.4.5 Preparation of the Specification

The final phase of developing the metadata schema entails preparing a comprehensive specification, as outlined by Zeng and Qin [225, p. 222]. This involves producing a complete, machine-actionable specification compatible with semantic web applications. The specification should include the relevant aspects of the schema, like the element set, the element definitions, and the controlled value vocabularies. Also, the schema needs to be documented, e.g., with detailed guidelines on how to use the metadata schema, content guidelines, and crosswalks.

Therefore, the element set and the value vocabularies were combined into a single metadata schema named *ERSmeta*. To satisfy both semantic web requirements and practical usability, *ERSmeta* was formalized in two syntaxes: SHACL expressed in Turtle and JSON-LD Schema. Both formats are introduced in Section 2.5. The SHACL representation was chosen to ensure high semantic web compatibility, enabling complex constraints for all elements, such as restricting an element to a specific ontology class, which is particularly beneficial for dynamic value vocabularies and the linkage to external knowledge graphs.

Additionally, a JSON-LD Schema was employed to provide easier human readability and to facilitate better interoperability with *CodeMeta*, which is also available in that format.

The formalization process started with the definition of the SHACL using the AIMS platform[19], which allowed the creation through a graphical user interface. Afterwards, the SHACL definition was refined in detail using a text editor. It was then translated into a JSON-LD Schema, which was manually improved to include all required information. To facilitate interoperability with other metadata standards and promote the reuse of *ERSmeta*, crosswalks were defined for *CFF*, *CodeMeta*, *DataDesc*, *Software Description Ontology*, and the OEP Framework Factsheets. These crosswalks enable the mapping and transformation of metadata between different schemas, enhancing the usability and adaptability of *ERSmeta* across various research contexts.

Following the formalization, a second test was conducted to verify the formalization of *ERSmeta* regarding correctness and usability. Informal metadata for one of the software from the first test round were used to create formal metadata in both formats (Turtle following the SHACL and JSON-LD following the JSON-LD Schema). Also, formal metadata were created for an additional software together with a developer that is responsible for that software. The test assessed how well all information could be expressed in both formats and evaluated the automatic validation against the schema. This rigorous testing allowed the identification and correction of errors, as well as improvements to the descriptions, refining *ERSmeta*.

ERSmeta was published in both SHACL and JSON-LD Schema together with its descriptions, crosswalks, and an example (metadata for a framework for agent-based simulation called mango [191]) in a GitHub repository[20] and archived on Zenodo [71] allowing for straightforward use and adoption by the research community. The resulting *ERSmeta* schema is presented in Section 4.5, providing an overview of its structure, elements, and formalization.

4.5 Metadata Schema: *ERSmeta*

The development led to *ERSmeta* which is illustrated in full in Figure 4.12. *ERSmeta* comprises 86 top-level elements clustered into ten thematic areas. For instance, the functionality area focuses on domain-specific aspects of a software's functionality, while other areas cover different facets of the software, such as its technical requirements, provenance, and usage. 23 of the top-level elements utilize data types that require separate schemas, such as `author`, which uses the data type `Person` to allow a more detailed description.

[19] https://coscine.rwth-aachen.de/coscine/apps/aimsfrontend/, last accessed 2026-03-03
[20] https://github.com/NFDI4Energy/ERSmeta, last accessed 2026-03-04

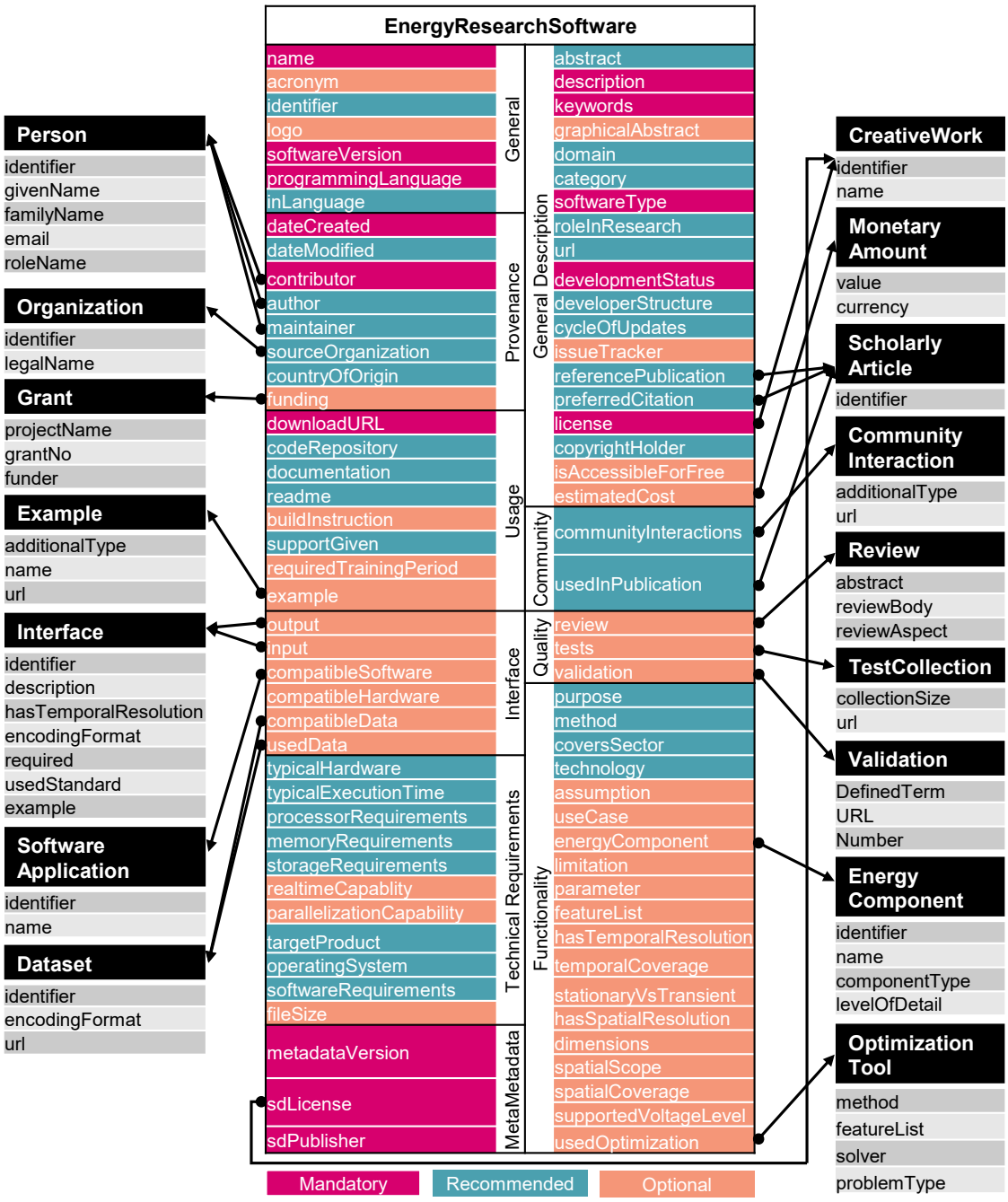


Figure 4.12: *ERSmeta*: a metadata schema for ERS including additional schemas for certain data types (on the outside in gray). The obligations of the elements are color-coded.

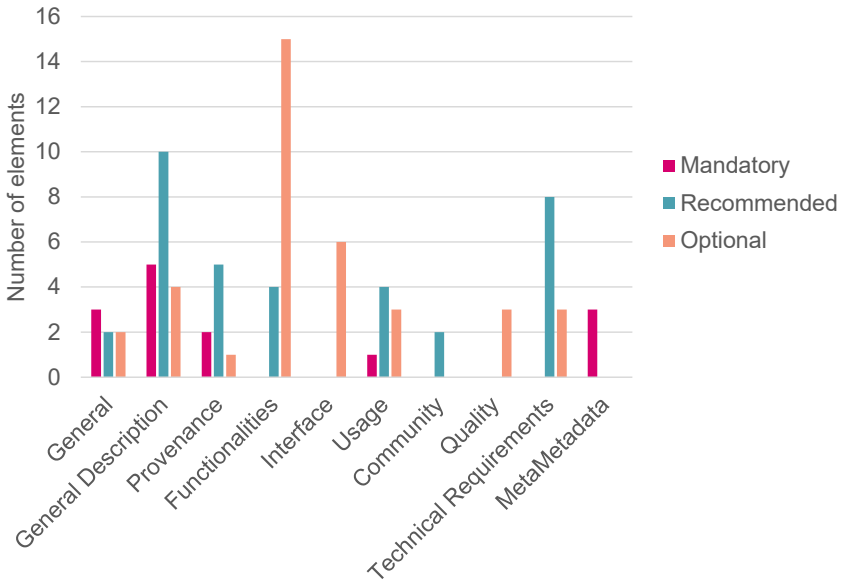
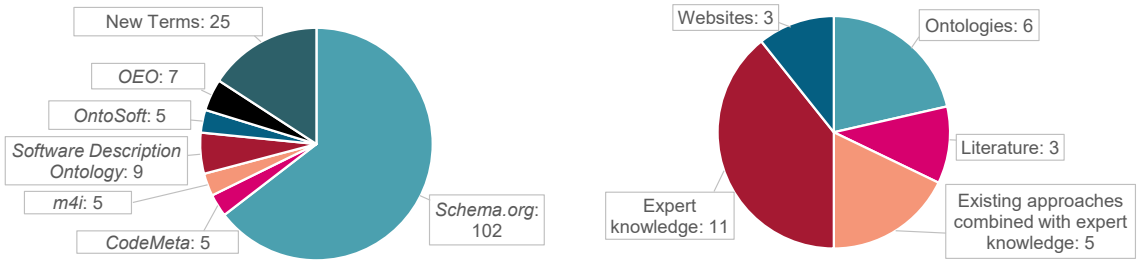


Figure 4.13: Distribution of the obligations in *ERSmeta* over the different thematic areas: mandatory, recommended, and optional.

To address this, additional 16 metadata schemas with 48 elements were defined, as shown in Figure 4.12. These supplementary schemas provide further granularity, ensuring that the captured metadata is accurate and comprehensive. When considering each top-level element filled at least once, the total number of elements presented in *ERSmeta* is 158.

To provide clarity on the importance and priority of the elements within the *ERSmeta* schema, they were sorted into one of three obligation levels: mandatory, recommended, or optional. Figure 4.13 provides an overview of how the mandatory, recommended, and optional elements are distributed across the ten thematic areas of *ERSmeta*. 14 mandatory elements ensure a basic level of metadata quality. These elements provide fundamental information about the software, such as its `name` and `programmingLanguage`. 35 recommended elements offer a more detailed overview of the software and users of the schema are encouraged to fill them. Examples of recommended elements include `referencePublication` and `purpose`. While not mandatory, these elements enhance the metadata's usefulness and facilitate a deeper understanding of the software's capabilities and limitations. 37 optional elements, while not essential, can still provide useful information about the software. Examples include `funding`, `input`, and `output`, which can offer additional insights into the software's development context and operational characteristics.

To achieve high interoperability with other metadata schemas, existing elements were reused for 133 elements in *ERSmeta*. Figure 4.14a provides a detailed overview of the reuse of elements from various existing metadata schemas and ontologies. The majority of reused



(a) Reuse of elements.

(b) Sources for value vocabularies.

Figure 4.14: Statistics of reused elements and used value vocabularies in *ERSmeta*.

elements originated from *Schema.org*. For the purpose of this analysis, elements reused from *CodeMeta* that are also part of *Schema.org* are counted as *Schema.org*. In addition to the elements from *Schema.org*, some elements were reused from *CodeMeta* that are not yet part of *Schema.org*, such as `developmentStatus`. To cover additional general aspects of research software, elements were also reused from the *Software Description Ontology* [96], including `preferredCitation`, and from *OntoSoft* [97], such as `compatibleSoftware`. For more energy-specific aspects, elements were drawn from the *OEO* [25], including `coversSector`, and from *Metadata4Ing* [125], such as `method`. Furthermore, 25 new elements were defined to cover both domain-agnostic and energy-specific aspects of ERS. These include elements like `developerStructure`, which provides information on the organizational structure of the development team, and `energyComponent`, which is specific to the energy domain. The integration of these new elements, along with the reused elements from existing schemas, ensures that *ERSmeta* fulfills the relevant requirements while also being interoperable to existing approaches.

The sources used for the value vocabularies in *ERSmeta* are illustrated in Figure 4.14b, showcasing the diversity of resources leveraged to ensure the comprehensiveness and accuracy of the vocabularies. Existing concepts from ontologies, e.g., from the *OEO* [25], were incorporated, along with knowledge graphs like Wikidata[11]. Literature-based categorizations were also utilized, for example, the categorization for `roleInResearch` from Hasselbring et al. [111]. Also, approaches from websites were used, e.g., repository status[21] for `developmentStatus`. In some cases, existing approaches were extended with expert knowledge, such as the value vocabulary for `softwareType`, which expanded a vocabulary from *BiotoolsSchema* [128]. Additionally, predefined lists of elements were created using expert knowledge, such as the list for `supportedVoltageLevel`.

To further evaluate *ERSmeta*, a quantitative study was performed which is presented in Chapter 6.

[21] https://www.repostatus.org/, last accessed 2026-03-03

5 Metadata Extraction and Curation

While a well-designed metadata schema, like *ERSmeta*, is a prerequisite for FAIR research software, the actual creation of metadata remains a complex task. A wide variety of information, e.g., technical details, provenance, licenses, and usage instructions, must be collected and entered manually to produce meaningful metadata. In many cases, much of this information already exists in different locations such as in GitHub repositories or in README files. However, it is often fragmented, incomplete, or not known in its entirety by a single person, especially for long-standing software projects. Consequently, generating high-quality metadata for ERS is currently often a labor-intensive activity that calls for dedicated support tools.

Automated or semi-automated tools can address this problem by harvesting existing information and guiding users through the remaining gaps. This leads to RQ3:

RQ3: How can a tool support the creation of metadata for ERS?

To address this RQ, Section 5.1 first derives a set of functional and non-functional requirements for such an automated tool. Next, Section 5.2 surveys existing approaches and evaluates them against these requirements. As none of the existing tools fulfills all requirements, Section 5.3 introduces the newly developed tool, the Software Metadata Extraction and Curation Software (SMECS), and describes its architecture and main features. Section 5.4 reports a small initial evaluation of SMECS and discusses its limitations and results.

Parts of this chapter are already published in [82] (see Appendix A for detailed descriptions of the author's contributions to this publication). The initial evaluation presented at the end of this chapter and included in [82] was conducted as part of the master thesis of Aida Jafarbigloo which was supervised by the author of this thesis.

5.1 Requirements

Within this section, an overview of the functional and non-functional requirements for a tool supporting metadata creation is presented. Generally, the tool must minimize the manual effort required to create metadata for research software by automatically harvesting information that already exists in a variety of resources, while still allowing to review, correct, and enrich the extracted data. Therefore, its intended users are research software engineers and researchers who develop software (collectively referred to as *users*).

The Software Requirements (SRs) for the tool are derived from general RSE literature (see Section 2.1), from metadata fundamentals (see Section 2.3 and Section 2.5), and from a

Table 5.1: Overview of the Software Requirements (SRs) classified as requirements for basic and extended features.

	ID	Requirement
Basic features	SR1	Extract metadata from structured files (e.g., *CodeMeta*).
	SR2	Extract metadata via APIs (e.g., GitHub[1], GitLab[2], Zenodo[3], Open Researcher and Contributor ID (ORCID)[4]).
	SR3	Allow full curation of extracted metadata (review, edit, add).
	SR4	Provide a user-friendly, sectioned interface suitable for researchers.
	SR5	Enforce and support controlled value vocabularies.
	SR6	Export metadata as a structured file (JSON or RDF-based).
	SR7	Support *CodeMeta* as metadata standard for research software.
	SR8	Allow exchange of the metadata schema (e.g., via JSON Schema).
Extended features	SR9	Extract metadata from unstructured files (e.g., README).
	SR10	Allow dynamic sources for value vocabularies (e.g., Wikidata[5]).
	SR11	Enable upload of metadata to knowledge graphs, repositories, and/or registries (e.g., Zenodo[3], Open Research Knowledge Graph (ORKG)[6]).
	SR12	Reuse existing open source frameworks for handling research software metadata.

[1] https://github.com/, last accessed 2026-03-03
[2] https://about.gitlab.com/, last accessed 2026-03-03
[3] https://zenodo.org/, last accessed 2026-03-03
[4] https://info.orcid.org/documentation/, last accessed 2026-03-15
[5] https://www.wikidata.org/wiki/Wikidata:Main_Page, last accessed 2026-03-03
[6] https://orkg.org/, last accessed 2026-03-03

first analysis of the landscape of existing tools as described in Section 5.2. The requirements are differentiated between those for basic features for the tool and extended features. Basic features are necessary for a first working prototype while extended features can further improve the usability of the tool. Table 5.1 summarizes all requirements with identifiers and differentiates them between basic and extended features. All SRs are presented in more detail in the following.

Metadata extraction When a research software project is hosted in a repository on GitHub or GitLab, the tool must start by extracting as much metadata as possible from that repository. By invoking the respective APIs (SR1), it can retrieve some metadata, e.g., `name`, `programmingLanguage`, and `dateCreated`. In addition to API calls, the tool must be able to read structured files that are commonly present in software repositories (SR2), e.g., *CodeMeta* files, *CFF* files, or language-specific files such as `pyproject.toml` for Python and DESCRIPTION file for R. Table 5.2 lists possible sources and indicates whether they are basic or extended features.

Beyond structured sources, the tool should also be capable of parsing unstructured files (SR9), such as README, source code, and documentation files, to harvest additional information (e.g., usage instructions or high-level descriptions). Because extracting information from free-text material is a complex task and depends on how consistently the source is written, this is an extended feature.

Table 5.2: List of possible extraction sources classified as basic and extended features. The sources are grouped into three categories: (1) structured files that follow a defined format and, therefore, are machine-readable, (2) web APIs that expose specific information, and (3) unstructured files requiring advanced techniques to extract the relevant data.

	Structured Files	**APIs**	**Unstructured Files**
Basic features	• *CodeMeta* • *Citation File Format (CFF)*	• GitHub[1] • GitLab[2]	
Extended features	• git history • `pyproject.toml`, `setup.py` (Python) • `DESCRIPTION` (R) • `Cargo.toml` (Rust) • `package.json` (Node)	• Zenodo[3] • Digital Object Identifier (DOI) • ORCID[4] • OpenAlex[7]	• README • Source code • Documentation • License text

[7] `https://openalex.org/`, last accessed 2026-03-03

Curation and user interaction All extracted data must be presented to the users for review and editing (SR3). The interface must be user-friendly, organizing the metadata into logical sections rather than displaying all elements on one single page (SR4). It should be intuitive for the users who may not have deep expertise in ontologies or semantic web technologies. Users must be able to add metadata that cannot be obtained automatically, e.g., `developmentStatus`. For every element that is linked to a controlled vocabulary, the tool must enforce the use of that vocabulary (SR5). Ideally, the tool should support dynamic vocabularies (for instance, queried from Wikidata) to remain up-to-date without manual effort (SR10).

Output and interoperability The final metadata record must be exportable as a structured file, either in JSON format or an RDF-based representation (e.g., JSON-LD or Turtle) (SR6). Ideally, the tool should be able to directly upload the created metadata to relevant knowledge graphs (like the ORKG), repositories (such as Zenodo), or dedicated research software registries (like the research software directory[8]) (SR11). Full compatibility with *CodeMeta* is required (SR7), as it is the de-facto general standard for research software metadata (see Section 4.1). Moreover, the tool must allow to change the underlying metadata schema without much effort, thereby facilitating reuse in different research domains (SR8). Finally, whenever possible the tool should build on existing open source frameworks for handling research software metadata because the reuse of established libraries reduces development effort and improves the reliability of the tool (SR12).

5.2 Related Work

After outlining the requirements for the metadata tool, this section presents existing approaches and analyzes to which extent they fulfill the SRs. Table 5.3 outlines a summary of this analysis.

The CodeMeta Generator[9], developed by the Software Heritage initiative[10], is a web-based form that enables users to manually create a *CodeMeta* record. The interface presents the majority of *CodeMeta* elements on a single page and provides drop-down menus for `license` and `developmentStatus`. Users can import an existing *CodeMeta* file, validate it, and export a metadata record as a JSON-LD file. The tool does not retrieve any information from external sources, nor does it offer a mechanism for uploading the generated metadata to external services.

[8] https://research-software-directory.org/, last accessed 2026-03-03
[9] https://codemeta.github.io/codemeta-generator/, last accessed 2026-03-03
[10] https://www.softwareheritage.org/, last accessed 2026-03-03

Table 5.3: Overview of existing approaches to create research software metadata analyzed based on the requirements from Section 5.1.

✓: fulfilled (✓): partly fulfilled ✗: not fulfilled

Tool	Extracts structured metadata (SR1, SR2)	Allows curation (SR3)	Provides user-friendly interface (SR4)	Supports controlled vocabularies (SR5)	Exports structured file (SR6)	Supports CodeMeta (SR7)	Extracts unstructured metadata (SR9)	Enables upload (SR11)	Is reusable/integratable for new approach
CodeMeta Generator[9]	✗	✓	✗	✓	✓	✓	✗	✗	✗
Betty's research engine[11] [193, 194]	✓	(✓)	✗	✗	✓	✗	✗	✓	✗
HERMES [57, 149, 161]	✓	(✓)	✗	✗	✓	✓	✗	✓	✓
SoMEF [147, 158]	✓	✗	✗	✗	✓	✓	✓	✗	(✓)
Auto-CodeMeta Generator[12] [93]	✓	✓	✗	✓	✓	✓	✓	✗	✗

Betty's research engine [193, 194] is a specialized search engine for research software, available as a web service[11]. The system harvests structured metadata from a broad range of available services, including GitHub, GitLab, Zenodo, OpenAlex, DataCite[13], and OpenCitations[14]. It combines these sources using a cascading-search strategy. Although the engine aggregates different metadata, it does not support *CodeMeta*. Users can select records of interest and export them as JSON, CSV, or bib file. The tool allows users to upload the metadata to the ORKG and to briefly edit them beforehand. Thus, the engine satisfies many extraction requirements and provides a minimal form of curation, but it has a different scope, lacks support for controlled vocabularies, and does not support *CodeMeta*.

The Helmholtz Rich Metadata Software Publication (HERMES) [57, 149, 161] framework automates the publication of software metadata by using CI/CD pipelines. Its workflow is

[11] https://nfdi4ing.rz-housing.tu-clausthal.de/, last accessed 2026-03-03
[12] https://autocodemeta.linkeddata.es/, last accessed 2026-03-03
[13] https://datacite.org/, last accessed 2026-03-03
[14] https://opencitations.net/, last accessed 2026-03-03

organized into five sequential phases: harvest, process, curate, deposit, and post-process. It uses a *CodeMeta*-based data model to pass metadata from one phase to the next. During the harvest phase, HERMES extracts structured metadata from a limited set of sources: *CFF* files, *CodeMeta* files, the local git history, and the `pyproject.toml` file. Curation is carried out indirectly via pull- or merge-requests on GitHub or GitLab, without a dedicated graphical interface. After processing, the final record can be deposited in InvenioRDM[15]-based platforms such as Zenodo. HERMES meets several of the extraction and upload requirements. However, its extraction functionalities are confined to a narrow set of structured sources and its curation capability is limited. HERMES allows to add additional functionalities as plugins for each phase. Thus, HERMES forms a good base to build on.

The Software Metadata Extraction Framework (SoMEF) [94, 147, 158] focuses on the extraction of metadata. It harvests structured information from the GitHub API and from common configuration files such as `setup.py`, `pyproject.toml`, and `package.json`. Additionally, it parses unstructured files, including README, license, and Docker files, to capture usage instructions and other information. SoMEF does not provide any curation interface. The extracted record can be exported to a *CodeMeta* JSON-LD file or to an RDF representation based on the *Software Description Ontology*. Designed for strong semantic web compatibility, SoMEF can be employed to generate knowledge graphs, as demonstrated by Kelley and Garijo [147]. A web demonstrator (SOMEF-Vider) is publicly available[16]. Since SoMEF focuses on extraction, a harvest plugin for HERMES can be developed that uses SoMEF.

The Auto-CodeMeta Generator [93] extends the original CodeMeta Generator by incorporating SoMEF and is available as on online demonstrator[12]. It automatically harvests both structured and unstructured information from a repository, including information from README files. The resulting record is displayed in the single-page form used by the CodeMeta Generator. By combining these tools, the Auto-CodeMeta Generator brings together good extraction capabilities with curation. Still, it does not provide a well-structured curation interface and lacks functionality for directly uploading the completed record to external services.

Overall, Betty's research engine offers a broad source coverage but does not adopt *CodeMeta*. The CodeMeta Generator and SoMEF each only address a single aspect, curation or extraction. While the Auto-CodeMeta Generator combines both tools, it still lacks a

[15] InvenioRDM is a framework for research data management repositories initially developed by European Organization for Nuclear Research (CERN) and used by multiple research institutions, like the Helmholtz-Zentrum Dresden-Rosendorf (HZDR) or the Universität Münster; https://inveniosoftware.org/products/rdm/, last accessed 2026-03-03

[16] https://somef.linkeddata.es/, last accessed 2026-03-03

user-friendly curation interface. HERMES provides an end-to-end pipeline, but its curation is limited to pull-requests and its extraction scope is narrower than the one of SoMEF. These observations reveal a clear gap: no existing solution combines comprehensive extraction with an intuitive curation interface. The proposed tool, SMECS, is designed to fill this niche by integrating the strengths of the existing approaches while addressing their shortcomings.

5.3 Description of the Developed Tool: SMECS

To meet the requirements for the basic features outlined in Section 5.1, SMECS was developed as a new web application. The software builds on the architecture of the Meta Tool [124] created by the Reiner Lemoine Institut[17], which already supports the creation of metadata for energy research data following the *Open Energy Metadata (OEMetadata)* schema [122]. For the automatic extraction of software metadata, SMECS uses HERMES extending it with additional plugins that enable new harvesting sources. SMECS is written in Python using the Django framework (v 4.2.22)[18]. The frontend relies on standard JavaScript, HTML, and CSS. The source code is hosted on GitHub[19] under an AGPL license and stable releases are archived on Zenodo [80]. A dedicated branch incorporates *ERSmeta*[20] while the main branch uses *CodeMeta* 3.0 addressing requirement SR7. SMECS follows a workflow consisting of four phases as illustrated in Figure 5.1: start, extraction, curation, and export.

Start On the landing page of SMECS (see Figure 5.2), a user can enter the URL of a software repository hosted on GitHub or GitLab and, optionally, a personal access token for the corresponding platform. The token is used by SMECS to authenticate calls to the API of GitHub or GitLab, which can increase the amount of retrievable metadata. Tokens are mandatory for private repositories and self-hosted GitLab instances. For public repositories on GitHub or the main GitLab instance, SMECS can use internal default tokens if the user do not provide one.

Extraction Based on the provided information, SMECS invokes the harvesting component of HERMES in the extraction phase. Minor modifications to HERMES (v 0.8.1) were made to handle remote repositories, which will be included in the official HERMES release. For additional functionalities, plugins for HERMES were written to harvest metadata from

[17] https://reiner-lemoine-institut.de/en/, last accessed 2026-03-03
[18] https://www.djangoproject.com/, last accessed 2026-03-03
[19] https://github.com/NFDI4Energy/SMECS, last accessed 2026-03-04
[20] https://github.com/NFDI4Energy/SMECS/tree/v0.9.1_ERSmeta, last accessed 2026-03-03

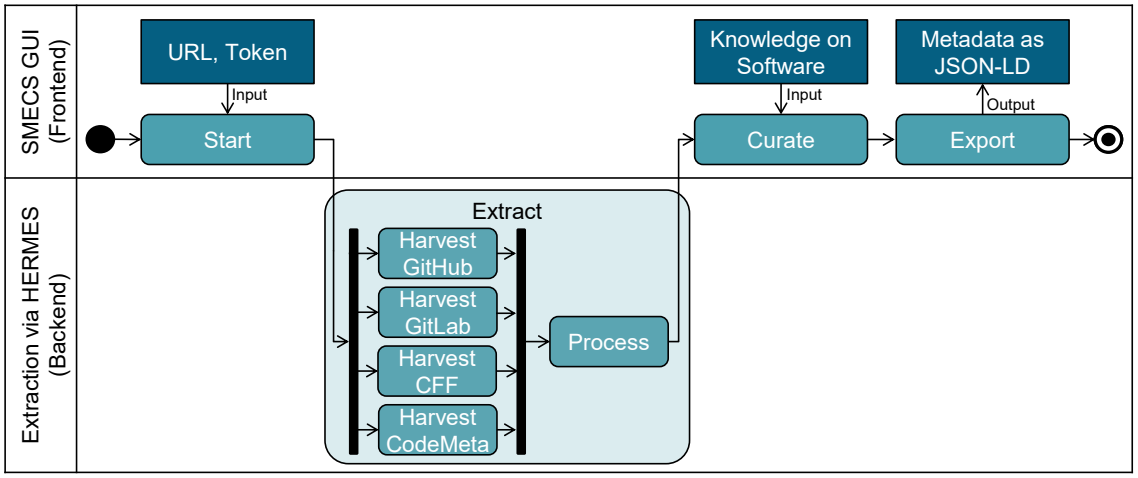


Figure 5.1: UML activity diagram of SMECS with the four phases: start, extraction, curation, and export. Round boxes contain activities. Rectangular boxes contain inputs and output.

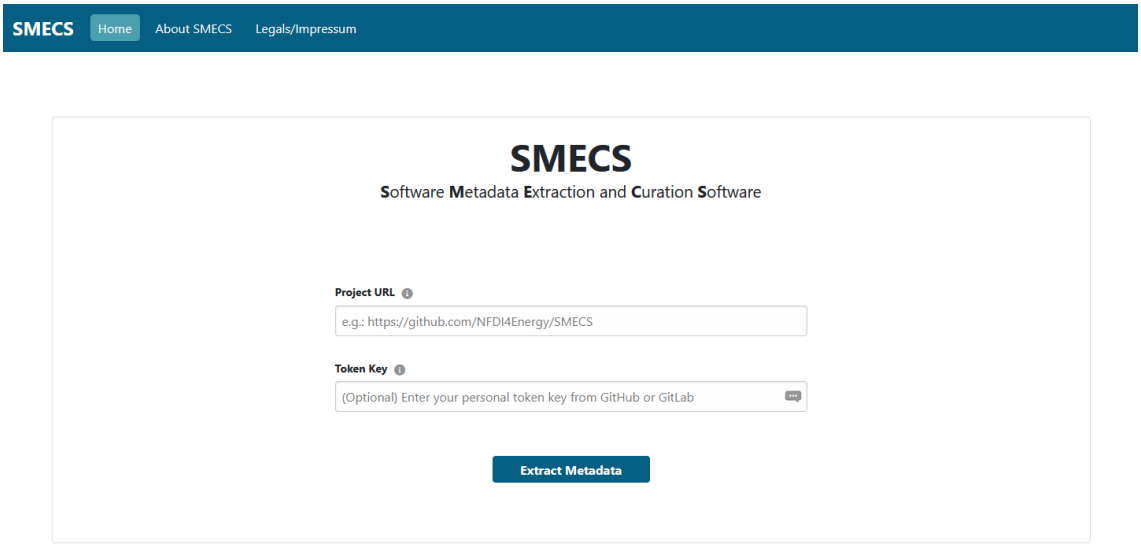


Figure 5.2: SMECS landing page for its start phase: a user can enter the URL of a software repository (e.g., on GitHub or GitLab) and, optionally, a personal access token.

GitHub and GitLab. Thereby, long-term interoperability with future versions of HERMES is achieved.

As summarized in Table 5.4, four harvesters are employed addressing the requirements SR1 and SR2: two query the GitHub and GitLab APIs, while the other two read *CFF* and *CodeMeta* files that may be present in the repository. The original harvesters have been extended to operate on remote repositories. Information obtained from the APIs is translated to *CodeMeta*-based metadata using the published crosswalk for GitHub and a custom crosswalk that was developed for GitLab. The crosswalk maps 16 distinct elements, that can be retrieved from the APIs, covering aspects such as the project description, programming languages, collaboration metrics (e.g., `issueTracker`), and activity statistics. The quality of the extracted metadata highly depends on the available information in a repository.

All harvested data are merged by using the process phase of HERMES, yielding a single coherent metadata record. Finally, SMECS converts this record to the currently active schema, either *CodeMeta* or *ERSmeta*, so that it can be shown to the user for curation.

Curation The curation page shows the harvested metadata according to the active schema, which is loaded as JSON Schema. From the schema file, SMECS gets the obligation level, the data type, the cardinality, and any associated controlled vocabulary for each element. In this way, the metadata schema can be exchanged, addressing requirement SR8.

Based on this information, the tool renders a form-like interface organized into tabs. Figure 5.3 illustrates the first curation tab for a *CodeMeta* record. When using *CodeMeta* as metadata schema, the tabs are *General Information*, *Provenance*, *Related Persons*, and *Technical Aspects*. For *ERSmeta*, the thematic areas are used as tabs with small adjustments. Thereby, the interface fulfills the requirement SR3 and SR4. It accepts free-text for the data types URL, text, and numbers, while elements that have controlled vocabularies, like `license` or `programmingLanguage`, are accompanied by selection boxes that offer filterable lists of admissible values addressing requirement SR5. Nested elements are shown as tables

Table 5.4: Overview of the supported extraction sources via HERMES and the reference for their mapping to *CodeMeta*.

Source	Description	Reference for Mapping
GitHub	Metadata are extracted from GitHub API	Official *CodeMeta* crosswalk
GitLab	Metadata are extracted from GitLab API	Self-created Crosswalk
CFF	*CFF* files in a repository	Crosswalk by HERMES
CodeMeta	*CodeMeta* files in a repository	No Crosswalk needed

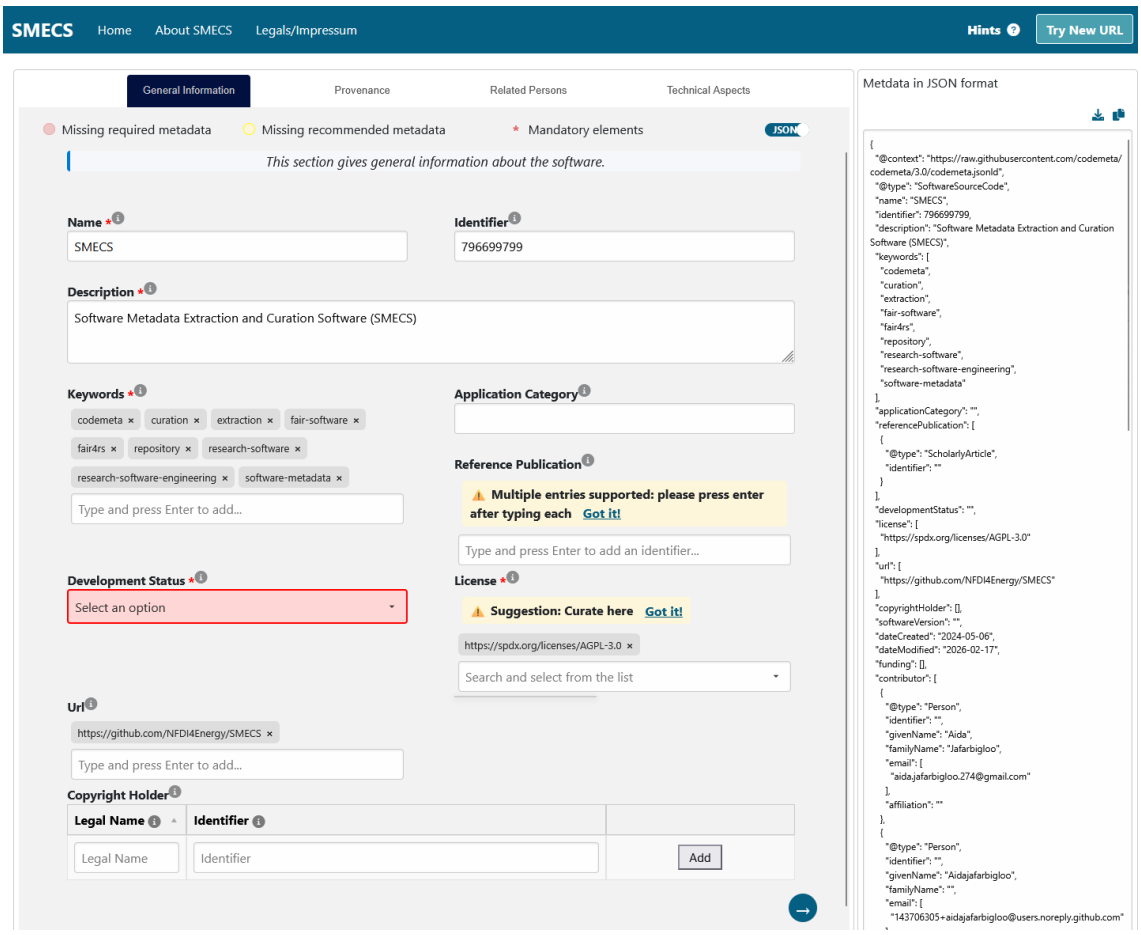


Figure 5.3: SMECS curation page: extracted metadata are displayed in editable fields organized by thematic tabs as shown at the top.

and the cardinality defined in the schema determines whether a single entry or multiple entries are allowed for a certain element. Some of these features are specific to *ERSmeta* and are not required for *CodeMeta*.

Required fields are marked with an asterisk. If the automatic extraction fails or is not available for these elements, the corresponding input box is shown in red in order to draw the user's attention, e.g., `developmentStatus` in Figure 5.3. The input boxes of recommended elements are shown in yellow if no information was extracted. The user can freely edit any field, add missing information, or select values from the provided lists. Next to the form, SMECS directly shows the resulting JSON code.

Export After the user reviewed and completed the record, the metadata can be exported as a JSON-LD file that conforms to the selected schema addressing requirement SR6.

5.4 Initial Evaluation

As an initial evaluation of SMECS, a small study involving six participants was carried out. Section 5.4.1 describes the methodology of this study, including data collection procedures and analysis techniques. Afterwards, Section 5.4.2 presents the findings. Finally, Section 5.4.3 discusses the limitations and outcomes of the study.

5.4.1 Research Design and Method, Data Collection and Analysis

The initial evaluation focused on the usability of SMECS, an important requirement identified in Section 5.1. Consequently, a usability study was conducted concentrating on the user interface. The study used an earlier version of SMECS[21] with the *CodeMeta* configuration of the tool.

Usability testing is an established method to evaluate how easily users can interact with a system and for identifying problems early in the development process [104]. The performed usability testing aimed to determine how effectively users can navigate the interface, comprehend the tool's functionalities, and operate the available features.

Six researchers from OFFIS[22] were invited to participate in the study. They were chosen to cover different perspectives on energy research and different familiarity with metadata. All participants (P1–P6) were PhD students in energy research working on medium- to large-scale research projects. Their academic backgrounds were mainly in computer science (master's degrees), with one participant holding a degree in engineering physics. Self-reported familiarity with metadata ranged from two to four on a five-point scale (five indicating expert knowledge). The study was conducted in-person in English between September and October 2024. Each session lasted between 45 and 60 minutes.

Figure 5.4 shows the overall structure of the usability test. Each session began with a brief introduction that explained the purpose of the study and assured participants confidentiality to minimize potential response biases, as suggested by Myers and Newman [165]. As second step, participants performed a set of predefined tasks designed to test key aspects of the user experience and to exercise the main functions of the tool. The tasks were organized around two scenarios:

Familiar repository In the first scenario, participants used SMECS on a repository they were familiar with. They entered the GitHub or GitLab URL, optionally supplied a personal access token, and initiated the extraction. After the automatic harvest, they reviewed and edited the extracted information, added any missing elements such as a

[21] Commit number: aefc7a9. Subsequent updates added HERMES integration, support for multiple inputs per element, an expanded curation page, redistributed tabs, and improved responsiveness.
[22] `https://www.offis.de/en/index.html`, last accessed 2026-03-03

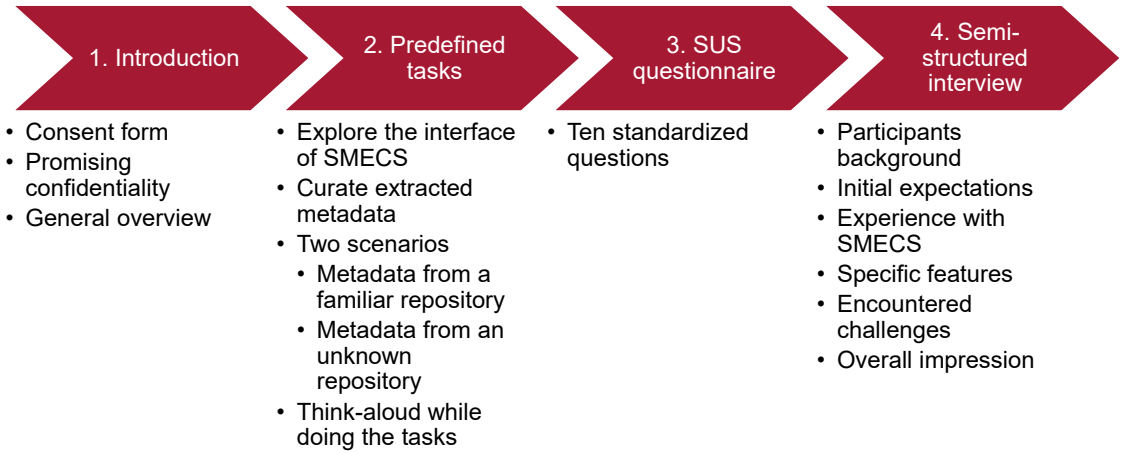


Figure 5.4: Structure of the usability testing: The workflow comprises an introductory briefing, two scenario-based tasks with the think-aloud method, the System Usability Scale (SUS) questionnaire, and a concluding semi-structured interview.

license, and exported the curated metadata as a JSON-LD file while freely exploring the interface and providing comments.

Unfamiliar repository The second scenario required participants to repeat the workflow with an unfamiliar repository supplied by the experimenter. In this case, they compared the extracted metadata against a printed reference sheet, added contributor details, and exported the final record.

The think-aloud method was used during the tasks to capture the participants' feedback, insights, and observations. In this approach, users verbally express their thoughts while performing a task or immediately afterward, providing real-time insight into their decision-making process [61]. All verbalized ideas and thoughts were audio-recorded with the consent of the participants for subsequent analysis.

As third step, participants filled out the System Usability Scale (SUS) questionnaire [28] after completing the tasks. The SUS questionnaire is a common approach to quickly assess the usability of a system [154]. It provides a robust quantitative assessment of perceived usability even with small sample sizes [211]. The SUS questionnaire includes ten statements rated on a five-point Likert scale ranging from *Strongly Disagree* to *Strongly Agree* [28] and is widely employed to evaluate the usability of mobile applications, software, and websites [174].

As fourth and final step, the session concluded with a semi-structured interview that gathered additional qualitative insights. The interview guide addressed participants' backgrounds, initial expectations, first-time experiences, specific features, encountered challenges,

and overall impressions. With the consent of the participants, the interviews were also audio-recorded.

The audio recordings of the think-aloud part and the semi-structured interviews were transcribed using Whisper[23] [181] and subsequently reviewed for accuracy. The transcripts were analyzed using qualitative content analysis. Relevant statements were highlighted and coded into categories, like positive impressions, negative impressions, reported bugs, and suggestions for improvement. Responses to the SUS questionnaire were scored following the standard SUS scoring procedure [28] and reported as an aggregate usability metric.

5.4.2 Evaluation Results

The SUS scores obtained from the usability experiment are shown in Figure 5.5 as a box plot[24]. The distribution indicates a positive experience: the mean score is 92.08 and the median is 92.5, both well above the *Acceptable* threshold of 70 by Bangor et al. [13] and the *Excellent* threshold of 85 by Bangor et al. [12]. Five of the six scores fall in the *Excellent* range, while the remaining participant still reports a high level of usability. This consistent pattern indicates the tool meets user needs and expectations.

The qualitative data gathered through the think-aloud method and the semi-structured interviews provide valuable insights into users' experiences and perceptions. Overall, participants expressed positive impressions, emphasizing the clarity, navigation, and ease of use of SMECS. Several users described the interface as intuitive (P1: "It's very easy to use, very fast. You can change your information quite fast and it already gives you a lot of information.", P2: "The tool is plain, but in a good sense."). Two participants highlighted the smooth, responsive behavior of the application, similar to familiar web-based tools (P4: "The tool feels organized, easy to use and feels like an online tool.").

All participants commented that organizing the metadata into separate tabs made the interface well-structured and simple to curate (P2: "The curation form looks really good. It looks clean in my opinion and less intimidating for users that are not technically inclined.",

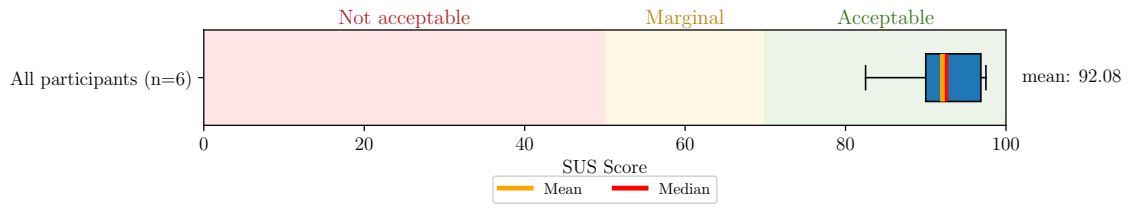


Figure 5.5: SUS scores for the initial evaluation of SMECS with a mean of 92.08.

[23] https://github.com/openai/whisper, last accessed 2026-03-03
[24] The box plot was generated using [72].

P5: “I think it helps not to have too much information on one page, especially when the list of contributors gets very long. If all the metadata is on a single page, you might have to scroll a lot before reaching the next point.”). They acknowledged the color-coded clues matching familiar web conventions (P3: “Colors in color coding perfectly matching to the expectation and they are perfectly matching to the standard, the typical things you would see on the Internet and you are used to.”). The dual visualization during the curation phase, the editable form alongside a live JSON viewer, was well received (P1: “The live view of the JSON is pretty nice, and makes it easy to edit metadata in the form and see the result simultaneously.”).

However, the participants also suggested several enhancements. One request was for advanced automation, such as automatically selecting the appropriate license and generating a software description. Another idea was to allow users to import an existing metadata record, produced by SMECS or obtained elsewhere, so that it could be edited and curated within the tool. Additional suggestions included adding validation rules for specific metadata types and enabling verification of the data displayed in the JSON viewer. Finally, participants recommended providing more detailed role information for authors and contributors (e.g., code development, documentation, writing), similar to the contributor role statements used in scientific publications.

As a result of the feedback during the usability testing, the copy-function in the contributors table was removed during later development. Originally, it was meant to let users duplicate a contributor entry into the authors table, but the participants found the purpose unclear and often misused it. Consequently, contributors and authors have been combined into a single table. The new layout lists each individual once and provides check-boxes for the associated roles, allowing a person to be marked as a contributor, an author, or both.

5.4.3 Limitations and Discussion

Although the initial evaluation provided some interesting insights, some limitations also need to be acknowledged. First, the study was limited to six participants, all of whom were PhD students in computer science oriented energy research. This homogeneity does not reflect the full spectrum of the intended users of SMECS, such as domain scientists from other domains and/or with limited software development experience. Consequently, the findings cannot be assumed to generalize to all potential users. Future evaluations should recruit a larger and more diverse cohort to capture a wider range of expertise and expectations.

Second, the study employed a set of predefined tasks that focused on specific functionalities of SMECS. While this approach ensured that key features were used, it also constrained

participants to a narrow workflow and may have omitted usability issues that arise in more exploratory or real-world usage scenarios. Complementary field studies, e.g., letting researchers just create metadata without further tasks, could reveal additional strengths and weaknesses.

Third, the study was conducted with direct interaction between the participants and the interviewer. Although this setting facilitated direct observation and immediate feedback, it may have introduced a social-desirability bias [108], with participants potentially providing more favorable comments in the presence of the interviewer.

Fourth, the version of the tool used in the experiment predates several recent enhancements, e.g., the integration of HERMES, support for multiple inputs per element, and improved responsiveness. Consequently, the usability observations refer to an earlier version and may not fully represent the experience with the current version.

Despite these limitations, the study led to valuable insights. Participants consistently rated the tool with high SUS scores, described the navigation as intuitive, and appreciated the tabbed layout and the color-coded cues. Specific suggestions, such as removing the copy-function for authors and contributors and merging them into a single table, have already been incorporated into the subsequent development.

In summary, the evaluation demonstrates that SMECS is a promising solution for supporting metadata creation, but further research is required to confirm its effectiveness across a broader user base and with the most recent software version. The integration of *ERSmeta* into SMECS introduces an additional use case that is examined in detail in the next chapter (Chapter 6), where Section 6.4 continues the discussion of SMECS.

6 Evaluation

After introducing multiple artifacts in the previous chapters, this chapter focuses on evaluating two of them more in detail. Chapter 4 presents the metadata schema *ERSmeta* and provides first answers to RQ2.

> **RQ2:** Which metadata are suited to improve the FAIRness of ERS and how can they be structured within a metadata schema?

In order to determine whether the schema is usable for energy researchers, it is necessary to investigate how they perceive the schema during the metadata creation process. For this purpose, the semi-automated metadata generation tool SMECS, introduced in Chapter 5, can be employed. Evaluating *ERSmeta* integrated into SMECS can lead to additional, relevant knowledge about the schema itself, addressing RQ2. Simultaneously, it can provide further insight into SMECS addressing RQ3.

> **RQ3:** How can a tool support the creation of metadata for ERS?

Consequently, this chapter presents a combined evaluation of the two artifacts. First, Section 6.1 outlines the methodology for the evaluation. Afterwards, Section 6.2 presents the empirical findings and discusses the study's limitations.

Finally, the implications for RQ2 and RQ3 are examined by integrating the insights presented in Chapter 4 and Chapter 5 with the results of the evaluation. To keep the discussion focused, it is divided according to the two RQs. Section 6.3 addresses the findings related to *ERSmeta* and RQ2, whereas Section 6.4 focuses on SMECS and RQ3.

Parts of this chapter are included in a paper that is currently under review, available as preprint [91] (see Appendix A for detailed descriptions of the author's contributions to this publication).

6.1 Research Design and Method, Data Collection and Analysis

This evaluation aims to assess how easily researchers and research software engineers in the energy domain can create metadata for their own ERS by using *ERSmeta* together with SMECS. To test both artifacts in a realistic, domain-specific setting, an in-vivo experiment was designed. Participants generated metadata for an ERS that they actively maintain. For this purpose, they used SMECS which embeds *ERSmeta*[1]. They reported their experience by filling an online questionnaire. This approach follows the established practice of evaluating metadata schemas with questionnaires, as illustrated by Palavitsinis et al. [170].

The questionnaire designed for this evaluation is published as [88] and comprised several sections, as shown in Figure 6.1. First, participants provided demographic information, e.g., years of research experience, field of experience, and self-rated familiarity with metadata and ERS. In the second step, they were asked to open an online instance of SMECS and create metadata for their own ERS. Afterwards, the participants returned to the survey and answered question items about their experience (step 3 and 4). Because the participants interacted with both SMECS and *ERSmeta* in a single workflow, a clear differentiation between those artifacts is a difficult task for the participants. Nevertheless, the questionnaire contained separate blocks of questions targeting the tool and the schema individually.

Since SMECS is the primary instrument that participants interacted with, the questionnaire began with items focused on SMECS in the third step. First, the System Usability Scale (SUS) [28] was used to get an impression on the usability also allowing comparability with the results of Section 5.4. Additional items comprised behavioral-intention questions

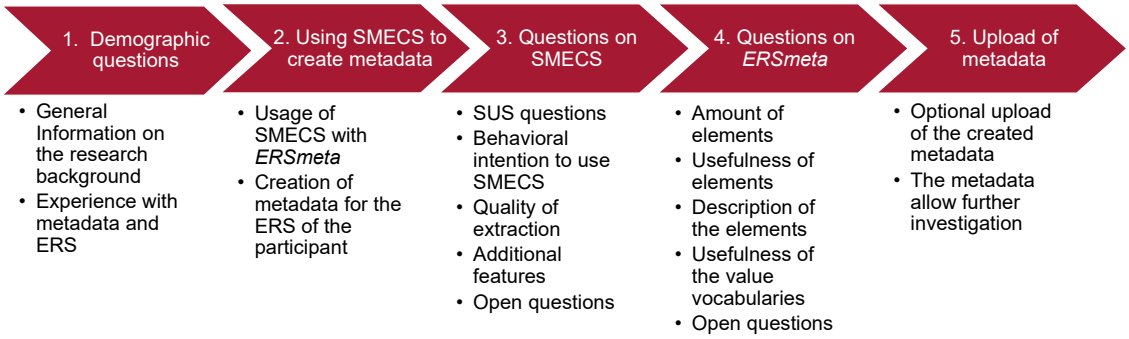


Figure 6.1: Overview of the evaluation process.

[1] https://github.com/NFDI4Energy/SMECS/tree/v0.9.1_ERSmeta, last accessed 2026-01-19

derived from the Unified Theory of Acceptance and Use of Technology (UTAUT) [214] and its third-generation extension (UTAUT3) [68]. Also, the quality of extraction was focused with items based on the Technology Acceptance Model 2 (TAM2) [213]. Additionally, open questions invited suggestions for new features and general comments on the tool.

The fourth step addressed *ERSmeta*. Participants rated the number and obligation (mandatory, recommended, and optional) of the elements with items adapted from TAM [50] and UTAUT [214]. They judged the usefulness of the elements and of their descriptions using items inspired by TAM [50], TAM2 [213], TAM3 [212], and UTAUT [214]. Also, the controlled vocabularies were evaluated with items inspired from TAM [50].

In the fifth and last step, a separate submission form allowed the participants to upload their metadata files which were generated with SMECS. Thereby, the produced metadata can be examined without deanoymizing the questionnaire responses. The whole setup including the questionnaire and the usage of SMECS was pretested with three researchers[2] without revealing any problems.

Regarding the participants recruitment, a broad field was aimed. Therefore, multiple channels were used: the Strommarkttreffen mailing list[3], the open energy modeling initiative (openmod) mailing list[4], the German National Research Data Infrastructure for Interdisciplinary Energy System Research (NFDI4Energy) newsletter[5], the NFDI4Energy LinkedIn page[6], and direct e-mail invitations to known German ERS developers. The survey remained open from July to September 2025. It was accessed 79 times of which twelve participants completed the whole questionnaire. In ten cases, the metadata files generated with SMECS were also submitted. Figure 6.2 visualizes where participants dropped out of the process. Besides early dropouts, it was stated ten times that the software under consideration was not publicly available or hosted in a closed repository. The dropout rates are similar over all domain backgrounds.

The survey responses are publicly available in [87] while the metadata files created by participants remain confidential due to privacy constraints. For the analysis, the SUS scores were derived following the standard scoring procedure [28]. The Likert-scale items were mapped to numbers (*strongly disagree* = 1, ... , *strongly agree* = 5) and the submitted metadata were analyzed regarding used elements and number of entries for these elements. All

[2] All from OFFIS - Institute for Information Technology, Energy Division, https://www.offis.de/en/applications/energy.html, last accessed 2026-03-03

[3] https://www.strommarkttreffen.org/english/, last accessed 2026-03-03, a mailing list of energy professionals in research and industry in Germany

[4] https://groups.google.com/g/openmod-initiative, last accessed 2026-03-03

[5] https://nfdi4energy.uol.de/community-and-updates/newsletter/, last accessed 2026-03-03, NFDI4Energy is the consortium for energy research with the NFDI aiming to establish infrastructure services for research data and research software in the energy domain

[6] https://www.linkedin.com/company/nfdi4energy/, last accessed 2026-03-03

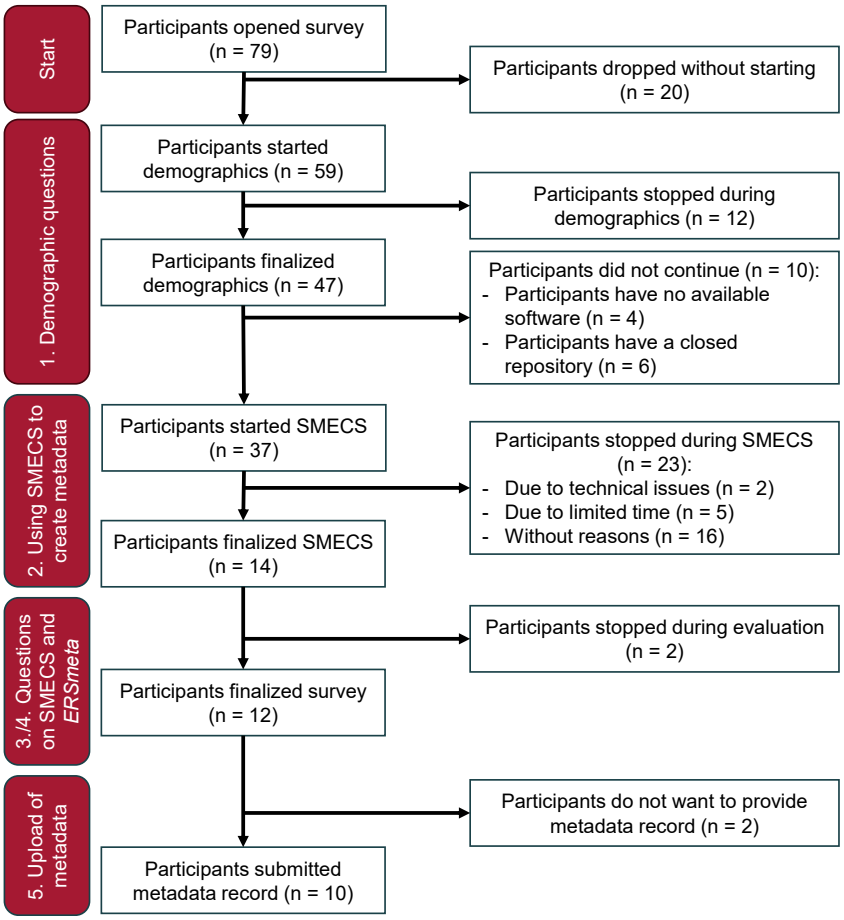


Figure 6.2: Overview of survey completion and dropout reasons.

quantitative data were processed with descriptive statistics by calculating means, medians, and inter-quartile ranges. Qualitative comments underwent a broad content-analysis to identify recurring themes and suggestions. All analysis and visualization scripts, including the code generating the figures in this chapter, are released as open source software [72]. Section 6.2 presents the empirical findings.

6.2 Results and Limitations

Results The following presents an analysis of the answers provided by the twelve participants who completed the full questionnaire. They held a variety of positions within their research institutions, including research-supporting staff, PhD students, postdoctoral researchers, and other research staff. Their disciplinary backgrounds comprised economics (1), computer science (5), and engineering (6). All participants reported experience with

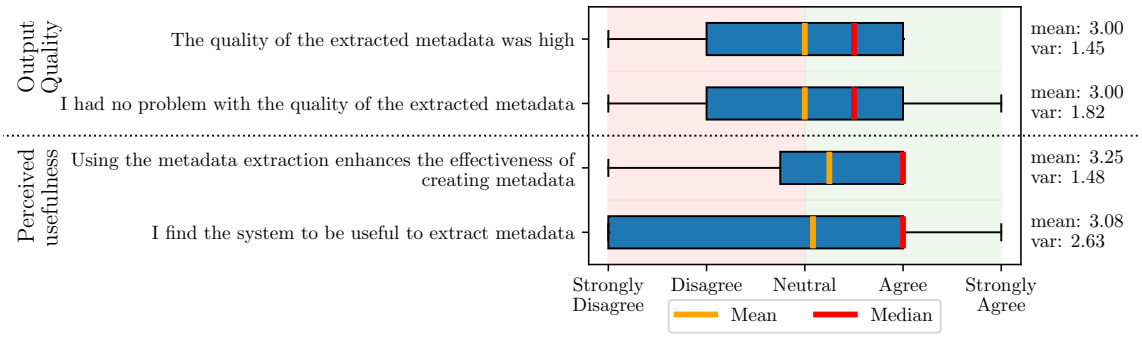


Figure 6.3: Impression of the extraction capability of SMECS (n=12).

ERS. Two participants indicated no familiarity with metadata schemas, a factor that later proved relevant for their perception of the tool and the schema. Even though the analysis only consists of twelve data points for each survey item, an aggregated visualization was chosen to present a clearer overview.

When evaluating SMECS, the participants rated the quality of the automatic extraction on average neutral, as illustrated in Figure 6.3. When looking at the median, both the functionality of the extraction component and the overall quality of the extracted metadata received slightly positive evaluations. However, the free-text responses highlighted a desire for improvements in the quantity and precision of the extracted information. This desire was also reflected in the rating of prospective extensions of SMECS which are summarized in Figure 6.4. The majority expressed a need for broader extraction capabilities (e.g., information from package repositories and information from related publications). Participants also slightly indicated a willingness to directly publish the resulting metadata in a dedicated software registry, although this aspect was not mentioned in the free-text responses.

The SUS scores are shown in Figure 6.5. Over all participants, the mean of 61.75 lies

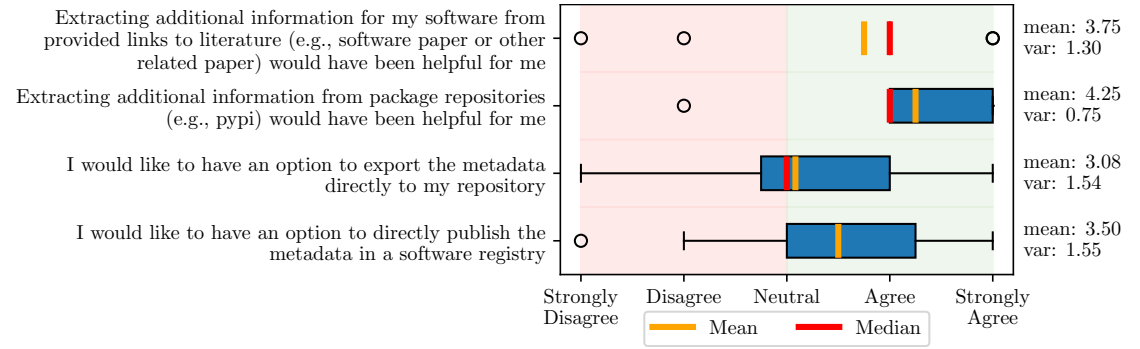


Figure 6.4: Attitude towards additional features in SMECS (n=12).

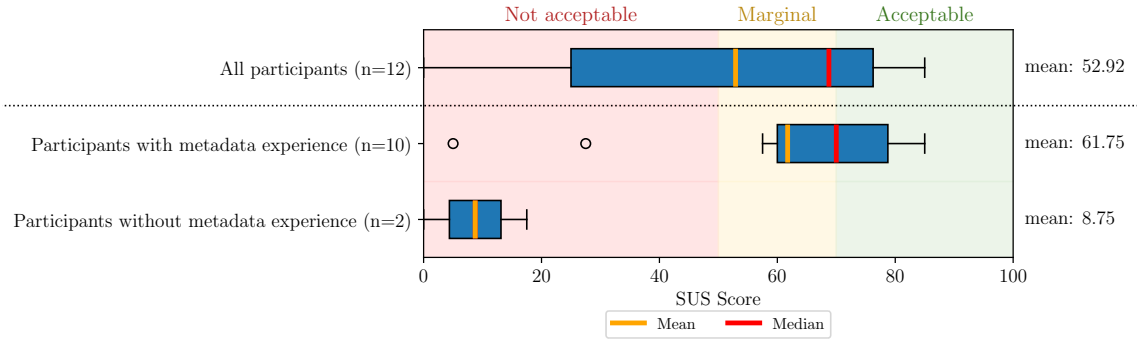


Figure 6.5: SUS scores for participants with and without familiarity with metadata schemas and the overall distribution.

in the marginal range. Participants without metadata schema experience (n=2) gave clear low scores, reflecting a more critical view. This trend is also observed in other survey items, suggesting that a lack of experience or motivation for metadata schemas may contribute to a more negative perception of SMECS and *ERSmeta*. It is assumed that this missing experience affects the various survey items in different ways, while a detailed analysis of this effect is not possible due to the small sample size. Regarding the SUS, the items that most contributed to the reduction in SUS scores concerned perceived complexity ("I found the system unnecessarily complex") and the missing desire to use the system frequently ("I think that I would like to use this system frequently"). This criticism is assumed to be primarily caused by the high number of elements in *ERSmeta*, as the free-text responses revealed that participants were concerned with the high number of elements and the time-consuming nature of the metadata creation process. This is additionally reflected in the participants' responses to the items related to the amount of elements, as shown in Figure 6.6. Notably, the majority of participants agreed that "The amount of elements is too high", indicating that participants prefer less elements to be filled.

Nevertheless, the usefulness of the elements was rated positively, indicating that the element set contributed to a rich description of ERS. This is further supported by the free-text responses, where participants mentioned that the metadata were relevant and helpful. The analysis of the ten submitted metadata records revealed that only three elements (`typicalExecutionTime`, `example`, and `estimatedCosts`) were never populated. Recommended elements were filled more frequently than optional ones, as shown in Figure 6.7, confirming that participants followed the prioritization suggested by the obligations in the schema. The distribution of populated elements across thematic areas is illustrated in Figure 6.8. Elements related to functionality were often left empty, likely because they are not applicable to every ERS and they are also optional to a large degree.

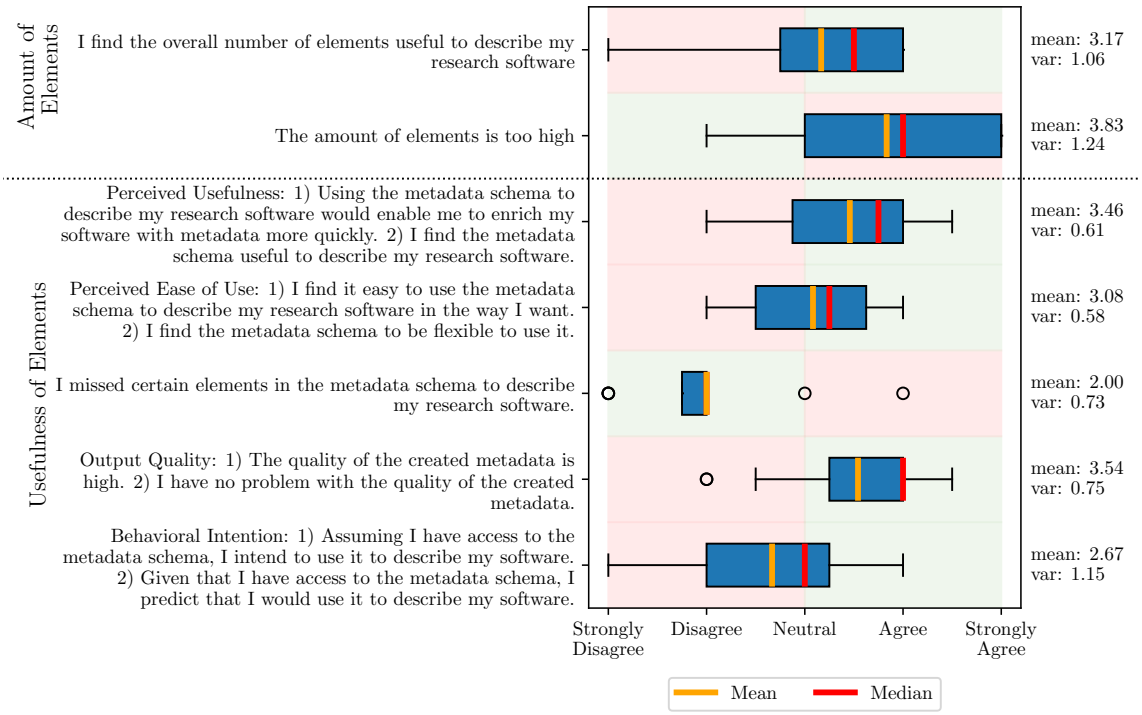


Figure 6.6: Impressions on the amount of elements and their usefulness (n=12). For the usefulness of elements, items from the construct (questions aiming for the same aspect) are averaged based on the Likert responses mapped to numbers (1 = *strongly disagree*, ..., 5 = *strongly agree*).

Almost none of the participants missed elements to describe their ERS, as shown in Figure 6.6. Also, when asked to specify missing elements in the free-text responses, only one participant mentioned four aspects. Of these, two are already included in the schema (related publication, related projects), one is out-of-scope (metadata for data), and the fourth (intended public) can be added to the metadata schema.

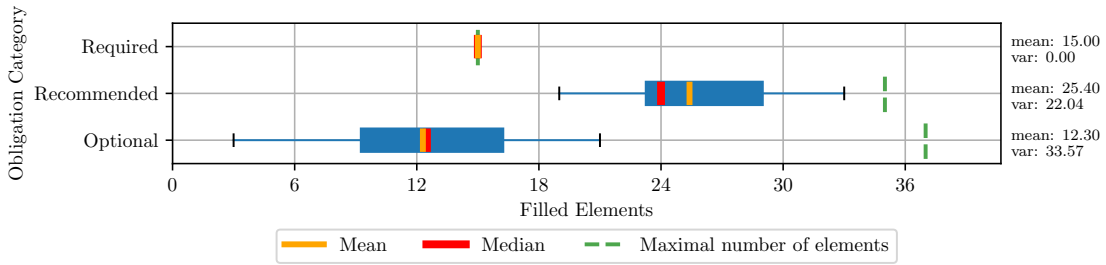


Figure 6.7: Filled elements per obligation category (mandatory, recommended, and optional) (n=10).

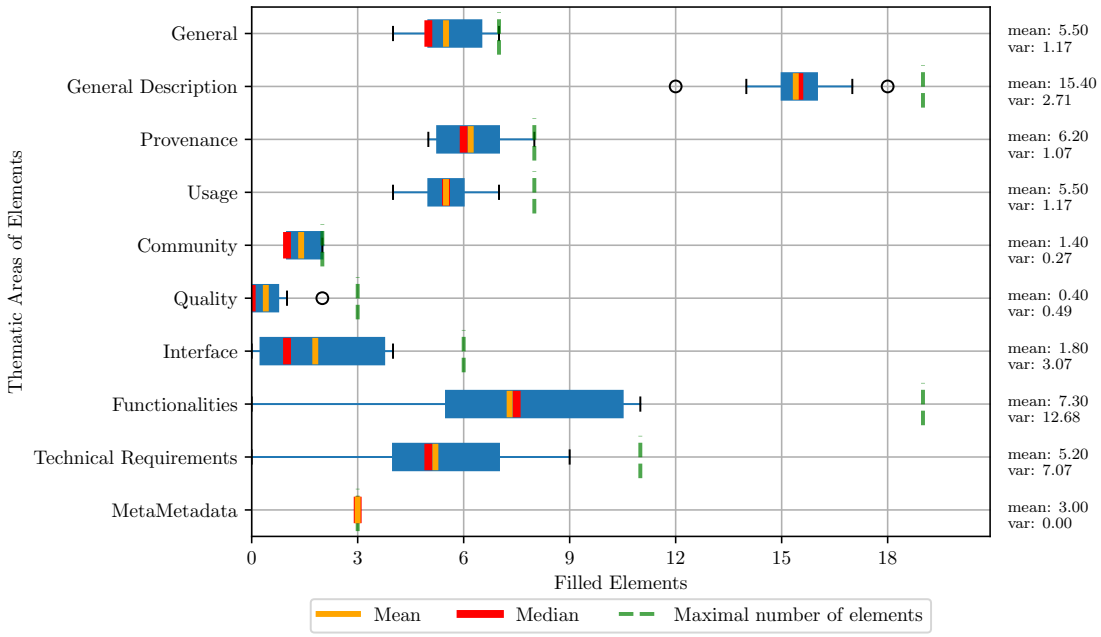


Figure 6.8: Filled elements per thematic area (n=10).

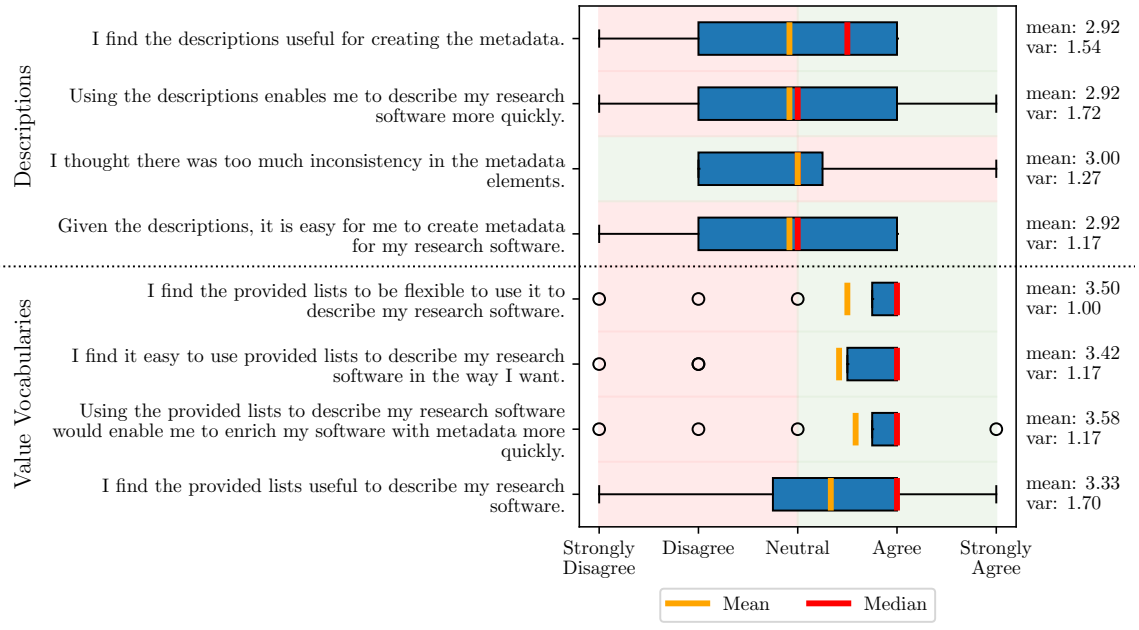


Figure 6.9: Usefulness of the element descriptions and of the controlled value vocabularies (n=12).

Figure 6.9 displays the ratings for the clarity of the element descriptions and the usefulness of the controlled value vocabularies. Participants judged the descriptions as neutral to slightly positive, indicating that they are a useful starting point but would benefit from additional information, e.g., examples as mentioned in the free-text responses. The value vocabularies received a more positive assessment. Respondents found them flexible and helpful, although a few suggested modest extensions to increase coverage.

Overall, the evaluation demonstrates that SMECS provides a useful platform for generating ERS metadata and that *ERSmeta* supplies a comprehensive, but too extensive, metadata schema. Quantitative evidence shows a modest SUS mean (61.75), solid perceived extraction quality, and positive judgments of the usefulness of the elements despite concerns about their total number. The qualitative comments complement these findings by highlighting desires for broader extraction capabilities, richer examples in element descriptions, and minor extensions to the value vocabularies. These results inform the discussion of *ERSmeta* and SMECS in the following sections (Section 6.3 and Section 6.4).

Limitations While the evaluation led to valuable insights, the study also had some limitations. The submitted metadata sets showed that the participant pool was geographically concentrated in Germany, which may have narrowed the diversity of the perspectives captured. Energy research spans many disciplines and although the requirement analysis phase incorporated a broader set of perspectives [90], the evaluation did not capture the full diversity of this research field. While developers of smaller ERS were underrepresented in the survey, it captured developers of large-scale ERS well. Therefore, it remains unclear to which extent the results also represent developers of smaller ERS. The limited number of participants prevented an analysis of how participants' background or prior experience with metadata influenced their perception of *ERSmeta* and SMECS. Nevertheless, meaningful general results concerning both artifacts were obtained.

Evaluating SMECS and *ERSmeta* together yielded valuable insights but introduced a confounding factor: participants accessed the schema only via the tool, so their impressions of *ERSmeta* were inevitably shaped by SMECS's user interface and vice versa. Even though the questionnaire was structured to address SMECS first, the feedback on *ERSmeta* remained intertwined with perceptions of the tool and vice versa. This made it difficult to isolate the specific strengths and weaknesses of each artifact.

6.3 Discussion *ERSmeta*

Despite its limitations, the evaluation in Section 6.2 delivered additional insights into *ERSmeta* while Chapter 4 describes the development process of *ERSmeta* and the schema

itself. Also, that chapter introduces the Core Metadata Requirements (CMRs) for the metadata schema, summarized in Table 4.1, which are all addressed by the schema. The most relevant aspects of *ERSmeta* are brought together and discussed in the following.

Coverage of relevant aspects *ERSmeta* is based on a detailed requirements analysis based on over 30 interviews with energy researchers and on a profound analysis of relevant RSE literature as described in Section 4.2. The evaluation revealed that *ERSmeta* covers all aspects deemed relevant by the participants. No substantial gaps were reported and nearly no additional elements were suggested. Also, nearly all elements of *ERSmeta* were used by the participants of the evaluation.

Still, there are some small areas to include additional information in *ERSmeta*, e.g., more detailed information on software interfaces or a better representation of research software quality. Their integration would need to be handled with care since the size of the metadata schema which was already criticized as complex in the evaluation.

Descriptions and value vocabularies For the elements, mainly existing descriptions were reused in *ERSmeta*. Still, the descriptions only received a neutral to slightly positive rating in the evaluation, indicating that they are usable but could benefit from additional examples and clarification. In contrast, the controlled value vocabularies were rated positively and were highlighted by the participants as a strong point of the schema. These vocabularies contributed to the perceived usefulness of the metadata and support consistent, machine-readable annotation.

Interoperability Due to its formalization, *ERSmeta* is more flexible and interoperable than the previous, informal approaches described in Section 4.1. For example, this formalization enabled the seamless integration of *ERSmeta* into SMECS.

Also, numerous controlled value vocabularies were incorporated into *ERSmeta* to ensure consistent, interoperable, and machine-readable metadata. In this way, the schema supports additional use cases such as Software Management Plans (SMPs) or simulation coupling.

Moreover, *ERSmeta* reuses a large number of elements from established metadata schemas and ontologies, especially from *Schema.org* and *CodeMeta* as shown in Section 4.5. This ensures interoperability with those sources and allows the reuse of the resulting metadata in a variety of contexts. The compatibility with *CodeMeta* additionally supported the integration of *ERSmeta* into SMECS because the metadata extraction of SMECS (and HERMES) is based on *CodeMeta*. Also, future extensions to *CodeMeta* can be incorporated into *ERSmeta* with minimal effort.

Relevance for domain-agnostic research software metadata While most elements of *ERSmeta* are reused from existing standards, some new elements were introduced. Part of them are domain-agnostic, e.g., `communityInteraction`, and could be valuable additions to generic schemas such as *CodeMeta*. Also, a few *CodeMeta* elements were refined, e.g., with value vocabularies which are potentially also relevant for *CodeMeta*.

Overall, *ERSmeta* includes relevant metadata for ERS and shows their formalization as metadata schema. The evaluation presents a mixed assessment of *ERSmeta*: the elements of *ERSmeta* were rated useful, yet the overall number of elements resulted in complexity, a clear drawback of the schema. Nevertheless, the findings provide input to refine the schema. Further research, particularly within a software registry context, is needed to further assess usability. Still, *ERSmeta* is already usable to describe ERS in a structured, machine-readable way and, therefore, contributes to the realization of FAIR ERS.

6.4 Discussion SMECS

Chapter 5 describes the development of SMECS and outlines its first initial evaluation. The development of the tool is based on Software Requirements (SRs) which are summarized in Table 5.1. The development of SMECS addressed all SRs for the basic features, such as extraction of structured metadata, provision of a user-friendly interface, and export of a JSON file. Several SRs for extended features (e.g., support for unstructured extraction, advanced export destinations, and richer export formats) remain only partially addressed and will require further investigation. Additionally, the evaluation brought further insights into SMECS. These different facets of SMECS are discussed in the following.

Extraction capabilities and future extensions The current metadata extraction of SMECS covers *Citation File Format (CFF)* and *CodeMeta* files as well as the GitHub and GitLab APIs. Participants rated the overall extraction quality positively, yet they also highlighted the need for more harvesting functionalities. Table 5.2 outlines different sources that could be incorporated while SR9 explicitly calls for extraction from unstructured files. Because README files are the most prevalent description in research software repositories [120], integrating functionalities from SoMEF [158], which can parse README files, Dockerfiles, and other free-text resources, would substantially broaden the coverage of SMECS. By using HERMES as foundation for the extraction of metadata, further metadata harvesters can be used in a broader context. Also, SMECS can directly profit when additional harvesters become available as part of HERMES.

Frontend usability The initial evaluation of SMECS (see Section 5.4) that employed *CodeMeta* as the underlying schema yielded high SUS scores with a mean of 92.02. These contrast with the lower scores (mean: 61.75) in the evaluation with *ERSmeta* presented in this chapter. Two factors may explain this discrepancy: first, the initial evaluation used a more direct interview methodology, possibly leading to some degree of a social desirability bias. Second, *ERSmeta* contains substantially more elements than *CodeMeta*. Besides improving *ERSmeta*, additional features in the frontend of SMECS may address the second factor, e.g., mechanisms to hide or collapse non-essential elements or to adapt the displayed elements based on the number of automatically extracted elements or on the user's expertise level. Generally, users with less metadata experience rated the interface more negatively indicating potential to especially improve the frontend for this user group.

Export and publishing functionality While SMECS can currently export metadata as a JSON-LD file, participants expressed a desire for more integrated publishing options. One option is to include a direct upload to the repository of the research software (e.g., on GitHub or GitLab), so the metadata becomes directly available next to the source code. Another option is a direct upload of the metadata to software registries or knowledge bases such as the Open Research Knowledge Graph (ORKG) as highlighted by the SR11. Extending the export module to support these destinations, e.g., by integrating such functionality from HERMES, would close a relevant gap in the workflow and strengthen compliance with the FAIR4RS principles which require the registration of a research software in a registry in addition to providing good metadata for it.

Semantic web limitations The frontend of SMECS uses the information of a metadata schema provided as JSON-LD Schema since *CodeMeta* uses this format. Although JSON-LD enables basic semantic web capabilities, it offers limited support for dynamic ontology-based value vocabularies and complex class hierarchies. Consequently, fulfilling SR10 (dynamic value-vocabulary integration) is challenging within the current technical stack. Investigating alternative formats for the metadata schema (e.g., SHACL) could provide the necessary expressivity for future extensions.

Overall, SMECS proves to be a usable and interoperable tool for generating metadata for research software. By building on HERMES, it inherits a solid foundation for metadata harvesting. Nevertheless, opportunities for improvements remain, especially in expanding the extraction to unstructured sources, simplifying the user interface, enhancing export features, and overcoming the semantic web constraints. Continued research, e.g., with more diverse participant groups and deeper integration into software registry ecosystems, will be essential to fully realize the potential of SMECS as a catalyst for FAIR research software.

7 Summary and Outlook

While ERS is an important foundation for energy research, it faces multiple challenges, e.g., missing documentation, leading to limited reproducibility and reusability. The FAIR4RS principles provide a general framework to improve the reusability of research software, but their adoption in the energy domain is still at the beginning. Therefore, the overarching goal of this thesis was to identify process steps and define resources that can enable FAIR ERS. In the following, Section 7.1 first summarizes the key findings of this thesis along with answers to the RQs. Subsequently, Section 7.2 provides an outlook on future work including open research challenges to achieve fully FAIR ERS and improve the environment for RSE.

7.1 Key Findings

Section 1.2 introduced a conceptual ecosystem composed of resources that support and enable FAIR ERS. An overview of the ecosystem is published in [89] and visually summarized in Figure 7.1. Based on this conceptualization, Section 1.3 formulated three RQs for this thesis which target three central resources, marked in magenta in Figure 7.1: the recommendations for engineering ERS, the metadata schema, and the metadata generation tool. In the following, the key findings for each RQ are presented including the related main contributions of this thesis and their limitations.

7.1.1 Recommendations for Engineering Energy Research Software

RQ1: Which recommendations can help energy researchers to develop more reusable ERS?

Key findings Chapter 3 presented a set of ten recommendations for energy researchers that was derived from three sources. An initial literature review identified the most relevant RSE topics, which formed the basis for a first exploratory workshop with energy researchers. The outcomes of this workshop were refined in a second consolidating workshop.

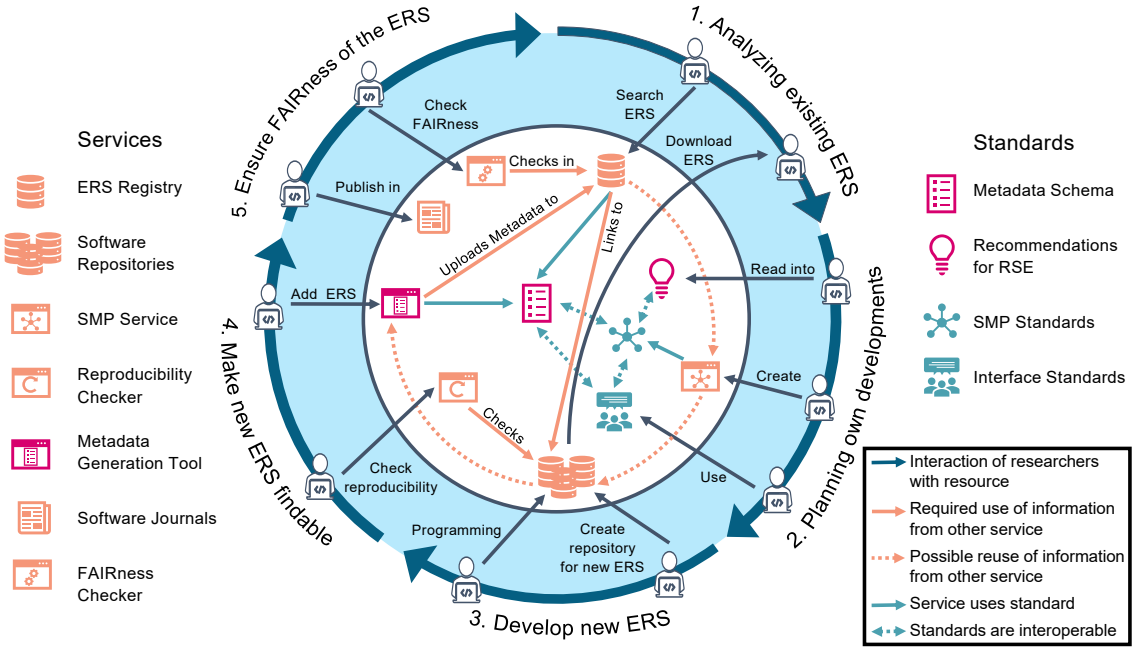


Figure 7.1: Vision of an ecosystem enabling FAIR ERS supported through resources (services and standards) extending Figure 1.3. The resources focused in this thesis are highlighted in magenta.

The combined results constitute the final recommendations published in [77]. Generally, the recommendations align with existing domain-agnostic research software guidance and especially emphasize energy-specific aspects such as industry collaboration, interdisciplinary collaboration, and the explicit inclusion of RSE in project proposals. Besides the recommendations, the workshops conducted in this research step suggest that training of RSE skills should be focused early in research projects. Since the implementation of the recommendations depends on existing infrastructure for ERS as presented in Section 1.2, they need to be revised as new resources emerge.

Limitations The recommendations resulted from the two interactive workshops which were mainly joined by energy researchers with many practical experience. Consequently, the insights reflect the perspectives of practitioners but may not capture the latest advances in software engineering. Additionally, due to the heterogeneity of ERS projects the recommendations cannot be applied uniformly but the special characteristics of each project need to be taken into account.

7.1.2 Metadata Schema for Energy Research Software

RQ2: Which metadata are suited to improve the FAIRness of ERS and how can they be structured within a metadata schema?

Key findings Chapter 4 introduced *ERSmeta* as a metadata schema specifically designed to improve the FAIRness of ERS. Its metadata elements are based on the requirements derived from more than 30 semi-structured interviews with energy researchers, published in [90], and enriched by insights from the RSE literature. The elements were combined to a formalized schema, available as JSON-LD and SHACL, demonstrating how the metadata can be structured in a machine-readable way. The formalized schema includes descriptions, obligation levels, and data types for all elements. By reusing many elements from *CodeMeta*, *ERSmeta* is interoperable with domain-agnostic research software metadata and can accommodate future extensions of *CodeMeta* with minimal effort.

The schema was evaluated positively in a quantitative in-vivo study presented in Chapter 6. The results, submitted as [91], showed that the metadata elements were rated as useful and relevant without identifying substantial gaps. This indicates that the schema captures relevant aspects of ERS. In summary, *ERSmeta* constitutes an important first step towards a community-wide metadata standard for ERS while already being ready to describe ERS in a machine-readable way.

Limitations The interview-based requirement gathering was centered on the perspectives of energy researchers, ensuring that the resulting metadata elements are directly relevant to their daily practice. Perspectives from other key stakeholders, such as RSE/HPC units or funding agencies, were incorporated only indirectly via the RSE literature. Also, the viewpoint of industry was not focused by the requirement analysis. Consequently, the requirements may be incomplete with respect to the needs of these additional stakeholders.

Regarding the element descriptions, the evaluation yielded neutral to slightly positive ratings highlighting the need for more explicit documentation and concrete examples. Additionally, concern about the high number of metadata elements was expressed in the evaluation, noting that a large element set can impose a substantial burden on those creating the metadata. Finally, *ERSmeta* has so far only been integrated into SMECS and not in any software registry or other ecosystem components, which limited the insights gathered from the current evaluation to a specific environment.

7.1.3 Metadata Extraction and Curation

RQ3: How can a tool support the creation of metadata for ERS?

Key findings Chapter 5 presented the current landscape of tools that support research software metadata creation and introduced SMECS as a novel solution that combines multiple aspects and is published in [81]. SMECS can automatically extract existing information for a software from its repository (GitHub, GitLab), provides them in a web-based, user-friendly interface for curating, and can export them as a JSON-LD metadata record following *CodeMeta* or *ERSmeta*. For the extraction, SMECS uses the framework HERMES and currently supports extraction from *CFF* and *CodeMeta* files as well as from the GitHub and GitLab APIs.

The initial usability study of SMECS yielded an excellent mean System Usability Scale (SUS) score, as presented in Section 5.4. A marginal mean SUS score was produced by a subsequent study in which *ERSmeta* was integrated into SMECS as described in Chapter 6. This decline is probably due to the larger number of metadata elements introduced by *ERSmeta* and, possibly, to a social-desirability bias in the first, interview-based assessment. In summary, SMECS constitutes a usable and interoperable platform that supports the creation of ERS metadata, laying the groundwork for FAIR-compliant metadata workflows for research software.

Limitations While SMECS is designed to be applicable beyond ERS, the evaluation was conducted exclusively with energy researchers. As a result, the extent to which the findings can be generalized to other research domains remains uncertain. The current extraction module is limited to a few structured sources (*CodeMeta* and *CFF* files as well as GitHub and GitLab APIs). It does not handle unstructured documentation, which restricts its applicability. Moreover, the evaluation revealed that working with large metadata schemas such as *ERSmeta* is demanding, contributing to the lower usability scores observed in the second study. Finally, the frontend's reliance on JSON-LD offers only basic semantic web functionality, impeding the integration of dynamic, ontology-based value vocabularies.

7.1.4 Summary

Beyond the artifact-specific considerations, several overarching limitations affect all findings that should be acknowledged. Even when an international perspective was sought, the data gathering from the energy research community had a German focus, e.g., in the requirement analysis for *ERSmeta* approximately 80 % of interviewees were based in Ger-

many. This geographic concentration likely narrowed the diversity of viewpoints captured. In addition, although the participants formed an interdisciplinary cohort, the inherently broad and heterogeneous nature of energy research may have led to some subcommunities and disciplinary perspectives not being fully represented.

Overall, this thesis establishes a solid foundation for a deeper understanding of ERS and for advancing its engineering towards FAIR ERS. Thereby, the thesis achieved its primary objective. As prototypes for three resources, three concrete artifacts were created to support researchers in this endeavor: First, a set of ten domain-specific recommendations offers practical guidance for developing reusable ERS. Second, the metadata schema *ERSmeta* structures the information required for making ERS FAIR. Third, the tool SMECS simplifies the creation of ERS metadata. While technical solutions such as *ERSmeta* and SMECS are indispensable for lowering the practical barriers to FAIR compliance, the findings also underscore the need for a cultural change. The workshops and interviews revealed that research software is still undervalued as a scientific contribution, often being seen merely as a means to generate results rather than as a citable, reusable research output. Recognizing and rewarding high-quality, well-documented research software is therefore a prerequisite for encouraging researchers to invest the additional effort to improve the quality and FAIRness of their ERS. The introduced artifacts can help catalyze this cultural change, paving the way towards more transparent, reproducible, and reusable ERS.

7.2 Future Work

The work presented in this thesis not only achieved its primary objective but also revealed a number of promising avenues for further work. While some of them arose from the boundaries of the thesis, others emerged directly from the findings and the development of the three artifacts. The following section outlines the most compelling directions. While some of them constitute research topics, others represent additional features for the artifacts or necessary steps to enable their broader adoption. Table 7.1 summarizes and clusters these various aspects.

Recommendations Because ERS is highly heterogeneous and the presented recommendations target all types of ERS, energy researchers must adapt them to the specific type and maturity of their ERS. Future research can analyze how the recommendations can be tailored for the different types of ERS to provide more nuanced, context-sensitive advice. To this end, a better understanding of the different types of ERS is necessary. This can be achieved by extending the multi-dimensional research software categories proposed by Hasselbring et al. [111] with domain-specific aspects.

Table 7.1: Overview of future work: further research, further features, and adoption.

	Recommendations	***ERSmeta***	**SMECS**	**Further Resources for FAIR ERS**	**Impact of Resources**
Further Research	Usage in other research domains	Detailed interface description (e.g., *DataDesc* [150])	AI-based approaches for metadata extraction	Characteristics of ERS	Support of cultural change through the resources
	Tailored recommendations for different types of ERS	Effect of metadata experience on *ERSmeta* perception	Effect of metadata experience on the perception of SMECS	Tailoring of general services for energy research	
		Software quality metrics in a metadata schema	Usage of SMECS in other research domains		Adapting the artifacts in other research domains
		Including additional information without increasing complexity	Interactive ways to deal with large metadata schemas	Interactions between resources	
		Industry perspective		Measurable research software quality metrics	Application of the methodical approaches in other domains
		Interaction between software and data metadata			
		Usability in a registry			
		Domain-agonistic elements from *ERSmeta* for *CodeMeta*			
Further Features	Keep recommendations aligned with new resources	Better value vocabularies	Metadata schema as SHACL		
		Better descriptions (e.g., with examples)	Additional extraction capabilities (e.g., from unstructured sources)		
			Direct upload of metadata to software registries		
			Direct upload of metadata to the software repository		
Adoption	Dissemination to energy researchers	*ERSmeta* in a registry	Integration in a software registry		
	Usage support for energy researchers	Forster usage by energy researchers	Integration in a SMP tool		
			Available online service		

In addition, the transferability of the recommendations to other research domains is interesting for further investigation and may also lead to a better understanding of the variations of RSE in different research domains.

The recommendations should be further updated when new resources become available and the community evolves. To facilitate ongoing community engagement, a moderated GitHub repository[1] was established where researchers can discuss the recommendations and provide feedback. Finally, the recommendations should be further disseminated, so that energy researchers can use them for their ERS development. This is planned within the consortium NFDI4Energy.

ERSmeta Table 7.1 outlines multiple new features that may improve *ERSmeta* and shows various aspects for further research. One of them concerns the provision of richer, machine-readable interface descriptions, similar to *DataDesc* [150]. Additionally, ongoing advances in research software, e.g., regarding quality metrics, should be monitored closely so they may be integrated into *ERSmeta*. To include these additional information, it is necessary to investigate how they can be added without increasing the perceived complexity of the schema. Especially regarding the perceived complexity, further investigations of the effect of prior metadata experience and training is relevant.

By integrating *ERSmeta* into a dedicated software registry, additional insights into its impact on the discoverability of ERS can be gained. Within NFDI4Energy, the development of such a registry with *ERSmeta* as the underlying metadata model is planned [180], providing a concrete evaluation environment and fostering broader community uptake.

Finally, the schema's contributions should be carefully examined for relevance to other metadata standards. For example several newly introduced, domain-agnostic elements, e.g., the element `communityInteraction` in *ERSmeta*, could be valuable additions to generic frameworks such as *CodeMeta*. Thereby, the impact of *ERSmeta* can be extended beyond the energy domain strengthening the overall FAIR research software ecosystem.

SMECS The functionalities of SMECS can be improved in multiple ways as summarized in Table 7.1. The evaluation identified a need for broader harvesting, especially from unstructured sources like README files. Incorporating parsers, e.g., from SoMEF, and exploring AI-driven techniques for extraction will increase coverage. HERMES as foundation of SMECS allows new harvesters to be plugged in as they become available. In further research, it can be investigated which mechanisms can deal with larger metadata schemas to increase the usability of SMECS. In terms of export, SMECS should support direct uploads to software registries and knowledge graphs, e.g., to the Open Research Knowledge

[1] `https://github.com/NFDI4Energy/ERSE_Recommendations`, last accessed 2026-03-03

Graph (ORKG), to close a workflow gap. Some of these functionalities will be included into SMECS as part of the project Connected Open Source Software (ConnOSS) [35]. They would broaden SMECS's applicability and improve its user experience.

By offering a public instance of SMECS, it can be evolved into a research infrastructure software. Additionally, embedding SMECS into existing software registries or using it to populate sections of SMPs would streamline metadata creation and establish SMECS as a central component for handling metadata creation for research software. This would allow further research on SMECS, e.g., how its usability perception depends on different metadata experience levels and on its usability in other domains. Furthermore, its adoption across the energy research community and beyond can be promoted.

Further resources for FAIR ERS The envisioned ecosystem, as shown in Figure 7.1, contains multiple resources that can simplify and enable FAIR ERS. While three of them were already covered through the artifacts of this thesis, further resources are needed for the ecosystem, e.g., a software registry or interface standards. To tailor those resources to the needs of energy researchers, a deeper understanding of the characteristics of ERS is necessary. Therefore, further research on the special characteristics of ERS is needed. Also, the required interactions between the resources in the ecosystem need further investigation, e.g., how to synchronize up-to-date metadata between different resources.

Another critical area to enable reusable ERS concerns the assessment of its quality and its overall maturity. Interview participants in the presented requirement study (see Section 4.3) stressed that perceived quality is a relevant factor for reuse. Standardized quality metrics can simplify the selection of ERS for reuse and enable better targeted funding for ERS. Yet, reliable quality metrics and maturity models are currently lacking for research software. Developing them based on existing software engineering approaches, like the Technology Readiness Levels (TRLs), would provide a systematic way to evaluate and communicate the robustness of ERS, thereby supporting the FAIR4RS principles.

Impact of resources The evaluations presented in this thesis showed that the three developed artifacts generally support the FAIRification of ERS. However, the broader behavioral consequences of making these resources available have not yet been quantified. Therefore, future research should analyze to which extent the artifacts stimulate a cultural change towards FAIR ERS, e.g., by measuring changes in metadata creation practices or reuse patterns among researchers. In parallel, it should be investigated to which degree the developed artifacts can be adapted to other research domains and if the chosen methods are reusable for other domains. Thereby, a wider cultural change towards FAIR and more reusable research software can be fostered across science.

A Related Publications and Outcomes

In the course of this thesis a number of peer-reviewed publications, conference presentations, poster contributions, and research artifacts were produced. This first part of the appendix provides an overview of these outputs and clarifies the author's role for each. Therefore, the contributions of all co-authors are classified according to the Contributor Role Taxonomy (CRediT)[1] for each listed work to highlight the specific responsibilities of the thesis author. First, Section A.1 lists the publications that lie directly within the scope of this thesis. Section A.2 then summarizes additional papers that are closely related to the research in this thesis. Regarding the dissemination activities at scientific conferences, Section A.3 summarizes oral presentations that are not directly connected to a publication, while Section A.4 documents the poster sessions in which results of this thesis were showcased. Finally, Section A.5 presents research artifacts that were created and released as part of this work including research data, research software, and a metadata schema.

A.1 Within the Scope of this Work

The following publications present early and final results of the research outlined in this thesis, directly contributing to the different chapters. Table A.1 shows the contributions of the authors for each publication based on CRediT.

Towards Improved Findability of Energy Research Software by Introducing a Metadata-based Registry (first author, 2023) [84] This publication introduces and formally defines Energy Research Software (ERS) for the first time. It motivates that describing ERS with rich metadata and registering it in a metadata-based registry are essential aspects to improve its findability and overall FAIRness. Building on a review of the state of the art, the paper proposes an initial set of requirements for both the metadata schema and the registry.

This publication laid the foundation for the conceptualization in Chapter 1 and the early analysis of the state of the art of metadata schemas outlined in Chapter 4. The work was

[1] https://credit.niso.org/, last accessed 2026-03-03

presented at the *NFDI4Ing Conference 2022* by the author of this thesis and subsequently published as a post-conference proceeding in the journal *ing.grid*[2].

Table A.1: CRediT roles of all authors for all related publication within the scope of this work.

Author	Conceptualization	Formal analysis	Investigation	Methodology	Project administration	Visualization	Writing – original draft	Writing – review & editing
Towards Improved Findability of Energy Research Software by Introducing a Metadata-based Registry [84]								
Stephan Ferenz	✓						✓	✓
Astrid Nieße	✓							✓
Ten Recommendations for Engineering Research Software in Energy Research [77]								
Stephan Ferenz	✓		✓	✓	✓	✓	✓	✓
Emilie Frost	✓		✓				✓	✓
Rico Schrage	✓		✓				✓	✓
Thomas Wolgast	✓		✓				✓	✓
Inga Beyers	✓					✓		✓
Oliver Karras	✓							✓
Oliver Werth	✓			✓				✓
Astrid Nieße	✓			✓				✓
Requirements for a Metadata Scheme to Enable FAIR Energy Research Software [90]								
Stephan Ferenz	✓	✓	✓	✓		✓	✓	✓
Oliver Werth	✓			✓				✓
Astrid Nieße	✓			✓				✓

Continued on next page

[2] https://www.inggrid.org/, last accessed 2026-03-03

Table A.1: CRediT roles of all authors for all related publication within the scope of this work (continued).

Author	Conceptualization	Formal analysis	Investigation	Methodology	Project administration	Visualization	Writing – original draft	Writing – review & editing
SMECS: A Software Metadata Extraction and Curation Software [82]								
Stephan Ferenz	✓			✓			✓	✓
Aida Jafarbigloo	✓	✓	✓	✓		✓	✓	✓
Oliver Werth	✓			✓				✓
Astrid Nieße	✓			✓				✓
Towards a Metadata Schema for Energy Research Software [91]								
Stephan Ferenz	✓		✓	✓		✓	✓	✓
Oliver Werth	✓			✓				✓
Astrid Nieße	✓			✓				✓
Enabling FAIR Energy Research Software [89]								
Stephan Ferenz	✓					✓	✓	✓
Oliver Werth	✓							✓
Astrid Nieße	✓							✓

Ten Recommendations for Engineering Research Software in Energy Research (first author, 2025) [77] This article introduces ten recommendations to improve the engineering of ERS. The recommendations span the full software lifecycle including conceptualization, development, testing, and publication. They are based on the outcomes of two interdisciplinary workshops with energy researchers. By raising awareness of RSE best practices, the recommendations are intended to increase the quality and reproducibility of energy research.

The recommendations and their creation are discussed in Chapter 3, while Appendix B presents all recommendations. The paper was presented at the *16th ACM International*

Conference on Future and Sustainable Energy Systems (ACM e-Energy 2025)[3] by the author of this thesis and is part of the conference proceedings[4].

Requirements for a Metadata Scheme to Enable FAIR Energy Research Software (first author, 2025) [90] This paper reports a qualitative requirement analysis for a FAIR-compliant metadata schema for ERS. The study identified a set of metadata elements, e.g., community information, support options, and domain-specific properties like temporal and geographical scope, that are essential for FAIR ERS. Based on these requirements, a domain model was constructed, providing the conceptual foundation for a metadata schema for ERS.

The requirement analysis is presented in Section 4.3 as part of the development of *ERSmeta*. The results were first presented at the *4th Conference for Research Software Engineering in Germany (deRSE 2024)*[5] by the author and later published in the post-conference proceedings[6].

SMECS: A Software Metadata Extraction and Curation Software (first author, 2025) [82] The publication describes SMECS, a tool that can automatically harvest software metadata from a public repository (e.g., GitHub) and offers an interactive web interface for manual curation and export of the metadata. By simplifying the creation of metadata, SMECS can lower the barrier to FAIRifying research software. A usability study demonstrated that SMECS delivers a satisfactory user experience.

The related work, architecture, and evaluation results are detailed in Chapter 5. The software was presented at the *5th Conference for Research Software Engineering in Germany (deRSE 2025)*[7] by the author of this thesis and subsequently appeared in the post-conference proceedings[8].

Enabling FAIR Energy Research Software (first author, 2026) [89] This article refines the definition of ERS and surveys the ecosystem of services required to make ERS FAIR. Thereby, it provides the state of the art for each service category for ERS.

The definition of ERS is presented in Section 1.1 in this thesis. The general overview of potential services is described in Section 1.2. The overview of the state of the art of all services summarizes different results of this thesis. The work was presented at the *3rd*

[3] https://energy.acm.org/conferences/eenergy/2025/, last accessed 2026-03-03
[4] https://dl.acm.org/doi/proceedings/10.1145/3679240, last accessed 2026-03-16
[5] https://events.hifis.net/event/994/contributions/7927/, last accessed 2026-03-03
[6] https://eceasst.org/index.php/eceasst/issue/view/205, last accessed 2026-03-03
[7] https://events.hifis.net/event/1741/contributions/13331/, last accessed 2026-03-03
[8] https://eceasst.org/index.php/eceasst/issue/view/207, last accessed 2026-03-03

NFDI4Energy Conference[9] by the author of this thesis and is included in the conference proceedings[10].

Towards a Metadata Schema for Energy Research Software (first author, submitted to PeerJ Computer Science) [91] This manuscript presents the metadata schema *ERSmeta* and reports on an empirical evaluation in which researchers applied the schema to describe their own software artifacts.

The schema description is integrated in Chapter 4, while the evaluation design, results, and discussion are described in Chapter 6. The paper has been submitted to *PeerJ Computer Science*[11] and is currently under review. A preprint is published on arXiv[12].

A.2 Close to the Scope of this Work

While the following publications are not part of the core of this thesis, they provided valuable insights that informed the research presented herein. Many of them arose from collaborations with multiple research groups fostering scientific exchange. The specific contributions of each author are shown for these publications in Table A.2 using CRediT.

Reactive Power Markets: A Review (second author, 2022) [222] This paper offers a comprehensive review of reactive power markets, describing their main characteristics and design options (both local and system-operator level). Thereby, it summarizes current trends and research gaps based on the literature synthesis and formulates general research recommendations to guide future work in this field.

The study underlines the necessity of clear classifications for approaches used in ERS to foster reuse and interoperability. The article was published in *IEEE Access*[13].

[9] https://events.hifis.net/event/3058/contributions/22331/, last accessed 2026-03-04
[10] https://www.tib-op.org/ojs/index.php/ocp/issue/view/189, last accessed 2026-03-28
[11] https://peerj.com/computer-science/, last accessed 2026-03-04
[12] https://arxiv.org/, last accessed 2026-03-29
[13] https://ieeeaccess.ieee.org/, last accessed 2026-03-04

Table A.2: CRediT roles of all authors for all related publication close to the scope of this work.

Author	Conceptualization	Data curation	Formal analysis	Investigation	Methodology	Project administration	Software	Supervision	Validation	Visualization	Writing – original draft	Writing – review & editing
Reactive Power Markets: A Review [222]												
Thomas Wolgast	✓			✓	✓					✓	✓	✓
Stephan Ferenz	✓			✓	✓					✓	✓	✓
Astrid Nieße	✓							✓				✓
Requirements for an Open Digital Platform for Interdisciplinary Energy Research and Practice [217]												
Oliver Werth	✓		✓	✓	✓						✓	✓
Stephan Ferenz	✓		✓	✓							✓	✓
Astrid Nieße	✓											✓
An Open Digital Platform to Support Interdisciplinary Energy Research and Practice—Conceptualization [86]												
Stephan Ferenz	✓					✓				✓	✓	✓
Annika Ofenloch	✓									✓	✓	✓
Fernando Penaherrera Vaca	✓									✓	✓	✓
Henrik Wagner	✓									✓	✓	✓
Oliver Werth	✓									✓	✓	✓
Michael H. Breitner	✓											✓
Bernd Engel	✓											✓
Sebastian Lehnhoff	✓											✓
Astrid Nieße	✓											✓

Continued on next page

Table A.2: CRediT roles of all authors for all related publication close to the scope of this work (continued).

Author	Conceptualization	Data curation	Formal analysis	Investigation	Methodology	Project administration	Software	Supervision	Validation	Visualization	Writing – original draft	Writing – review & editing
Choosing the right model for unified flexibility modeling [27]												
Jonathan Brandt	✓		✓	✓	✓		✓			✓	✓	✓
Emilie Frost	✓										✓	✓
Stephan Ferenz	✓				✓							✓
Paul Hendrik Tiemann	✓											✓
Astrid Bensmann	✓											✓
Richard Hanke-Rauschenbach	✓											✓
Astrid Nieße	✓							✓				✓
DataDesc: A framework for creating and sharing technical metadata for research software interfaces [150]												
Patrick Kuckertz	✓			✓	✓	✓			✓	✓	✓	✓
Jan Göpfert	✓			✓	✓					✓	✓	✓
Oliver Karras	✓			✓	✓	✓	✓				✓	✓
David Neuroth				✓	✓		✓				✓	✓
Julian Schönau		✓		✓	✓		✓				✓	✓
Rodrigo Pueblas				✓	✓				✓		✓	✓
Stephan Ferenz	✓			✓	✓						✓	✓
Felix Engel					✓							✓
Noah Pflugradt												✓
Jann M. Weinand												✓
Astrid Nieße								✓				✓

Continued on next page

Table A.2: CRediT roles of all authors for all related publication close to the scope of this work (continued).

Author	Conceptualization	Data curation	Formal analysis	Investigation	Methodology	Project administration	Software	Supervision	Validation	Visualization	Writing – original draft	Writing – review & editing
Sören Auer								✓				✓
Detlef Stolten								✓				✓
Choosing the Right Ontology to Describe Research Data in the Energy Domain [201]												
Alexandro Steinert	✓			✓	✓						✓	✓
Stephan Ferenz	✓				✓			✓				✓
Astrid Nieße	✓							✓				✓
Multi-Dimensional Research Software Categorization [111]												
Wilhelm Hasselbring	✓			✓						✓	✓	✓
Stephan Druskat	✓			✓							✓	✓
Jan Bernoth	✓			✓								✓
Philine Betker	✓			✓								✓
Michael Felderer	✓			✓								✓
Stephan Ferenz	✓			✓								✓
Ben Hermann	✓			✓								✓
Anna-Lena Lamprecht	✓			✓								✓
Jan Linxweiler	✓			✓								✓
Arnau Prat	✓			✓								✓
Bernhard Rumpe	✓			✓								✓
Kathrin Schoening-Stierand	✓			✓								✓
Shinhyung Yang	✓			✓								✓

Requirements for an Open Digital Platform for Interdisciplinary Energy Research and Practice (second author, 2022) [217] This article investigates the concept of an open digital platform for sharing knowledge and experience across the energy sector. Requirements were identified from 36 semi-structured interviews with diverse stakeholders and organized around five core elements: competence, best practices, repository, simulation, and transparency.

The findings provide first insights into current research software practice in the energy domain and, thereby, inform the conceptualization in Section 1.2. The methodology also inspired the requirement analysis presented in Chapter 4. The work was presented at the *Wirtschaftsinformatik 2022*[14] by Oliver Werth and was included in its proceedings[15].

An Open Digital Platform to Support Interdisciplinary Energy Research and Practice – Conceptualization (first author, 2022) [86] Building on the requirements identified in [217], this paper presents a detailed functional and technological concept for an open digital platform to support researchers and practitioners in the energy domain. The platform combines six elements: (1) Competence – matchmaking of researchers and practitioners; (2) Methods – an overview of applicable research methods; (3) Repository – discovery of data sets and models; (4) Simulation – coupling of models and data for scenario execution; (5) Transparency – publication of results and related artifacts; and (6) Core – a unified entry point that integrates the other elements. Use case discussions illustrate how the platform serves different stakeholder groups.

The conceptual design contributed to the conceptualization in Section 1.2. The article was published in *Energies*[16].

Choosing the Right Model for Unified Flexibility Modeling (co-author, 2022) [27] This contribution surveys a range of flexibility modeling approaches and evaluates five selected models against an introduced metric. The best performing models were subsequently implemented and simulated to illustrate their practical flexibility potential and to compare them more in detail.

The comparative analysis highlights the heterogeneity of existing models and underscores the need for improved reusability of ERS. The paper appeared in *Energy Informatics*[17].

[14] https://wi22.de/, last accessed 2026-03-04
[15] https://aisel.aisnet.org/wi2022/, last accessed 2026-03-04
[16] https://www.mdpi.com/journal/energies, last accessed 2026-03-04
[17] https://link.springer.com/journal/42162, last accessed 2026-03-04

DataDesc: A Framework for Creating and Sharing Technical Metadata for Research Software Interfaces (co-author, 2024) [150] The article introduces *DataDesc*, a metadata schema and framework that enable the description of software interfaces with machine actionable metadata. In addition to a dedicated metadata schema, the publication presents supporting tools that automate the collection and publication of interface documentation, thereby increasing the FAIRness of research software.

The approach is included and discussed in the related work on metadata schemas in Section 4.1 and contributes to the broader discussion of metadata schemas in this thesis. The paper was published in *Patterns*[18].

Choosing the Right Ontology to Describe Research Data in the Energy Domain (co-author, 2025) [201] This study compares eight energy domain ontologies using a set of 21 criteria grouped into four categories, thereby providing a systematic decision support for researchers. The analysis identifies the Open Energy Ontology (OEO) as the top-ranked option.

The paper informed the selection of value vocabularies for *ERSmeta* in Chapter 4. The work was presented at the *13th DACH+ Conference on Energy Informatics*[19] by Alexandro Steinert and was published in the *ACM SIGEnergy Energy Informatics Review*[20].

Multidimensional Research Software Categorization (co-author, 2025) [111] This paper proposes a multidimensional categorization for research software that combines role-based, readiness-based, developer-based, and dissemination-based categories.

The discussion that led to the taxonomy informed the conceptualization in Chapter 1 and several of the categories were incorporated into *ERSmeta* in Chapter 4. The article was published in *Computing in Science & Engineering*[21] by IEEE.

A.3 Presentations

In addition to the peer-reviewed publications and their associated presentations, many results of this thesis were disseminated through oral talks by the author of this thesis at several conferences.

[18] https://www.cell.com/patterns/home, last accessed 2026-03-04
[19] https://energy-informatics2024.org/, last accessed 2026-03-04
[20] https://dl.acm.org/toc/sigenergy-eir/2024/4/4, last accessed 2026-03-04
[21] https://www.computer.org/csdl/magazine/cs, last accessed 2026-03-04

Improved Findability of Energy Research Software by Introducing a Metadata-based Registry (NFDI4Energy Conference 2024) [83] The general idea of this thesis was presented at the *1st NFDI4Energy Conference in 2024*[22].

Towards Guidelines for Engineering of Energy Research Software (deRSE 2025) [79] The ten recommendations for RSE in the energy domain, as presented in Chapter 3, were showcased at the *5th Conference for Research Software Engineering in Germany (deRSE 2025)*[23]. The talk highlighted the domain-specific characteristics identified in the study and sparked a discussion on whether similar domain-tailored approaches are needed in other research fields.

Towards Services to Enable FAIR Research Software in a Typical Research Project Cycle (deRSE 2025) [85] The conceptual model of services, standards, and recommendations required for FAIR research software as described in Section 1.2 was presented and discussed at the *5th Conference for Research Software Engineering in Germany (deRSE 2025)*[24]. This presentation led to a paper [89].

Towards a Metadata Scheme for Energy Research Software (NFDI4Energy Conference 2025) [92] A preliminary version of *ERSmeta* was introduced at the *2nd NFDI4Energy Conference in 2025*[25]. The schema is described in full detail in Chapter 4.

Towards Common Metadata for Research Software in the NFDI (CoRDI 2025) [76] Results of the NFDI working group on research software metadata were presented at the *2nd Conference on Research Data Infrastructure (CoRDI) 2025*[26]. The discussion led to improved interoperability of the metadata schema presented in Chapter 4.

A.4 Posters

Besides oral talks, the thesis outcomes were also communicated through poster presentations at several conferences.

[22] https://nfdi4energy.uol.de/community-and-updates/1st-nfdi4energy-conference-2024/, last accessed 2026-03-04
[23] https://events.hifis.net/event/1741/contributions/13914/, last accessed 2026-03-04
[24] https://events.hifis.net/event/1741/contributions/13929/, last accessed 2026-03-04
[25] https://nfdi4energy.uol.de/community-and-updates/2nd-nfdi4energy-conference-2025/, last accessed 2026-03-04
[26] https://www.nfdi.de/cordi-2025/, last accessed 2026-03-04

Towards More Findable Energy Research Software by Introducing a Metadata-based Registry (DACH+ Conference on Energy Informatics 2022) [74, 75] The core idea of the thesis, a metadata-based registry to make Energy Research Software more findable, was displayed as a poster at the *11th DACH+ Conference on Energy Informatics*[27]. The poster subsequently led to a more detailed paper [84].

Towards Guidelines for Engineering of Energy Research Software (NFDI4-Energy Conference 2025) [78] The ten engineering recommendations for ERS, presented in Chapter 3, were summarized in a poster presented at the *2nd NFDI4Energy Conference in 2025*[28].

A.5 Published Research Artifacts

Several research artifacts were created to obtain the results presented in this thesis. These artifacts are described in the following. Table A.3 provides a concise overview of the contributions to each artifact, using an adjusted version of CRediT in which the *Software* role has been replaced by *Development*.

***ERSmeta* (Metadata Schema) [71]** *ERSmeta* is the metadata schema for ERS that was developed as part of this thesis. Its evolution is described in detail in Chapter 4. The schema is maintained on GitHub[29] to ensure full traceability of changes. Each released version has been archived on Zenodo [71].

SMECS (Research Software) [80] The Software Metadata Extraction and Curation Software (SMECS) is a software for extracting and curating research software metadata, developed within the scope of this thesis as technology research software. The full description of its architecture, implementation, and user interface is provided in Chapter 5. The code base is hosted on GitHub[30], where every change is tracked, and released versions are deposited on Zenodo [80].

[27] https://energy-informatics2022.org/, last accessed 2026-03-04
[28] https://nfdi4energy.uol.de/community-and-updates/2nd-nfdi4energy-conference-2025/, last accessed 2026-03-04
[29] https://github.com/NFDI4Energy/ERSmeta, last accessed 2026-03-04
[30] https://github.com/NFDI4Energy/SMECS, last accessed 2026-03-04

Table A.3: Contributor roles for all published research artifacts based on CRediT with *development* instead of *software*.

Author	Conceptualization	Data curation	Development	Investigation	Methodology	Project administration	Supervision	Visualization	Writing – original draft	Writing – review & editing
ERSmeta [71]										
Stephan Ferenz	✓		✓		✓			✓		
Software Metadata Extraction and Curation Software (SMECS) [80]										
Stephan Ferenz	✓		✓			✓	✓	✓		
Aida Jafarbigloo	✓		✓					✓		
Syed Sundraiz Shah			✓							
Interview Guide for the Requirements Analysis for an Application Profile/Metadata Schema for Energy Research Software [73]										
Stephan Ferenz	✓				✓				✓	✓
Dataset of the Evaluation of a Metadata Schema for Energy Research Software [87]										
Stephan Ferenz	✓	✓		✓	✓				✓	✓
Oliver Werth	✓				✓					✓
ERSmeta Evaluation Scripts [72]										
Stephan Ferenz	✓		✓					✓		

Interview Guideline for the Requirement Analysis of *ERSmeta* (Research Data) [73] The interview guideline documents the protocol used to gather requirements for *ERSmeta* in Section 4.3. The final version of the guideline has been archived on Zenodo [73]. All raw interview data are kept confidential to protect participant privacy.

Evaluation Dataset (Research Data) [87] The evaluation dataset comprises the raw responses for the empirical assessment of *ERSmeta* and SMECS. Also, it includes the full questionnaire. The design of the evaluation, the applied methodology, and the

results are described in detail in Chapter 6. The complete dataset has been deposited on Zenodo [87].

Evaluation Scripts (Research Software) [72] The evaluation scripts constitute the data analytics software that was developed to process the material gathered for the evaluation of *ERSmeta* and SMECS. The entire code base is version-controlled on GitHub[31], guaranteeing that every change is recorded, and frozen releases have been archived on Zenodo [72]. The scripts were used to analyze the evaluation data and to generate all result figures presented in Chapter 6 which describes the evaluation, including its methodology and results. Also, the script to produce the figures in Section 5.4 (Figure 5.5) is included in the repository.

[31] https://github.com/Digitalized-Energy-Systems/ERSmeta-survey-analysis-scripts, last accessed 2026-06-08

B Detailed Comparison Between Different RSE Recommendations

This part of the appendix focuses on the recommendation for engineering ERS. As presented in Section 3.4, the created recommendations are compared with different sets of recommendations introduced in the literature (precisely with [11, 123, 133, 155, 168, 178, 208, 221]). In Section 3.4, Table 3.2 already shows a summary of this comparison. Within this part, Table B.1 and Table B.2 provide a detailed comparison, also providing the names of the recommendations in the other papers. Table B.3 shows recommendations in the other publications where no equivalent is available in the developed recommendations.

Table B.1: Detailed comparison between the developed recommendations for the energy domain and existing recommendations in other domains (Part 1).

The developed recommendations [77]	Ten simple rules for quick and dirty scientific programming [11]	Ten simple rules on writing clean and reliable open-source scientific software [123]	Four simple recommendations to encourage best practices in research software [133]	Ten Simple Rules for Developing Usable Software in Computational Biology [155]
1: Reuse and extend other software if possible and useful	3: Look for opportunities for code reuse			1: Identify the Missing Pieces
2: Use version control				
3: Develop open source and make your software findable			1: Make source code publicly accessible from day one, 2: Make software easy to discover by providing software metadata via a popular community registry, 3: Adopt a license and comply with the license of third-party dependencies	
4: Organize yourself and your team				
5: Consider the reuse of your research software				4: Use Standard Data Formats for Input and Output
6: Keep the architecture as simple as possible (but as complex as necessary)	1: Think before you code, 4: Modularize your code	3: Reduce complexity when possible		
7: You are not done until the documentation is done	8: Write self-documenting code for programmers and a readme file for users			
8: Test your software based on a test strategy				4: Use Standard Data Formats for Input and Output
9: Grow your community			4: Define clear and transparent contribution, governance and communication processes	
10: Include RSE in your research project proposals				

Table B.2: Detailed comparison between the developed recommendations for the energy domain and existing recommendations in other domains (Part 2).

The developed recommendations [77]	Ten Simple Rules for Effective Computational Research [168]	Ten Simple Rules for the Open Development of Scientific Software [178]	Ten simple rules for making research software more robust [208]	Best Practices for Scientific Computing [221]
1: Reuse and extend other software if possible and useful	1: Look Before You Leap	1: Don't Reinvent the Wheel	5: Reuse software (within reason)	4: Don't repeat yourself (or others)
2: Use version control	7: Version Control Everything		1: Use version control, 4: Version your releases	
3: Develop open source and make your software findable	9: Share Everything	4: Be Transparent		
4: Organize yourself and your team				
5: Consider the reuse of your research software	3: Make Your Code Understandable to Others (and Yourself)			
6: Keep the architecture as simple as possible (but as complex as necessary)		5: Be Simple		
7: You are not done until the documentation is done			2: Document your code and usage	7: Document design and purpose, not mechanics
8: Test your software based on a test strategy	3: Make Your Code Understandable to Others (and Yourself)			
9: Grow your community		7: Nurture and Grow Your Community, 8: Promote Your Project		
10: Include RSE in your research project proposals				

Table B.3: Overview of recommendations where no equivalent is included in the developed recommendations.

Ten simple rules for quick and dirty scientific programming [11]	Ten simple rules on writing clean and reliable open-source scientific software [123]	Ten Simple Rules for Developing Usable Software in Computational Biology [155]	Ten Simple Rules for Effective Computational Research [168]	Ten Simple Rules for the Open Development of Scientific Software [178]	Ten simple rules for making research software more robust [208]	Best Practices for Scientific Computing [221]
2: Start with prototypes and expand them in short development cycles	1: Choose a style guide—And stick with it	2: Collect Feedback from Prospective Users	2: Develop a Prototype First	2: Code Well	3: Make common operations easy to control	1: Write programs for people, not computers
5: Avoid premature optimization	2: Consider using an integrated development environment	3: Be Ready for Data Growth	4: Don't Underestimate the Complexity of Your Task	3: Be Your Own User	6: Rely on build tools and package managers for installation	2: Let the computer do the work
7: Refactor frequently	4: Refactor as needed	5: Expose Only Mandatory Parameters	5: Understand the Mathematical, Numerical, and Computational Methods Underpinning Your Work	6: Don't Be a Perfectionist	7: Do not require root or other special privileges to install or run	3: Make incremental changes
9: Grow your libraries and tools organically from your research	5: Program defensively	6: Expect Users to Make Mistakes		9: Find Sponsors	8: Eliminate hard-coded paths	5: Plan for mistakes
10: Go explore and be rigorous when you publish		7: Provide Logging Information	6: Use Pictures: They Really Are Worth a Thousand Words	10: Science Counts	9: Include a small test set that can be run to ensure the software is actually working	6: Optimize software only after it works correctly
		8: Get Users Started Quickly	10: Keep Going!			8: Collaborate
		9: Offer Tutorial Material				
		10: Consider the Future of Your Tool			10: Produce identical results when given identical inputs	

C Requirements for Metadata for Energy Research Software

This part of the appendix provides additional information on the results of the requirement analysis for the metadata schema as presented in Section 4.3 and published in [90]. Table C.1 lists the identified elements and their definitions for all parts of the domain model while figures of the different parts are shown in Section 4.3.

Table C.1: Definition of elements in the domain model.

	Element	Definition
Provenance	Categorization	Research category for which the software was written.
	Contributor	A person who has contributed to the software.
	Country of Origin	Country of the institution.
	Domain	Research domain for which the software was written.
	Geographical Spread	Geographical spread of the contributors.
	Institution	Institution which developed the software.
	Name	Name of the software.
	Number of Contributors	Number of contributors.
	Publication Describing the Software	An academic publication related to the software.
	Related Project	Name of the project during which the software was developed.
	Software Foundation	Software this software is based on.
Usage	Contact Mail	Individual responsible for maintaining the software.
	Date of First Version	The date on which the CreativeWork was initially created.

Continued on next page

Table C.1: Definition of elements in the domain model (continued).

	Element	Definition
Usage	Date of Last Version	The date on which the CreativeWork was most recently modified.
	Download Link	If the file can be downloaded, URL to download the binary.
	Example	Examples for using the software.
	Example/Link	Link to the example.
	Example/Type	Type of the example (e.g., minimal).
	Further Development Planned?	Is a further development planned?
	Interval of Updates	How often is the software updated?
	Link to Documentation	Link to documentation.
	Required Training Period	How long would a researcher need to be able to use the software?
	Support Given?	Will support be given?
Community	Existing Interactions	Ways how the software developers interact with the software users (community).
	Existing Interactions/Link	Link to further information on the community interaction.
	Existing Interactions/Type of Interaction	Type of the interaction.
	Git Downloads	Number of Git downloads.
	Number of Downloads	Number of downloads.
	Number of Git Forks	Number of forks on Git.
	Number of Git Issues	Number of Git issues.
	Number of Git Stars	Number of Git stars.
	Publications Using the Software	A publication that uses the software for its research.

Continued on next page

Table C.1: Definition of elements in the domain model (continued).

	Element	Definition
Interface	API	Description of the input and output functions (APIs) of the software.
	Compatible Research Data	Research data that is compatible with this software.
	Compatible Research Data/Automatic Download	Is compatible data automatically downloaded by the software?
	Compatible Research Hardware	Research hardware that is compatible with this software (e.g., in labs).
	Compatible Software	A software which is compatible with this software.
	Content (Semantic)	What type of data is usable as input?
	Content (Semantic)/ Resolution	Resolution of the variable.
	Direction (Input, Output, ..)	How does the connected element relate to the software?
	Required?	What data is required to get the software running?
	Syntax	What should the data look like?
	Standard Used	Standards for In/Output.
Software-specific	Compatible Programming Languages	Target programming environments to which the code applies. If applies to several versions, just the product name can be used.
	Dependencies	Required software dependencies.
	Dependencies/Availability	The availability of the dependency.
	Dependencies/License	The license of the dependency.
	Dependencies/Version Number	The version number of the dependency.
	Licenses/Costs	Typical cost for licensing the software.
	Licenses/Name	Name of the license.
	Licenses/Open vs. Not Open vs. Academic	A flag to signal that the software is accessible for free.

Continued on next page

Table C.1: Definition of elements in the domain model (continued).

	Element	Definition
Software-specific	Licenses/What is Allowed?	Details on the allowed reuse with these licenses.
	Minimal System Configuration	Minimal hardware requirements.
	Operating System	Operating systems supported (Windows 7, OSX 10.6, Android 1.6).
	Parallelization Capability	Does the software support parallelization?
	Programming Language	The computer programming language.
	Quality/Number of Tests	Number of tests.
	Quality/Open Review	A review of the software.
	Quality/Stars	Rating of the software in stars.
	Quality/Validation	Information on validations done for the software.
	Real-time Capability	Does the software support real-time simulations?
	Software Type (GUI vs. Commandline)	How can I interact with the software?
	Typical Execution Time	Typical execution time of the software.
	Typical Hardware	Typical hardware which is required to run the software.
Functionalities	Abstract	A description of the software.
	Assumptions	A required assumption of the software.
	Category	Research category for which the software was written.
	Component Model	A component of the energy system included in the software.
	Component Model/Level of Detail	Level of detail of the component.
	Component Model/Name	Name of the component.
	Feature	A feature of the software.
	Goals/Purpose	A general goal of the software.

Continued on next page

Table C.1: Definition of elements in the domain model (continued).

	Element	Definition
Functionalities	Graphical Abstract	Graphical overview of the functionality of the software.
	Method	An used method in the software.
	Optimization	Optimization used in the software.
	Optimization/Features	Features of the optimization.
	Optimization/Method	Method of the optimization.
	Optimization/Problem Type	Problem class of the optimization.
	Optimization/Solver	Used solver for the optimization.
	Parameters	Relevant parameters of the software.
	Primary Use Case	Primary use case of the software.
	Sector	A supported sector of the software.
	Technology	A supported technology of the software.
	Time-Space	Time and space required for using the software.
	Time-Space/Dimensions	Supported dimensions of the software.
	Time-Space/Geo Scope	Support geographical scope of the software.
	Time-Space/Possible Time Resolution	Supported time resolution of the software.
	Time-Space/Transient vs. Stationary	Does the software support stationary or transient effects?
	Time-Space/Voltage Level	Supported voltage level of the software.
	Use Case	A supported use case.
	Used Datasets	Is compatible data used within the software?

D Curriculum Vitæ

D.1 Education

09/2020 – 06/2026 **Doctor of Engineering – Computer Science**,
Carl von Ossietzky Universität Oldenburg, Germany, Finale grade: magna cum laude.

Supervisor: Prof. Astrid Nieße

Topic: Towards FAIR Energy Research Software

10/2016 – 11/2019 **Master of Science – Electrical Engineering and Information Technology**,
Leibniz Universität Hannover, Germany, Final grade: 1.0.

Thesis: Analysis of Economic Incentives for Decentralized Coordinated Voltage Regulation in the Distribution Grid

01/2017 – 05/2017 **Study Abroad – Electrical and Computer Engineering**,
Michigan State University, United States, GPA 4.0.

10/2013 – 11/2016 **Bachelor of Science – Electrical Engineering and Information Technology**,
Leibniz Universität Hannover, Germany, Final grade: 1.3.

Major: Communication Engineering

D.2 Work Experience

since 09/2023 **Senior Researcher**, *Standardized Systems Engineering and Assessment Group*, OFFIS e. V., Oldenburg, Germany.

- Coordination of the German National Research Data Infrastructure for the Inderdisciplinary Energy System Research (NFDI4Energy)

since 09/2020 **Researcher**, *Group for Digitalized Energy Systems (Prof. Astrid Nieße)*, Carl von Ossietzky Universität Oldenburg, Germany.

- Coordination of the German National Research Data Infrastructure for Inderdisciplinary Energy System Research (NFDI4Energy)
- Research on metadata for energy research software
- Research on flexibility for microgrids and the digitalization of energy research within the project future energy lab of the state Lower-Saxony
- Supervision of different seminars ("Introduction to Energy Informatics", "Smart Grid Research")
- Supervision of multiple Bachelor's and Master's thesis'

since 09/2020 **Visiting Researcher**, *Institute for Electric Power Systems, Electric Energy Storage Systems Group (Prof. Richard Hanke-Rauschenbach)*, Leibniz Universität Hannover, Germany.

09/2020 – 08/2023 *3 years* **Researcher**, *Co-Simulation of Multimodal Energy Systems Group*, OFFIS e. V., Oldenburg, Germany.

- Research within the project future energy lab of the state Lower-Saxony: Co-Simulation of a microgrid energy system
- Project Coordination (maternity leave replacement) of the future energy lab of the state Lower-Saxony

03/2020 – 08/2020 *6 months* **Researcher**, *Institute of Systems Engineering, Energy Informatics Group (Prof. Astrid Nieße)*, Leibniz Universität Hannover, Germany.

10/2019 – 02/2020
10/2018 – 03/2019 *11 months* **Student Teaching Assistant**, *Institute of Systems Engineering, Energy Informatics Group*, Leibniz Universität Hannover, Germany.

- Supported the professor in teaching about 20 students in the course "Agent-based control in energy systems"
- Creating the script for the course "Introduction to Energy Informatics"

01/2018 – 02/2019
11/2016 – 02/2017 **Student Research Assistant**, *Institut für Informationsverarbeitung*, Leibniz Universität Hannover, Germany.

09/2015 – 04/2016 *2 years* Software development within the project Low Bit Rate Region of Interest-based Video Coding for HDTV Aerial Surveillance Video

List of Figures

List of Tables

Bibliography

[1] *8. Energieforschungsprogramm Zur Angewandten Energieforschung – Forschungsmissionen Für Die Energiewende*. Tech. rep. Bundesministerium für Wirtschaft und Klimaschutz (BMWK), Oct. 2023. URL: https://www.bundeswirtschaftsministerium.de/Redaktion/DE/Publikationen/Energie/8-energieforschungsprogramm-zur-angewandten-energieforschung.html (last accessed 2026-03-04).

[2] Getaneh Alemu. *The Future of Enriched, Linked, Open and Filtered Metadata : Making Sense of IFLA, LRM, RDA, Linked Data and BIBFRAME*. London, 2022. ISBN: 978-1-78330-494-3. DOI: 10.29085/9781783304943.

[3] Pierre Alliez, Roberto Di Cosmo, Benjamin Guedj, Alain Girault, Mohand-Saïd Hacid, Arnaud Legrand, and Nicolas Rougier. "Attributing and Referencing (Research) Software: Best Practices and Outlook From Inria". In: *Computing in Science & Engineering* 22.1 (Jan. 2020), pp. 39–52. ISSN: 1558-366X. DOI: 10.1109/MCSE.2019.2949413.

[4] Renato Alves, Dimitrios Bampalikis, Leyla Jael Castro, José María Fernández González, Jennifer Harrow, Mateusz Kuzak, Eva Martin, Fotis E. Psomopoulos, and Allegra Via. *ELIXIR Software Management Plan for Life Sciences*. Preprint. Oct. 2021. DOI: 10.37044/osf.io/k8znb.

[5] *Amsterdam Declaration on Funding Research Software Sustainability*. Tech. rep. Research Software Alliance, Sept. 2024. DOI: 10.5281/zenodo.13735888.

[6] Felicity Anderson, Julien Sindt, and Neil Chue Hong. "Who Do You Think You Are? Creating RSE Personas From GitHub Interactions". In: *Electronic Communications of the EASST* 85 (Dec. 2025). ISSN: 1863-2122. DOI: 10.14279/eceasst.v85.2717.

[7] Grigoris Antoniou, Paul Groth, Frank van van Harmelen, and Rinke Hoekstra. *A Semantic Web Primer*. Cambridge, Londen: The MIT Press, 2012. ISBN: 978-0-262-01828-9. URL: https://dl.acm.org/doi/book/10.5555/2381011.

[8] Hartwig Anzt, Felix Bach, Stephan Druskat, Frank Löffler, Axel Loewe, Bernhard Y. Renard, Gunnar Seemann, Alexander Struck, Elke Achhammer, Piush Aggarwal, Franziska Appel, Michael Bader, Lutz Brusch, Christian Busse, Gerasimos Chourdakis, Piotr Wojciech Dabrowski, Peter Ebert, Bernd Flemisch, Sven Friedl, Bernadette Fritzsch, Maximilian D. Funk, Volker Gast, Florian Goth, Jean-Noël Grad, Jan Hegewald, Sibylle Hermann, Florian Hohmann, Stephan Janosch, Dominik Kutra, Jan Linxweiler, Thilo Muth, Wolfgang Peters-Kottig, Fabian Rack, Fabian H.C. Raters, Stephan Rave, Guido Reina, Malte Reißig, Timo Ropinski, Joerg Schaarschmidt, Heidi Seibold, Jan P. Thiele, Benjamin Uekermann, Stefan Unger, and Rudolf Weeber. "An Environment for Sustainable Research Software in Germany and beyond: Current State, Open Challenges, and Call for Action". In: *F1000Research* 9 (Jan. 2021), p. 295. ISSN: 2046-1402. DOI: 10.12688/f1000research.23224.2.

[9] Elvira-Maria Arvanitou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Jeffrey C. Carver. "Software Engineering Practices for Scientific Software Development: A Systematic Mapping Study". In: *Journal of Systems and Software* 172 (Feb. 2021), p. 110848. ISSN: 01641212. DOI: 10.1016/j.jss.2020.110848.

[10] Sören Auer, Allard Oelen, Muhammad Haris, Markus Stocker, Jennifer D'Souza, Kheir Eddine Farfar, Lars Vogt, Manuel Prinz, Vitalis Wiens, and Mohamad Yaser Jaradeh. "Improving Access to Scientific Literature with Knowledge Graphs". In: *Bibliothek Forschung und Praxis* 44.3 (Dec. 2020), pp. 516–529. ISSN: 1865-7648. DOI: 10.1515/bfp-2020-2042.

[11] Gabriel Balaban, Ivar Grytten, Knut Dagestad Rand, Lonneke Scheffer, and Geir Kjetil Sandve. "Ten Simple Rules for Quick and Dirty Scientific Programming". In: *PLOS Computational Biology* 17.3 (Mar. 2021), e1008549. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008549.

[12] Aaron Bangor, Philip Kortum, and James Miller. "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale". In: *Journal of Usability Studies* 4.3 (May 2009), pp. 114–123. URL: https://dl.acm.org/doi/10.5555/2835587.2835589 (last accessed 2026-03-04).

[13] Aaron Bangor, Philip T. Kortum, and James T. Miller. "An Empirical Evaluation of the System Usability Scale". In: *International Journal of Human–Computer Interaction* 24.6 (July 2008), pp. 574–594. ISSN: 1044-7318. DOI: 10.1080/10447310802205776.

[14] Matthias Bannert. *Research Software Engineering : A Guide to the Open Source Ecosystem*. First edition. Chapman & Hall/CRC Data Science Series. Boca Raton, 2024. ISBN: 978-1-032-26127-0.

[15] Michelle Barker. *FAIR4RS Roadmap Report*. Tech. rep. Feb. 2022. DOI: 10.5281/zenodo.6239373.

[16] Michelle Barker, Domhnall Carlin, Jeremy Cohen, Eric A. Jensen, Catherine M Jones, Carlos Martinez Ortiz, and Dan Rudmann. *Resources for Supporting Policy Change in Research Institutions in Practice: A Report from Subgroup 2 of the ReSA & RDA Policies in Research Organisations for Research Software (PRO4RS) Working Group*. Tech. rep. Zenodo, June 2024. DOI: 10.5281/zenodo.11529659.

[17] Michelle Barker, Leyla Jael Castro, Bernadette Fritzsch, Daniel S. Katz, Carlos Martinez-Ortiz, Anna Niehues, Alexander Struck, and Qian Zhang. *The FAIR for Research Software Principles after Two Years: An Adoption Update*. Tech. rep. Zenodo, Mar. 2024. DOI: 10.5281/zenodo.10816032.

[18] Michelle Barker, Neil P. Chue Hong, Daniel S. Katz, Anna-Lena Lamprecht, Carlos Martinez-Ortiz, Fotis Psomopoulos, Jennifer Harrow, Leyla Jael Castro, Morane Gruenpeter, Paula Andrea Martinez, and Tom Honeyman. "Introducing the FAIR Principles for Research Software". In: *Scientific Data* 9.1 (Oct. 2022), p. 622. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01710-x.

[19] Michelle Barker, Jeremy Cohen, Pedro Hernández Serrano, Daniel S. Katz, Kim Martin, Dan Rudmann, and Hugh Shanahan. *Institutional Policy Pathways for Supporting Research Software: Global Trends and Local Practices*. Preprint. Sept. 2025. DOI: 10.48550/arXiv.2509.26422.

[20] Nick Barnes. "Publish Your Computer Code: It Is Good Enough". In: *Nature* 467.7317 (Oct. 2010), pp. 753–753. ISSN: 1476-4687. DOI: 10.1038/467753a.

[21] Dominique Batista, Alejandra Gonzalez-Beltran, Susanna-Assunta Sansone, and Philippe Rocca-Serra. "Machine Actionable Metadata Models". In: *Scientific Data* 9.1 (Sept. 2022), p. 592. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01707-6.

[22] Robert Baxter, Neil Chue Hong, Dirk Gorissen, James Hetherington, and Ilian Todorov. "The Research Software Engineer". In: *Digital Research 2012*. Sept. 2012. URL: https://www.research.ed.ac.uk/en/publications/the-research-software-engineer (last accessed 2026-03-04).

[23] Frédérique Belliard, Angelica Maria Maineri, Esther Plomp, Andrés Felipe Ramos Padilla, Junzi Sun, and Maryam Zare Jeddi. "Ten Simple Rules for Starting FAIR Discussions in Your Community". In: *PLOS Computational Biology* 19.12 (Dec. 2023), e1011668. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1011668.

[24] Jan Bernoth, Bettina Buchholz, Christine Hennig, Alicia Janz, Ulrike Lucke, Evgenia Samoilova, and Sonja Schimmler. "From Research Data to Research Software Competencies: Results from a Workshop". In: *INFORMATIK 2025*. Gesellschaft für Informatik e.V., 2025, p. 1227. URL: https://dl.gi.de/handle/20.500.12116/47585 (last accessed 2026-03-04).

[25] Meisam Booshehri, Lukas Emele, Simon Flügel, Hannah Förster, Johannes Frey, Ulrich Frey, Martin Glauer, Janna Hastings, Christian Hofmann, Carsten Hoyer-Klick, Ludwig Hülk, Anna Kleinau, Kevin Knosala, Leander Kotzur, Patrick Kuckertz, Till Mossakowski, Christoph Muschner, Fabian Neuhaus, Michaja Pehl, Martin Robinius, Vera Sehn, and Mirjam Stappel. "Introducing the Open Energy Ontology: Enhancing Data Interpretation and Interfacing in Energy Systems Analysis". In: *Energy and AI* 5 (Sept. 2021), p. 100074. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2021.100074.

[26] Paul Brack, Peter Crowther, Stian Soiland-Reyes, Stuart Owen, Douglas Lowe, Alan R. Williams, Quentin Groom, Mathias Dillen, Frederik Coppens, Björn Grüning, Ignacio Eguinoa, Philip Ewels, and Carole Goble. "Ten Simple Rules for Making a Software Tool Workflow-Ready". In: *PLOS Computational Biology* 18.3 (Mar. 2022), e1009823. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1009823.

[27] Jonathan Brandt, Emilie Frost, Stephan Ferenz, Paul Hendrik Tiemann, Astrid Bensmann, Richard Hanke-Rauschenbach, and Astrid Nieße. "Choosing the Right Model for Unified Flexibility Modeling". In: *Energy Informatics* 5.1 (July 2022), p. 10. ISSN: 2520-8942. DOI: 10.1186/s42162-022-00192-w.

[28] John Brooke. "SUS: A Quick and Dirty Usability Scale". In: *Usability Eval. Ind.* 1st Edition. Vol. 189. CRC Press, Nov. 1995. ISBN: 978-0-429-15701-1. DOI: 10.1201/9781498710411-35.

[29] *Bund-Länder-Vereinbarung Zu Aufbau Und Förderung Einer Nationalen Forschungsdateninfrastruktur (NFDI) Vom 26. November 2018*. Tech. rep. Gemeinsame Wissenschaftskonferenz (GWK). URL: https://www.gwk-bonn.de/fileadmin/Redaktion/Dokumente/Papers/NFDI.pdf (last accessed 2026-03-04).

[30] Nancy Carter, Denise Bryant-Lukosius, Alba DiCenso, Jennifer Blythe, and Alan J. Neville. "The Use of Triangulation in Qualitative Research". In: *Oncology Nursing Forum* 41.5 (Sept. 2014), pp. 545–547. ISSN: 0190-535X, 1538-0688. DOI: 10.1188/14.ONF.545-547.

[31] Jeffrey C. Carver, Nic Weber, Karthik Ram, Sandra Gesing, and Daniel S. Katz. "A Survey of the State of the Practice for Research Software in the United States". In: *PeerJ Computer Science* 8 (May 2022), e963. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.963.

[32] Wolfgang zu Castell, Doris Dransch, Guido Juckeland, Marcel Meistring, Bernadette Fritzsch, Ronny Gey, Britta Höpfner, Martin Köhler, Christian Meeßen, Hela Mehrtens, Felix Mühlbauer, Sirko Schindler, Thomas Schnicke, and Roland Bertelmann. *Towards a Quality Indicator for Research Data Publications and Research Software Publications – A Vision from the Helmholtz Association*. Preprint. Jan. 2024. DOI: 10.48550/arXiv.2401.08804.

[33] João Aguiar Castro, Ricardo Carvalho Amorim, Rúbia Gattelli, Yulia Karimova, João Rocha da Silva, and Cristina Ribeiro. "Involving Data Creators in an Ontology-Based Design Process for Metadata Models". In: *Developing Metadata Application Profiles*. Ed. by Mariana Curado Malta, Ana Alice Baptista, and Paul Walk. Hershey, PA, USA: IGI Global, 2017, pp. 181–214. ISBN: 978-1-5225-2221-8. DOI: 10.4018/978-1-5225-2221-8.ch008.

[34] Leyla Jael Castro, Stephan Ferenz, Marc Fuhrmans, Jan Göpfert, Dorothea Iglezakis, Oliver Karras, and Alexander Struck. *"Research Software Metadata" - Working Group Charter (NFDI Section-Metadata)*. Tech. rep. Zenodo, Dec. 2023. DOI: 10.5281/zenodo.10246218.

[35] Leyla Jael Castro, Brigitte Mathiak, and Astrid Nieße. *Connected Open Source Software - ConnOSS - Proposal*. Tech. rep. Zenodo, June 2025. DOI: 10.5281/zenodo.15616384.

[36] German Aerospace Center. *DLR Software Engineering Initiative*. Mar. 2024. URL: https://rse.dlr.de/ (last accessed 2026-03-04).

[37] Anthony J. Christe, Sergey I. Negrashov, Philip M. Johnson, Dylan Nakahodo, David Badke, and David Aghalarpour. "OPQ Version 2: An Architecture for Distributed, Real-Time, High Performance Power Data Acquisition, Analysis, and Visualization". In: *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. July 2017, pp. 1415–1419. DOI: 10.1109/CYBER.2017.8446111.

[38] Neil Chue Hong, Elena Breitmoser, Mario Antonioletti, Joy Davidson, Daniel Garijo, Alejandra Gonzalez-Beltran, Morane Gruenpeter, Robert Huber, Clement Jonquet, Mike Priddy, John Shepeherdson, Maaike Verburg, and Chris Wood. *D5.2 - Metrics for Automated FAIR Software Assessment in a Disciplinary Context*. Tech. rep. May 2025. DOI: 10.5281/zenodo.15535629.

[39] Neil P Chue Hong, Daniel S Katz, Michelle Barker, Anna-Lena Lamprecht, Carlos Martinez, Fotis E Psomopoulos, Jen Harrow, Leyla Jael Castro, Morane Gruenpeter, Paula Andrea Martinez, Tom Honeyman, Alexander Struck, Allen Lee, Axel Loewe, Ben van Werkhoven, Catherine Jones, Daniel Garijo, Esther Plomp, Francoise Genova, Hugh Shanahan, Joanna Leng, Maggie Hellström, Manodeep Sinha, Mateusz Kuzak, Patricia Herterich, Qian Zhang, Sharif Islam, Susanna-Assunta Sansone, Tom Pollard, Udayanto Dwi Atmojo, Alan Williams, Andreas Czerniak, Anna Niehues, Anne Claire Fouilloux, Bala Desinghu, Céline Richard, Charles Gray, Chris Erdmann, Daniel Nüst, Daniele Tartarini, Hartwig Anzt, Ilian Todorov, James McNally, Javier Moldon, Jessica Burnett, Khalid Belhajjame, Laurents Sesink, Lorraine Hwang, Marcos Roberto, Mark D Wilkinson, Mathieu Servillat, Matthias Liffers, Merc Fox, Nick Lynch, Paula Martinez Lavanchy, Sandra Gesing, Sarah Stevens, Martinez Cuesta, Silvio Peroni, Stian Soiland-Reyes, Tom Bakker, Tovo Rabemanantsoa, Vanessa Sochat, and Yo Yehudi. *FAIR Principles for Research Software (FAIR4RS Principles)*. Tech. rep. FAIR4RS WG, May 2022, p. 32. DOI: 10.15497/RDA00068.

[40] Jeremy Cohen, Daniel S. Katz, Michelle Barker, Neil Chue Hong, Robert Haines, and Caroline Jay. "The Four Pillars of Research Software Engineering". In: *IEEE Software* 38.1 (Jan. 2021), pp. 97–105. ISSN: 1937-4194. DOI: 10.1109/MS.2020.2973362.

[41] Julien Corman, Juan L. Reutter, and Ognjen Savković. "Semantics and Validation of Recursive SHACL". In: *The Semantic Web – ISWC 2018*. Ed. by Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl. Cham: Springer International Publishing, 2018, pp. 318–336. ISBN: 978-3-030-00671-6. DOI: 10.1007/978-3-030-00671-6_19.

[42] Ian A. Cosden, Kenton McHenry, and Daniel S. Katz. "Research Software Engineers: Career Entry Points and Training Gaps". In: *Computing in Science & Engineering* 24.6 (Nov. 2022), pp. 14–21. ISSN: 1558-366X. DOI: 10.1109/MCSE.2023.3258630.

[43] Guy Courbebaisse, Bernd Flemisch, Kay Graf, Uwe Konrad, Jason Maassen, and Raphael Ritz. *Research Software Lifecycle*. Tech. rep. Sept. 2023. DOI: 10.5281/zenodo.8324828.

[44] Stephen Crouch, Neil Chue Hong, Simon Hettrick, Mike Jackson, Aleksandra Pawlik, Shoaib Sufi, Les Carr, David De Roure, Carole Goble, and Mark Parsons. "The Software Sustainability Institute: Changing Research Software Attitudes and Practices". In: *Computing in Science & Engineering* 15.6 (Nov. 2013), pp. 74–80. ISSN: 1521-9615. DOI: 10.1109/MCSE.2013.133.

[45] Mariana Curado Malta and Ana Alice Baptista. "A Method for the Development of Dublin Core Application Profiles (Me4DCAP V0.2): Detailed Description". In: *Proceedings of the 2013 International Conference on Dublin Core and Metadata Applications*. Lisbon, Portugal: Dublin Core Metadata Initiative, 2013, pp. 90–103. URL: https://dcpapers.dublincore.org/pubs/article/viewFile/3674/1897.pdf (last accessed 2026-03-04).

[46] Mariana Curado Malta and Ana Alice Baptista. *Me4MAP V1.0: A Method for the Development of Metadata Application Profiles*. Unpublished Preprint.

[47] Mariana Curado Malta and Ana Alice Baptista. "The Development Process of a Metadata Application Profile for the Social and Solidarity Economy". In: *Developing Metadata Application Profiles*. Ed. by Mariana Curado Malta, Ana Alice Baptista, and Paul Walk. Hershey, PA, USA: IGI Global, 2017, pp. 98–117. ISBN: 978-1-5225-2221-8. DOI: 10.4018/978-1-5225-2221-8.ch005.

[48] Andreas Czerniak, Adrian Ehrenhofer, Bernadette Fritzsch, Maximilian Funk, Florian Goth, Reiner Hähnle, Carina Haupt, Marco Konersmann, Jan Linxweiler, Frank Löffler, Alexander Lüpges, Sebastian Nielebock, Bernhard Rumpe, Ina Schieferdecker, Tobias Schlauch, Robert Speck, Alexander Struck, Jan Philipp Thiele, Matthias Tichy, and Inga Ulusoy. *GI- und DE-RSE Muster-Leitlinie zur effizienten Entwicklung von Forschungssoftware*. Tech. rep. 2025. DOI: 10.18420/2025-gi_de-rse.

[49] Mario David, Miguel Colom, Daniel Garijo, Leyla Jael Castro, Violaine Louvet, Elisabetta Ronchieri, Massimo Torquati, Laura del Caño, Siew Hoon Leong, Maxime van den Bossche, Isabel Campos, and Roberto Di Cosmo. *Task Force Sub Group 3 - Review of Software Quality Attributes and Characteristics*. Tech. rep. Zenodo, Aug. 2023. DOI: 10.5281/zenodo.8221384.

[50] Fred D. Davis. "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology". In: *MIS Quarterly* 13.3 (1989), pp. 319–340. ISSN: 0276-7783. DOI: 10.2307/249008.

[51] Joseph DeCarolis, Hannah Daly, Paul Dodds, Ilkka Keppo, Francis Li, Will McDowall, Steve Pye, Neil Strachan, Evelina Trutnevyte, Will Usher, Matthew Winning, Sonia Yeh, and Marianne Zeyringer. "Formalizing Best Practice for Energy System Optimization Modelling". In: *Applied Energy* 194 (May 2017), pp. 184–198. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2017.03.001.

[52] Deekshitha, Rena Bakhshi, Jason Maassen, Carlos Martinez Ortiz, Rob van Nieuwpoort, and Slinger Jansen. *RSMM: A Framework to Assess Maturity of Research Software Project*. Preprint. June 2024. DOI: 10.48550/arXiv.2406.01788.

[53] Deekshitha, Siamak Farshidi, Jason Maassen, Rena Bakhshi, Rob Van Nieuwpoort, and Slinger Jansen. "FAIRSECO: An Extensible Framework for Impact Measurement of Research Software". In: *2023 IEEE 19th International Conference on E-Science (e-Science)*. Limassol, Cyprus: IEEE, Oct. 2023, pp. 1–10. ISBN: 979-8-3503-2223-1. DOI: 10.1109/e-Science58273.2023.10254664.

[54] Julian Dehne, Simon Christ, Florian Goth, Jean-Noël Grad, Magnus Hagdorn, Jan Philipp Thiele, Harald von Waldow, and Jan Linxweiler. "Research Software Engineering for Natural Sciences". In: Softwaretechnik-Trends 45.3 (2025). ISSN: 0720-8928. URL: https://dl.gi.de/handle/20.500.12116/47200 (last accessed 2026-03-04).

[55] Roberto Di Cosmo, Sabrina Granger, Konrad Hinsen, Nicolas Jullien, Daniel Le Berre, Violaine Louvet, Camille Maumet, Clémentine Maurice, Raphaël Monat, and Nicolas P. Rougier. "Stop Treating Code like an Afterthought: Record, Share and Value It". In: *Nature* 646.8084 (Oct. 2025), pp. 284–286. ISSN: 1476-4687. DOI: 10.1038/d41586-025-03196-0.

[56] Directorate-General for Research and Innovation (European Commission). *Turning FAIR into Reality: Final Report and Action Plan from the European Commission Expert Group on FAIR Data*. Tech. rep. Publications Office of the European Union, 2018. DOI: 10.2777/1524.

[57] Stephan Druskat, Oliver Bertuch, Guido Juckeland, Oliver Knodel, and Tobias Schlauch. *Software Publications with Rich Metadata: State of the Art, Automated Workflows and HERMES Concept*. Preprint. Jan. 2022. DOI: 10.48550/arXiv.2201.09015.

[58] Stephan Druskat, Nasir U. Eisty, Robert Chisholm, Neil P. Chue Hong, Ryan C. Cocking, Myra B. Cohen, Michael Felderer, Lars Grunske, Sarah A. Harris, Wilhelm Hasselbring, Thomas Krause, Jan Linxweiler, and Colin C. Venters. "Better Archi-

tecture, Better Software, Better Research". In: *Computing in Science & Engineering* 27.2 (Apr. 2025), pp. 45–57. ISSN: 1558-366X. DOI: 10.1109/MCSE.2025.3573887.

[59] Stephan Druskat, Michael Felderer, and Carina Haupt. "Software Engineering, Research Software and Requirements Engineering". In: *Softwaretechnik-Trends Band 44, Heft 1*. Gesellschaft für Informatik e.V., 2024. URL: https://dl.gi.de/handle/20.500.12116/44043 (last accessed 2026-03-04).

[60] Stephan Druskat, Jurriaan H. Spaaks, Neil Chue Hong, Robert Haines, James Baker, Spencer Bliven, Egon Willighagen, David Pérez-Suárez, and Alexander Konovalov. *Citation File Format*. Tech. rep. Zenodo, Aug. 2021. DOI: 10.5281/zenodo.5171937.

[61] David W. Eccles and Güler Arsal. "The Think Aloud Method: What Is It and How Do I Use It?" In: *Qualitative Research in Sport, Exercise and Health* 9.4 (Aug. 2017), pp. 514–531. ISSN: 2159-676X. DOI: 10.1080/2159676X.2017.1331501.

[62] Nasir U. Eisty and Jeffrey C. Carver. "Developers Perception of Peer Code Review in Research Software Development". In: *Empirical Software Engineering* 27.1 (Oct. 2021), p. 13. ISSN: 1573-7616. DOI: 10.1007/s10664-021-10053-x.

[63] Nasir U. Eisty and Jeffrey C. Carver. "Testing Research Software: A Survey". In: *Empirical Software Engineering* 27.6 (July 2022), p. 138. ISSN: 1573-7616. DOI: 10.1007/s10664-022-10184-9.

[64] Nasir U. Eisty, Jeffrey C. Carver, Johanna Cohoon, Ian A. Cosden, Carole Goble, and Samuel Grayson. "Ten Simple Rules for Catalyzing Collaborations and Building Bridges Between Research Software Engineers and Software Engineering Researchers". In: *Computing in Science & Engineering* 27.2 (Apr. 2025), pp. 10–17. ISSN: 1558-366X. DOI: 10.1109/MCSE.2025.3569786.

[65] Nasir U. Eisty, George K. Thiruvathukal, and Jeffrey C. Carver. "A Survey of Software Metric Use in Research Software Development". In: *2018 IEEE 14th International Conference on E-Science (e-Science)*. Oct. 2018, pp. 212–222. DOI: 10.1109/eScience.2018.00036.

[66] Nasir U. Eisty, George K. Thiruvathukal, and Jeffrey C. Carver. "Use of Software Process in Research Software Development: A Survey". In: *Proceedings of the 23rd International Conference on Evaluation and Assessment in Software Engineering*. EASE '19. New York, NY, USA: Association for Computing Machinery, Apr. 2019, pp. 276–282. ISBN: 978-1-4503-7145-2. DOI: 10.1145/3319008.3319351.

[67] Kolbjørn Engeland, Marco Borga, Jean-Dominique Creutin, Baptiste François, Maria-Helena Ramos, and Jean-Philippe Vidal. "Space-Time Variability of Climate Variables and Intermittent Renewable Electricity Production – A Review". In: *Renewable and Sustainable Energy Reviews* 79 (Nov. 2017), pp. 600–617. ISSN: 1364-0321. DOI: 10.1016/j.rser.2017.05.046.

[68] Muhammad Shoaib Farooq, Maimoona Salam, Norizan Jaafar, Alain Fayolle, Kartinah Ayupp, Mirjana Radovic-Markovic, and Ali Sajid. "Acceptance and Use of Lecture Capture System (LCS) in Executive Business Studies: Extending UTAUT2". In: *Interactive Technology and Smart Education* 14.4 (Nov. 2017), pp. 329–348. ISSN: 1741-5659. DOI: 10.1108/ITSE-06-2016-0015.

[69] Siamak Farshidi, Kwabena Bennin, Önder Babur, June Sallou, Ayalew Kassahun, and Bedir Tekinerdogan. *Advancing Research Software Engineering with AI: A Research Framework*. Preprint. Aug. 2025. DOI: 10.21203/rs.3.rs-7178452/v1.

[70] Michael Felderer, Michael Goedicke, Lars Grunske, Wilhelm Hasselbring, Anna-Lena Lamprecht, and Bernhard Rumpe. "Investigating Research Software Engineering: Toward RSE Research". In: *Communications of the ACM* 68.2 (Jan. 2025), pp. 20–23. ISSN: 0001-0782. DOI: 10.1145/3685265.

[71] Stephan Ferenz. *ERSmeta*. Metadata Schema. Oct. 2025. DOI: 10.5281/zenodo.17465772.

[72] Stephan Ferenz. *ERSmeta Evaluation Scripts*. Software. Mar. 2026. DOI: 10.5281/zenodo.18983893.

[73] Stephan Ferenz. *Interview Guide for the Requirements Analysis for an Application Profile/Metadata Schema for Energy Research Software*. Tech. rep. May 2024. DOI: 10.5281/zenodo.11189410.

[74] Stephan Ferenz. "Towards More Findable Energy Research Software by Introducing a Metadata-based Registry". In: *Abstracts of the 11th DACH+ Conference on Energy Informatics*. Ed. by Anke Weidlich, Gunther Gust and Mirko Schäfer. Springer, 2022. DOI: 10.1186/s42162-022-00215-6.

[75] Stephan Ferenz. *Towards More Findable Energy Research Software by Introducing a Metadata-based Registry*. Poster. Oct. 2022. DOI: 10.5281/zenodo.7243593.

[76] Stephan Ferenz, Leyla Jael Castro, Hamideh Hajiabadi, Matthias Löbe, Ulrik Stervbo, Lu Gan, and Florian Thiery. "Towards Common Metadata For Research Software in the NFDI". In: *2nd Conference on Research Data Infrastructure (CoRDI)*. Zenodo, Aug. 2025. DOI: 10.5281/zenodo.16735848.

[77] Stephan Ferenz, Emilie Frost, Rico Schrage, Thomas Wolgast, Inga Beyers, Oliver Karras, Oliver Werth, and Astrid Nieße. "Ten Recommendations for Engineering Research Software in Energy Research". In: *Proceedings of the 16th ACM International Conference on Future and Sustainable Energy Systems*. E-Energy '25. New York, NY, USA: Association for Computing Machinery, June 2025, pp. 446–459. ISBN: 979-8-4007-1125-1. DOI: 10.1145/3679240.3734606.

[78] Stephan Ferenz, Emilie Frost, Rico Schrage, Thomas Wolgast, Oliver Werth, Inga Beyers, and Astrid Nieße. *Poster: Towards Guidelines for Engineering of Energy Research Software*. Poster. Karlsruhe, Germany, Mar. 2025. DOI: 10.5281/zenodo.15357363.

[79] Stephan Ferenz, Emilie Frost, Rico Schrage, Thomas Wolgast, Oliver Werth, and Astrid Nieße. *Towards Guidelines for Engineering of Energy Research Software*. Presentation. Mar. 2025. DOI: 10.5281/zenodo.14968117.

[80] Stephan Ferenz and Aida Jafarbigloo. *Software Metadata Extraction and Curation Software (SMECS)*. Software. Feb. 2026. DOI: 10.5281/zenodo.18618460.

[81] Stephan Ferenz, Aida Jafarbigloo, Oliver Werth, and Astrid Nieße. *SMECS: A Software Metadata Extraction and Curation Software*. Presentation. Mar. 2025. DOI: 10.5281/zenodo.14968049.

[82] Stephan Ferenz, Aida Jafarbigloo, Oliver Werth, and Astrid Nieße. "SMECS: A Software Metadata Extraction and Curation Software". In: *Electronic Communications of the EASST* 85 (Dec. 2025). ISSN: 1863-2122. DOI: 10.14279/eceasst.v85.2708.

[83] Stephan Ferenz and Astrid Nieße. *Improved Findability of Energy Research Software by Introducing a Metadata-based Registry*. Presentation. Jan. 2024. DOI: 10.5281/zenodo.10854452.

[84] Stephan Ferenz and Astrid Nieße. "Towards Improved Findability of Energy Research Software by Introducing a Metadata-based Registry". In: *ing.grid* 1.2 (Nov. 2023). ISSN: 2941-1300. DOI: 10.48694/inggrid.3837.

[85] Stephan Ferenz and Astrid Nieße. *Towards Services to Enable FAIR Research Software in a Typical Research Project Cycle*. Presentation. Mar. 2025. DOI: 10.5281/zenodo.14968144.

[86] Stephan Ferenz, Annika Ofenloch, Fernando Penaherrera Vaca, Henrik Wagner, Oliver Werth, Michael H. Breitner, Bernd Engel, Sebastian Lehnhoff, and Astrid Nieße. "An Open Digital Platform to Support Interdisciplinary Energy Research

and Practice—Conceptualization". In: *Energies* 15.17 (Jan. 2022), p. 6417. ISSN: 1996-1073. DOI: 10.3390/en15176417.

[87] Stephan Ferenz and Oliver Werth. *Dataset of the Evaluation of a Metadata Schema for Energy Research Software*. Jan. 2026. DOI: 10.5281/zenodo.17935022.

[88] Stephan Ferenz and Oliver Werth. *Questionnaire for Evaluating a Metadata Schema for Energy Research Software*. Tech. rep. Zenodo, Oct. 2025. DOI: 10.5281/zenodo.17465939.

[89] Stephan Ferenz, Oliver Werth, and Astrid Nieße. "Enabling FAIR Energy Research Software". In: *Open Conference Proceedings*. Vol. 9. Mar. 2026. DOI: 10.52825/ocp.v9i.3279.

[90] Stephan Ferenz, Oliver Werth, and Astrid Nieße. "Requirements for a Metadata Scheme to Enable FAIR Energy Research Software". In: *Electronic Communications of the EASST* 83 (Feb. 2025). ISSN: 1863-2122. DOI: 10.14279/eceasst.v83.2594.

[91] Stephan Ferenz, Oliver Werth, and Astrid Nieße. *Towards a Metadata Schema for Energy Research Software*. Preprint. Jan. 2026. DOI: 10.48550/arXiv.2601.09456.

[92] Stephan Ferenz, Oliver Werth, and Astrid Nieße. *Towards a Metadata Scheme for Energy Research Software*. Presentation. Mar. 2025. DOI: 10.5281/zenodo.15101103.

[93] Daniel Garijo. *Auto CodeMeta Generator*. Ontology Engineering Group (UPM). June 2025. URL: https://github.com/oeg-upm/auto-codemeta-generator (last accessed 2026-03-04).

[94] Daniel Garijo, Allen Mao, Haripriya Dharmala, Cedant Diwanji, Jiajing Wang, Aidan Kelley, Miguel Angel García, Jenifer Ciucio-Kiss, and Juanje Mendoza. *A Framework for Creating Knowledge Graphs of Scientific Software Metadata*. 2025. DOI: 10.1162/qss_a_00167.

[95] Daniel Garijo, Hervé Ménager, Lorraine Hwang, Ana Trisovic, Michael Hucka, Thomas Morrell, and Alice Allen. "Nine Best Practices for Research Software Registries and Repositories". In: *PeerJ Computer Science* 8 (Aug. 2022), e1023. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.1023.

[96] Daniel Garijo, Maximiliano Osorio, Deborah Khider, Varun Ratnakar, and Yolanda Gil. "OKG-Soft: An Open Knowledge Graph with Machine Readable Scientific Software Metadata". In: *2019 15th International Conference on eScience (eScience)*. Sept. 2019, pp. 349–358. DOI: 10.1109/eScience.2019.00046.

[97] Yolanda Gil, Daniel Garijo, Saurabh Mishra, and Varun Ratnakar. "OntoSoft: A Distributed Semantic Registry for Scientific Software". In: *2016 IEEE 12th International Conference on E-Science (e-Science)*. Oct. 2016, pp. 331–336. DOI: 10.1109/eScience.2016.7870916.

[98] Yolanda Gil, Varun Ratnakar, and Daniel Garijo. "OntoSoft: Capturing Scientific Software Metadata". In: *Proceedings of the 8th International Conference on Knowledge Capture*. K-CAP 2015. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1–4. ISBN: 978-1-4503-3849-3. DOI: 10.1145/2815833.2816955.

[99] Florian Goth, Renato Alves, Matthias Braun, Leyla Jael Castro, Gerasimos Chourdakis, Simon Christ, Jeremy Cohen, Stephan Druskat, Fredo Erxleben, Jean-Noël Grad, Magnus Hagdorn, Toby Hodges, Guido Juckeland, Dominic Kempf, Anna-Lena Lamprecht, Jan Linxweiler, Frank Löffler, Michele Martone, Moritz Schwarzmeier, Heidi Seibold, Jan Philipp Thiele, Harald von Waldow, and Samantha Wittke. *Foundational Competencies and Responsibilities of a Research Software Engineer: Current State and Suggestions for Future Directions*. Preprint. Sept. 2025. DOI: 10.12688/f1000research.157778.2.

[100] Morane Gruenpeter, Roberto Di Cosmo, Hylke Koers, Patricia Herterich, Rob Hooft, Jessica Parland-von Essen, Jonas Tana, Tero Aalto, and Sarah Jones. *M2.15 Assessment Report on 'FAIRness of Software'*. Tech. rep. Oct. 2020. DOI: 10.5281/zenodo.4095092.

[101] Morane Gruenpeter, Sabrina Granger, Alain Monteil, Neil Chue Hong, Elena Breitmoser, Mario Antonioletti, Daniel Garijo, Esteban González Guardia, Alejandra Gonzalez Beltran, Carole Goble, Stian Soiland-Reyes, Nick Juty, and Gabriela Mejias. *D4.4 - Guidelines for Recommended Metadata Standard for Research Software within EOSC*. Tech. rep. Mar. 2024. DOI: 10.5281/zenodo.10786147.

[102] Morane Gruenpeter, Daniel S. Katz, Anna-Lena Lamprecht, Tom Honeyman, Daniel Garijo, Alexander Struck, Anna Niehues, Paula Andrea Martinez, Leyla Jael Castro, Tovo Rabemanantsoa, Neil P. Chue Hong, Carlos Martinez-Ortiz, Laurents Sesink, Matthias Liffers, Anne Claire Fouilloux, Chris Erdmann, Silvio Peroni, Paula Martinez Lavanchy, Ilian Todorov, and Manodeep Sinha. *Defining Research Software:*

A Controversial Discussion. Tech. rep. Zenodo, Sept. 2021. DOI: 10.5281/zenodo.5504016.

[103] Lars Grunske, Anna-Lena Lamprecht, Wilhelm Hasselbring, and Bernhard Rumpe. "Research Software Engineering". In: *Forschung & Lehre* 3 (Mar. 2024), p. 12. ISSN: 0945-5604. DOI: 10.37307/j.0945-5604.2024.03.12.

[104] Linda Guarascio-Howard and Robert Gray. "Using Kinematics to Assess Software Usability and Ergonomic Risk Factors". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 48.10 (Sept. 2004), pp. 1184–1188. ISSN: 1071-1813. DOI: 10.1177/154193120404801008.

[105] *Guidelines for Safeguarding Good Research Practice. Code of Conduct*. Tech. rep. Deutsche Forschungsgemeinschaft (DFG), Jan. 2025. URL: https://zenodo.org/records/14281892.

[106] Martin Hammitzsch, Christian Busse, Bernd Flemisch, Konrad Förstner, Tibor Kalman, Frank Löffler, Marius Politze, and Dirk Riehle. *Concept for Setting up a Working Group Research Software Engineering in the NFDI Section Common Infrastructures*. Tech. rep. Apr. 2022. DOI: 10.5281/zenodo.6483449.

[107] Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. "How Do Scientists Develop and Use Scientific Software?" In: *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*. Vancouver, BC, Canada: IEEE, May 2009, pp. 1–8. ISBN: 978-1-4244-3737-5. DOI: 10.1109/SECSE.2009.5069155.

[108] Claire M. Hart, Timothy D. Ritchie, Erica G. Hepper, and Jochen E. Gebauer. "The Balanced Inventory of Desirable Responding Short Form (BIDR-16)". In: *Sage Open* 5.4 (Oct. 2015), p. 2158244015621113. ISSN: 2158-2440. DOI: 10.1177/2158244015621113.

[109] Wilhelm Hasselbring, Leslie Carr, Simon Hettrick, Heather Packer, and Thanassis Tiropanis. "From FAIR Research Data toward FAIR and Open Research Software". In: *it - Information Technology* 62.1 (Feb. 2020), pp. 39–47. ISSN: 1611-2776, 2196-7032. DOI: 10.1515/itit-2019-0040.

[110] Wilhelm Hasselbring, Leslie Carr, Simon Hettrick, Heather Packer, and Thanassis Tiropanis. "Open Source Research Software". In: *Computer* 53.8 (Aug. 2020), pp. 84–88. ISSN: 1558-0814. DOI: 10.1109/MC.2020.2998235.

[111] Wilhelm Hasselbring, Stephan Druskat, Jan Bernoth, Philine Betker, Michael Felderer, Stephan Ferenz, Ben Hermann, Anna-Lena Lamprecht, Jan Linxweiler, Arnau Prat, Bernhard Rumpe, Katrin Schöning-Stierand, and Shinhyung Yang. "Multidimensional Research Software Categorization". In: *Computing in Science & Engineering* 27.2 (Apr. 2025), pp. 59–68. ISSN: 1558-366X. DOI: 10.1109/MCSE.2025.3555023.

[112] Wilhelm Hasselbring, Daniel S. Katz, and Rob van Nieuwpoort. "Technology Research Software: An Often Overlooked Category of Research Software". In: *Computing in Science & Engineering* 28.1 (Jan. 2026), pp. 94–99. ISSN: 1558-366X. DOI: 10.1109/MCSE.2026.3662117.

[113] David Haynes. *Metadata for Information Management and Retrieval : Understanding Metadata and Its Use*. Second edition. London, 2018. ISBN: 978-1-85604-824-8. DOI: 10.1080/23257962.2019.1664426.

[114] Sibylle Hermann and Jörg Fehr. "Documenting Research Software in Engineering Science". In: *Scientific Reports* 12.1 (Apr. 2022), p. 6567. ISSN: 2045-2322. DOI: 10.1038/s41598-022-10376-9.

[115] Sibylle Hermann, Dorothea Iglezakis, and Anett Seeland. "Requirements for Finding Research Data and Software". In: *PAMM* 19.1 (2019), e201900480. ISSN: 1617-7061. DOI: 10.1002/pamm.201900480.

[116] Michael A. Heroux. "Research Software Science: Expanding the Impact of Research Software Engineering". In: *Computing in Science & Engineering* 24.6 (Nov. 2022), pp. 22–27. ISSN: 1558-366X. DOI: 10.1109/MCSE.2023.3260475.

[117] Alexandre Hocquet, Frédéric Wieber, Gabriele Gramelsberger, Konrad Hinsen, Markus Diesmann, Fernando Pasquini Santos, Catharina Landström, Benjamin Peters, Dawid Kasprowicz, Arianna Borrelli, Phillip Roth, Clarissa Ai Ling Lee, Alin Olteanu, and Stefan Böschen. "Software in Science Is Ubiquitous yet Overlooked". In: *Nature Computational Science* 4.7 (July 2024), pp. 465–468. ISSN: 2662-8457. DOI: 10.1038/s43588-024-00651-2.

[118] Maximilian Hoffmann, Bruno U. Schyska, Julian Bartels, Tristan Pelser, Johannes Behrens, Manuel Wetzel, Hans Christian Gils, Chuen-Fung Tang, Marius Tillmanns, Jan Stock, André Xhonneux, Leander Kotzur, Aaron Praktiknjo, Thomas Vogt, Patrick Jochem, Jochen Linßen, Jann M. Weinand, and Detlef Stolten. "A Review of Mixed-Integer Linear Formulations for Framework-Based Energy System Models". In: *Advances in Applied Energy* 16 (Dec. 2024), p. 100190. ISSN: 2666-7924. DOI: 10.1016/j.adapen.2024.100190.

[119] Neil P. Chue Hong, Selina Aragon, Simon Hettrick, and Caroline Jay. "The Future of Research Software Is the Future of Research". In: *Patterns* 6.7 (July 2025). ISSN: 2666-3899. DOI: 10.1016/j.patter.2025.101322.

[120] Anas El Hounsri and Daniel Garijo. "Good Practice versus Reality: A Landscape Analysis of Research Software Metadata Adoption in European Open Science Clusters". In: *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. Apr. 2025, pp. 116–128. DOI: 10.1109/MSR66628.2025.00028.

[121] M. Hucka and M. J. Graham. "Software Search Is Not a Science, Even among Scientists: A Survey of How Scientists and Engineers Find Software". In: *Journal of Systems and Software* 141 (July 2018), pp. 171–191. ISSN: 0164-1212. DOI: 10.1016/j.jss.2018.03.047.

[122] Ludwig Hülk, Jonas Huber, Christian Hofmann, and Christoph Muschner. *Open Energy Metadata (OEMetadata)*. Jan. 2025. URL: https://github.com/OpenEnergyPlatform/oemetadata (last accessed 2026-03-04).

[123] Haley Hunter-Zinck, Alexandre Fioravante de Siqueira, Váleri N. Vásquez, Richard Barnes, and Ciera C. Martinez. "Ten Simple Rules on Writing Clean and Reliable Open-Source Scientific Software". In: *PLOS Computational Biology* 17.11 (Nov. 2021), e1009481. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1009481.

[124] Hendrik Huysksens, Ludwig Hülk, Jonas Huber, and Alexis Michaltsis. *Meta Tool*. Reiner Lemoine Institut. Aug. 2022. URL: https://github.com/rl-institut/meta_tool (last accessed 2026-03-04).

[125] Dorothea Iglezakis, Džulia Terzijska, Susanne Arndt, Sophia Leimer, Johanna Hickmann, Marc Fuhrmans, and Giacomo Lanza. "Modelling Scientific Processes With the M4i Ontology". In: *Proceedings of the Conference on Research Data Infrastructure* 1 (Sept. 2023). ISSN: 2941-296X. DOI: 10.52825/cordi.v1i.271.

[126] Damien Irving, Kate Hertweck, Luke Johnston, Joel Ostblom, Charlotte Wickham, and Greg Wilson. *Research Software Engineering with Python: Building Software That Makes Research Possible*. First edition. New York: Chapman and Hall/CRC, Aug. 2021. ISBN: 978-1-003-14348-2. DOI: 10.1201/9781003143482.

[127] Jon Ison, Hans Ienasescu, Piotr Chmura, Emil Rydza, Hervé Ménager, Matúš Kalaš, Veit Schwämmle, Björn Grüning, Niall Beard, Rodrigo Lopez, Severine Duvaud, Heinz Stockinger, Bengt Persson, Radka Svobodová Vařeková, Tomáš Raček, Jiří Vondrášek, Hedi Peterson, Ahto Salumets, Inge Jonassen, Rob Hooft, Tommi Nyrönen, Alfonso Valencia, Salvador Capella, Josep Gelpí, Federico Zambelli, Babis

Savakis, Brane Leskošek, Kristoffer Rapacki, Christophe Blanchet, Rafael Jimenez, Arlindo Oliveira, Gert Vriend, Olivier Collin, Jacques van Helden, Peter Løngreen, and Søren Brunak. "The Bio.Tools Registry of Software Tools and Data Resources for the Life Sciences". In: *Genome Biology* 20.1 (Aug. 2019), p. 164. ISSN: 1474-760X. DOI: 10.1186/s13059-019-1772-6.

[128] Jon Ison, Hans Ienasescu, Emil Rydza, Piotr Chmura, Kristoffer Rapacki, Alban Gaignard, Veit Schwämmle, Jacques van Helden, Matúš Kalaš, and Hervé Ménager. "biotoolsSchema: A Formalized Schema for Bioinformatics Software Description". In: *GigaScience* 10.1 (Jan. 2021), giaa157. ISSN: 2047-217X. DOI: 10.1093/gigascience/giaa157.

[129] Caroline Jay, Robert Haines, and Daniel S. Katz. "Software Must Be Recognised as an Important Output of Scholarly Research". In: *International Journal of Digital Curation* 16.1 (Dec. 2021), pp. 6–6. ISSN: 1746-8256. DOI: 10.2218/ijdc.v16i1.745.

[130] Eric A. Jensen and Daniel S. Katz. "Awareness of FAIR and FAIR4RS among International Research Software Funders". In: *Scientific Data* 12.1 (Apr. 2025), p. 627. ISSN: 2052-4463. DOI: 10.1038/s41597-025-04820-4.

[131] Eric A. Jensen and Daniel S. Katz. "From Code to Tenure: Valuing Research Software in Academia". In: *Commonplace* (Dec. 2023). DOI: 10.21428/6ffd8432.8f39775d.

[132] Eric A. Jensen and Daniel S. Katz. *Strategic Priorities and Challenges in Research Software Funding: Results from an International Survey*. Preprint. Jan. 2025. DOI: 10.12688/f1000research.155879.2.

[133] Rafael C. Jiménez, Mateusz Kuzak, Monther Alhamdoosh, Michelle Barker, Bérénice Batut, Mikael Borg, Salvador Capella-Gutierrez, Neil Chue Hong, Martin Cook, Manuel Corpas, Madison Flannery, Leyla Garcia, Josep Ll. Gelpí, Simon Gladman, Carole Goble, Montserrat González Ferreiro, Alejandra Gonzalez-Beltran, Philippa C. Griffin, Björn Grüning, Jonas Hagberg, Petr Holub, Rob Hooft, Jon Ison, Daniel S. Katz, Brane Leskošek, Federico López Gómez, Luis J. Oliveira, David Mellor, Rowland Mosbergen, Nicola Mulder, Yasset Perez-Riverol, Robert Pergl, Horst Pichler, Bernard Pope, Ferran Sanz, Maria V. Schneider, Victoria Stodden, Radosław Suchecki, Radka Svobodová Vařeková, Harry-Anton Talvik, Ilian Todorov, Andrew Treloar, Sonika Tyagi, Maarten van Gompel, Daniel Vaughan, Allegra Via, Xiaochuan Wang, Nathan S. Watson-Haigh, and Steve Crouch. "Four Simple Recom-

mendations to Encourage Best Practices in Research Software". In: *F1000Research* 6 (June 2017), p. 876. ISSN: 2046-1402. DOI: 10.12688/f1000research.11407.1.

[134] Arne Johanson and Wilhelm Hasselbring. "Software Engineering for Computational Science: Past, Present, Future". In: *Computing in Science & Engineering* 20.2 (Mar. 2018), pp. 90–109. ISSN: 1558-366X. DOI: 10.1109/MCSE.2018.021651343.

[135] Capers Jones. *Software Engineering Best Practices*. 1st edition. New York: McGraw Hill, Oct. 2009. ISBN: 978-0-07-162161-8. URL: https://dl.acm.org/doi/book/10.5555/1594755.

[136] Matthew B. Jones, Carl Boettiger, Abby Cabunoc Mayes, Arfon M. Smith, Morane Gruenpeter, Valentin Lorentz, Thomas Morrell, Daniel Garijo, Peter Slaughter, Kyle E. Niemeyer, Yolanda Gil, Martin Fenner, Krzysztof Nowak, Mark Hahnel, Luke Coy, Alice Allen, Mercè Crosas, Ashley Sands, Neil Chue Hong, Patricia Cruse, Daniel S Katz, Carole Goble, Bryce Mecum, Alejandra Gonzalez-Beltran, and Noam Ross. *CodeMeta: An Exchange Schema for Software Metadata. Version 3.0.* 2023. URL: https://codemeta.github.io/terms/ (last accessed 2026-03-04).

[137] Guido Juckeland, Doris Dransch, and Bernadette Fritzsch. "Research Software as a New Key Performance Indicator for Evaluation in the Helmholtz Association". In: *Electronic Communications of the EASST* 85 (Dec. 2025). ISSN: 1863-2122. DOI: 10.14279/eceasst.v85.2697.

[138] Reiner Jung, Sven Gundlach, and Wilhelm Hasselbring. "Software Development Processes in Ocean System Modeling". In: *International Journal of Modeling, Simulation, and Scientific Computing* 13.02 (Apr. 2022), p. 2230002. ISSN: 1793-9623, 1793-9615. DOI: 10.1142/S1793962322300023.

[139] Georgia M. Kapitsaki, Nikolaos D. Tselikas, and Ioannis E. Foukarakis. "An Insight into License Tools for Open Source Software Systems". In: *Journal of Systems and Software* 102 (Apr. 2015), pp. 72–87. ISSN: 0164-1212. DOI: 10.1016/j.jss.2014.12.050.

[140] Daniel S. Katz, Michelle Barker, Neil P. Chue Hong, Leyla Jael Castro, and Paula Andrea Martinez. *The FAIR4RS Team: Working Together to Make Research Software FAIR*. Tech. rep. Zenodo, June 2021. DOI: 10.5281/zenodo.5037157.

[141] Daniel S. Katz, Jeffrey C. Carver, Neil P. Chue Hong, Sandra Gesing, Simon Hettrick, Tom Honeyman, Karthik Ram, and Nicholas Weber. "Addressing Research Software Sustainability via Institutes". In: *2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS)*. June 2021, pp. 11–12. DOI: 10.1109/BoKSS52540.2021.00013.

[142] Daniel S. Katz, Neil P. Chue Hong, Tim Clark, August Muench, Shelley Stall, Daina Bouquin, Matthew Cannon, Scott Edmunds, Telli Faez, Patricia Feeney, Martin Fenner, Michael Friedman, Gerry Grenier, Melissa Harrison, Joerg Heber, Adam Leary, Catriona MacCallum, Hollydawn Murray, Erika Pastrana, Katherine Perry, Douglas Schuster, Martina Stockhause, and Jake Yeston. "The Importance of Software Citation". In: *F1000Research* 9 (Oct. 2020), p. 1257. ISSN: 2046-1402. DOI: 10.12688/f1000research.26932.1.

[143] Daniel S. Katz, Morane Gruenpeter, and Tom Honeyman. "Taking a Fresh Look at FAIR for Research Software". In: *Patterns* 2.3 (Mar. 2021). ISSN: 2666-3899. DOI: 10.1016/j.patter.2021.100222.

[144] Daniel S. Katz, Eric A. Jensen, and Michelle Barker. *Understanding and Advancing Research Software Grant Funding Models*. Preprint. July 2025. DOI: 10.12688/openreseurope.20210.1.

[145] Daniel S. Katz, Lois Curfman McInnes, David E. Bernholdt, Abigail Cabunoc Mayes, Neil P. Chue Hong, Jonah Duckles, Sandra Gesing, Michael A. Heroux, Simon Hettrick, Rafael C. Jimenez, Marlon Pierce, Belinda Weaver, and Nancy Wilkins-Diehr. "Community Organizations: Changing the Culture in Which Research Software Is Developed and Sustained". In: *Computing in Science & Engineering* 21.2 (Mar. 2019), pp. 8–24. ISSN: 1558-366X. DOI: 10.1109/MCSE.2018.2883051.

[146] Timo Kehrer, Robert Haines, Guido Juckeland, Shurui Zhou, and David E. Bernholdt. "Do Research Software Engineers and Software Engineering Researchers Speak the Same Language?" In: *Computing in Science & Engineering* 27.2 (Apr. 2025), pp. 18–26. ISSN: 1558-366X. DOI: 10.1109/MCSE.2025.3557236.

[147] Aidan Kelley and Daniel Garijo. "A Framework for Creating Knowledge Graphs of Scientific Software Metadata". In: *Quantitative Science Studies* 2.4 (Dec. 2021), pp. 1423–1446. ISSN: 2641-3337. DOI: 10.1162/qss_a_00167.

[148] Dominic Kempf, Frank Löffler, René Caspart, Bernd Flemisch, Florian Goth, Jan Linxweiler, Philipp Matthias Schäfer, Robert Speck, Alexander Struck, Markus J Ankenbrand, Leyla Jael Castro, Iris Ehlert, Jean-Noël Grad, Axel Loewe, Michael Schlottke-Lakemper, Uwe Schmitt, and S Sommer. *Establishing Central Research Software Engineering Units in German Research Institutions*. Nov. 2025. URL: https://de-rse.org/2023_paper-RSE-groups/paper.pdf (last accessed 2026-03-04).

[149] Sophie Kernchen, Michael Meinel, Stephan Druskat, Michael Fritzsche, David Pape, and Oliver Bertuch. "Extending and Applying Automated HERMES Software Publication Workflows". In: *Electronic Communications of the EASST* 83 (Feb. 2025). ISSN: 1863-2122. DOI: 10.14279/eceasst.v83.2624.

[150] Patrick Kuckertz, Jan Göpfert, Oliver Karras, David Neuroth, Julian Schönau, Rodrigo Pueblas, Stephan Ferenz, Felix Engel, Noah Pflugradt, Jann M. Weinand, Astrid Nieße, Sören Auer, and Detlef Stolten. "DataDesc: A Framework for Creating and Sharing Technical Metadata for Research Software Interfaces". In: *Patterns* 5.11 (Nov. 2024), p. 101064. ISSN: 2666-3899. DOI: 10.1016/j.patter.2024.101064.

[151] Anna-Lena Lamprecht, Leyla Garcia, Mateusz Kuzak, Carlos Martinez, Ricardo Arcila, Eva Martin Del Pico, Victoria Dominguez Del Angel, Stephanie van de Sandt, Jon Ison, Paula Andrea Martinez, Peter McQuilton, Alfonso Valencia, Jennifer Harrow, Fotis Psomopoulos, Josep Ll Gelpi, Neil Chue Hong, Carole Goble, and Salvador Capella-Gutierrez. "Towards FAIR Principles for Research Software". In: *Data Science* 3.1 (Jan. 2020), pp. 37–59. ISSN: 2451-8484. DOI: 10.3233/DS-190026.

[152] Anna-Lena Lamprecht, Carlos Martinez-Ortiz, Michelle Barker, Sadie L. Bartholomew, Justin Barton, Neil Chue Hong, Jeremy Cohen, Stephan Druskat, Jeremy Forest, Jean-Noël Grad, Daniel S. Katz, Robin Richardson, Robert Rosca, Douwe Schulte, Alexander Struck, and Marion Weinzierl. "What Do We (Not) Know About Research Software Engineering?" In: *Journal of Open Research Software* 10.1 (Dec. 2022), p. 11. ISSN: 2049-9647. DOI: 10.5334/jors.384.

[153] Benjamin D. Lee. "Ten Simple Rules for Documenting Scientific Software". In: *PLOS Computational Biology* 14.12 (Dec. 2018), e1006561. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1006561.

[154] James R. Lewis. "The System Usability Scale: Past, Present, and Future". In: *International Journal of Human–Computer Interaction* 34.7 (July 2018), pp. 577–590. ISSN: 1044-7318. DOI: 10.1080/10447318.2018.1455307.

[155] Markus List, Peter Ebert, and Felipe Albrecht. "Ten Simple Rules for Developing Usable Software in Computational Biology". In: *PLOS Computational Biology* 13.1 (Jan. 2017), e1005265. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005265.

[156] Konstantin Löffler, Karlo Hainsch, Thorsten Burandt, Pao-Yu Oei, Claudia Kemfert, and Christian Von Hirschhausen. "Designing a Model for the Global Energy System—GENeSYS-MOD: An Application of the Open-Source Energy Modeling System (OSeMOSYS)". In: *Energies* 10.10 (Oct. 2017), p. 1468. DOI: 10.3390/en10101468.

[157] Florian Mannseicher, Jan Linxweiler, Daniel Adrian Krupka, Robert Speck, Claire Wyatt, Dorothee Stängle, Katja Linnemann, Guido Juckeland, Frank Löffler, Eckhard Kadasch, Zeki Mustafa Dogan, and Falk Schlegelmilch. *Key issues for the design of a German research software institution - General challenges, specific needs and possible solutions*. Tech. rep. Oct. 2025. DOI: 10.5281/zenodo.17347366.

[158] Allen Mao, Daniel Garijo, and Shobeir Fakhraei. "SoMEF: A Framework for Capturing Scientific Software Metadata from Its Documentation". In: *2019 IEEE International Conference on Big Data (Big Data)*. Dec. 2019, pp. 3032–3037. DOI: 10.1109/BigData47090.2019.9006447.

[159] Jochen Markard. "The next Phase of the Energy Transition and Its Implications for Research and Policy". In: *Nature Energy* 3.8 (Aug. 2018), pp. 628–633. ISSN: 2058-7546. DOI: 10.1038/s41560-018-0171-7.

[160] Eva Martín del Pico, Josep Lluís Gelpí, and Salvador Capella-Gutierrez. "FAIRsoft—a Practical Implementation of FAIR Principles for Research Software". In: *Bioinformatics* 40.8 (July 2024). ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btae464.

[161] Michael Meinel, Stephan Druskat, Oliver Bertuch, Oliver Knodel, David Pape, Kernchen Sophie, and Nitai Heeb. *Hermes*. Mar. 2025. DOI: 10.5281/zenodo.14931650.

[162] Steven J. Miller. *Metadata for Digital Collections : A How-to-Do-It Manual*. Second edition. Chicago, 2022. ISBN: 978-0-8389-3800-3.

[163] Robert Mischke, Kathrin Schaffert, Dominik Schneider, and Alexander Weinert. *Automated and Manual Testing as Part of the Research Software Development Process of RCE*. Preprint. Apr. 2022. DOI: 10.48550/arXiv.2204.05600.

[164] Andrew Morin, Jennifer Urban, and Piotr Sliz. "A Quick Guide to Software Licensing for the Scientist-Programmer". In: *PLoS Computational Biology* 8.7 (July 2012). Ed. by Fran Lewitter, e1002598. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1002598.

[165] Michael D. Myers and Michael Newman. "The Qualitative Interview in IS Research: Examining the Craft". In: *Information and Organization* 17.1 (Jan. 2007), pp. 2–26. ISSN: 1471-7727. DOI: 10.1016/j.infoandorg.2006.11.001.

[166] Mikael Nilsson, Tom Baker, and Pete Johnston. *The Singapore Framework for Dublin Core Application Profiles*. Jan. 2008. URL: https://www.dublincore.org/specifications/dublin-core/singapore-framework/ (last accessed 2026-03-04).

[167] Helen Noble and Joanna Smith. "Issues of Validity and Reliability in Qualitative Research". In: *Evidence-Based Nursing* 18.2 (Apr. 2015), pp. 34–35. ISSN: 1367-6539, 1468-9618. DOI: 10.1136/eb-2015-102054.

[168] James M. Osborne, Miguel O. Bernabeu, Maria Bruna, Ben Calderhead, Jonathan Cooper, Neil Dalchau, Sara-Jane Dunn, Alexander G. Fletcher, Robin Freeman, Derek Groen, Bernhard Knapp, Greg J. McInerny, Gary R. Mirams, Joe Pitt-Francis, Biswa Sengupta, David W. Wright, Christian A. Yates, David J. Gavaghan, Stephen Emmott, and Charlotte Deane. "Ten Simple Rules for Effective Computational Research". In: *PLOS Computational Biology* 10.3 (Mar. 2014), e1003506. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003506.

[169] Rossana Paciello, Luca Trani, Daniele Bailo, Valerio Vinciarelli, and Manuela Sbarra. "SHAPEness: A SHACL-Driven Metadata Editor". In: *Metadata and Semantic Research*. Ed. by Emmanouel Garoufallou and Andreas Vlachidis. Cham: Springer Nature Switzerland, 2023, pp. 274–288. ISBN: 978-3-031-39141-5. DOI: 10.1007/978-3-031-39141-5_23.

[170] Nikos Palavitsinis, Nikos Manouselis, and Salvador Sanchez Alonso. "Evaluation of a Metadata Application Profile for Learning Resources on Organic Agriculture". In: *Metadata and Semantic Research*. Ed. by Fabio Sartori, Miguel Angel Sicilia, and Nikos Manouselis. Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2009, pp. 270–281. ISBN: 978-3-642-04590-5. DOI: 10.1007/978-3-642-04590-5_26.

[171] Hyoungjoo Park and Dietmar Wolfram. "Research Software Citation in the Data Citation Index: Current Practices and Implications for Research Software Sharing and Reuse". In: *Journal of Informetrics* 13.2 (May 2019), pp. 574–582. ISSN: 1751-1577. DOI: 10.1016/j.joi.2019.03.005.

[172] Bhavesh Patel, Sanjay Soundarajan, Hervé Ménager, and Zicheng Hu. "Making Biomedical Research Software FAIR: Actionable Step-by-step Guidelines with a User-support Tool". In: *Scientific Data* 10.1 (Aug. 2023), p. 557. ISSN: 2052-4463. DOI: 10.1038/s41597-023-02463-x.

[173] Matthew Patrick, James Elderfield, Richard O. J. H. Stutt, Andrew Rice, and Christopher A. Gilligan. "Software Testing in a Scientific Research Group". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. Pisa Italy: ACM, Apr. 2016, pp. 1454–1459. ISBN: 978-1-4503-3739-7. DOI: 10.1145/2851613.2851783.

[174] S. Camille Peres, Tri Pham, and Ronald Phillips. "Validation of the System Usability Scale (SUS): SUS in the Wild". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 57.1 (Sept. 2013), pp. 192–196. ISSN: 1071-1813. DOI: 10.1177/1541931213571043.

[175] Yasset Perez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin, and Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". In: *PLOS Computational Biology* 12.7 (July 2016), e1004947. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1004947.

[176] Nicolas Prat, Isabelle Comyn-Wattiau, and Jacky Akoka. "A Taxonomy of Evaluation Methods for Information Systems Artifacts". In: *Journal of Management Information Systems* 32.3 (2015), pp. 229–267. ISSN: 0742-1222. JSTOR: 26614012. URL: https://www.jstor.org/stable/26614012 (last accessed 2026-03-04).

[177] Nils Preuß, Matthias Bodenbenner, Benedikt Heinrichs, Jürgen Windeck, Mario Moser, and Marc Fuhrmans. *Creating Application-Specific Metadata Profiles While Improving Interoperability and Consistency of Research Data for the Engineering Sciences*. Preprint. 2023. DOI: 10.26083/TUPRINTS-00024573.

[178] Andreas Prlić and James B. Procter. "Ten Simple Rules for the Open Development of Scientific Software". In: *PLOS Computational Biology* 8.12 (Dec. 2012), e1002802. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1002802.

[179] Iratxe Puebla, Giorgio A. Ascoli, Jeffrey Blume, John Chodacki, Joshua Finnell, David N. Kennedy, Bernard Mair, Maryann E. Martone, Jamie Wittenberg, and Jean-Baptiste Poline. "Ten Simple Rules for Recognizing Data and Software Contributions in Hiring, Promotion, and Tenure". In: *PLOS Computational Biology* 20.8 (Aug. 2024), e1012296. ISSN: 1553-734X. DOI: 10.1371/journal.pcbi.1012296.

[180] Ramiz Qussous, Corinna Seiwerth, Jan Sören Schwarz, Nan Liu, Philipp Schmurr, Zhiyu Pan, and Stephan Ferenz. *D5.1.1.2 Energy Simulation Software Metadata Schema*. Tech. rep. Zenodo, Jan. 2026. DOI: 10.5281/zenodo.18390186.

[181] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. "Robust Speech Recognition via Large-Scale Weak Supervision". In: *Proceedings of the 40th International Conference on Machine Learning*. PMLR, July 2023, pp. 28492–28518. URL: https://proceedings.mlr.press/v202/radford23a.html (last accessed 2026-03-04).

[182] *Recommendations for the Implementation of Guidelines and Policies on Research Software Management at the Helmholtz Centers*. Tech. rep. Helmholtz Open Science Office, 2019. DOI: 10.48440/os.helmholtz.040.

[183] Joeri Rogelj, Drew Shindell, Kejun Jiang, Solomone Fifita, Piers Forster, Veronika Ginzburg, Collins Handa, Shigeki Kobayashi, Elmar Kriegler, Luis Mundaca, Roland Séférian, and Maria Virginia Vilariño. "Mitigation Pathways Compatible with 1.5°C in the Context of Sustainable Development". In: *Global Warming of 1.5° C. An IPCC Special Report on the Impacts of Global Warming of 1.5° C above Pre-Industrial Levels and Related Global Greenhouse Gas Emission Pathways, in the Context of Strengthening the Global Response to the Threat of Climate Change, Sustainable Development, and Efforts to Eradicate Poverty*. Ed. by Valérie Masson-Delmotte, Panmao Zhai, Hans-Otto Pörtner, Debra Roberts, Jim Skea, Priyadarshi R. Shukla, Anna Pirani, Wilfran Moufouma-Okia, Clotilde Péan, Roz Pidcock, Sarah Connors, J. B. Robin Matthews, Yang Chen, Xiao Zhou, Melissa I. Gomis, Elisabeth Lonnoy, Tom Maycock, Melinda Tignor, and Tim Waterfield. worldwide: In Press, 2018, pp. 93–174. URL: https://www.ipcc.ch/site/assets/uploads/sites/2/2019/02/SR15_Chapter2_Low_Res.pdf (last accessed 2026-03-04).

[184] Joseph D. Romano and Jason H. Moore. "Ten Simple Rules for Writing a Paper about Scientific Software". In: *PLOS Computational Biology* 16.11 (Nov. 2020), e1008390. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1008390.

[185] Mário Rosado de Souza, Robert Haines, Markel Vigo, and Caroline Jay. "What Makes Research Software Sustainable? An Interview Study with Research Software Engineers". In: *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. May 2019, pp. 135–138. DOI: 10.1109/CHASE.2019.00039.

[186] Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H. Nguyen, Sara Brin Rosenthal, Fernando Pérez, and Peter W. Rose. "Ten Simple Rules for Writing and Sharing Computational Analyses in Jupyter Notebooks". In: *PLOS Computational Biology* 15.7 (July 2019), e1007007. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1007007.

[187] Paul J. Runci and James J. Dooley. "Research and Development Trends for Energy". In: *Encyclopedia of Energy*. Ed. by Cutler J. Cleveland. New York: Elsevier, Jan. 2004, pp. 443–449. ISBN: 978-0-12-176480-7. DOI: 10.1016/B0-12-176480-X/00477-0.

[188] Caitlin Sadowski, Kathryn T. Stolee, and Sebastian Elbaum. "How Developers Search for Code: A Case Study". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, Aug. 2015, pp. 191–201. ISBN: 978-1-4503-3675-8. DOI: 10.1145/2786805.2786855.

[189] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. "Ten Simple Rules for Reproducible Computational Research". In: *PLOS Computational Biology* 9.10 (Oct. 2013), e1003285. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003285.

[190] Tobias Schlauch, Michael Meinel, and Carina Haupt. *DLR Software Engineering Guidelines*. Tech. rep. Zenodo, Aug. 2018. DOI: 10.5281/zenodo.1344612.

[191] Rico Schrage, Jens Sager, Jan Philipp Hörding, and Stefanie Holly. "Mango: A Modular Python-Based Agent Simulation Framework". In: *SoftwareX* 27 (Sept. 2024), p. 101791. ISSN: 2352-7110. DOI: 10.1016/j.softx.2024.101791.

[192] Jan Schwarz and Sebastian Lehnhoff. "Ontological Integration of Semantics and Domain Knowledge in Energy Scenario Co-simulation". In: *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Vienna, Austria: SCITEPRESS - Science and Technology Publications, 2019, pp. 127–136. ISBN: 978-989-758-382-7. DOI: 10.5220/0008069801270136.

[193] Vasiliy Seibert, Andreas Rausch, and Stefan Wittek. *Betty Research Engine*. Nov. 2024. URL: https://gitlab.com/tuc-isse/public/betty-research-engine (last accessed 2026-03-04).

[194] Vasiliy Seibert, Andreas Rausch, and Stefan Wittek. "Betty's (Re)Search Engine: A Client-Based Search Engine for Research Software Stored in Repositories." In: *ing.grid* 1.2 (May 2024). ISSN: 2941-1300. DOI: 10.48694/inggrid.3953.

[195] Kang G. Shin and Parameswaran Ramanathan. "Real-Time Computing: A New Discipline of Computer Science and Engineering". In: *Proceedings of the IEEE* 82.1 (Jan. 1994), pp. 6–24. ISSN: 1558-2256. DOI: 10.1109/5.259423.

[196] David Silverman, ed. *Qualitative Research*. Forth revised edition. Los Angeles: Sage, 2016. ISBN: 978-1-4739-1656-2.

[197] Arfon M. Smith, Daniel S. Katz, and Kyle E. Niemeyer. "Software Citation Principles". In: *PeerJ Computer Science* 2 (Sept. 2016), e86. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.86.

[198] Arfon M. Smith, Kyle E. Niemeyer, Daniel S. Katz, Lorena A. Barba, George Githinji, Melissa Gymrek, Kathryn D. Huff, Christopher R. Madan, Abigail Cabunoc Mayes, Kevin M. Moerman, Pjotr Prins, Karthik Ram, Ariel Rokem, Tracy K. Teal, Roman Valls Guimera, and Jacob T. Vanderplas. "Journal of Open Source Software (JOSS): Design and First-Year Review". In: *PeerJ Computer Science* 4 (Feb. 2018), e147. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.147.

[199] Laura Soito and Lorraine J. Hwang. "Citations for Software: Providing Identification, Access and Recognition for Research Software". In: *International Journal of Digital Curation* 11.2 (2016), pp. 48–63. ISSN: 1746-8256. DOI: 10.2218/ijdc.v11i2.390.

[200] Cornelius Steinbrink, Marita Blank-Babazadeh, André El-Ama, Stefanie Holly, Bengt Lüers, Marvin Nebel-Wenner, Rebeca P. Ramírez Acosta, Thomas Raub, Jan Sören Schwarz, Sanja Stark, Astrid Nieße, and Sebastian Lehnhoff. "CPES Testing with Mosaik: Co-Simulation Planning, Execution and Analysis". In: *Applied Sciences* 9.5 (Jan. 2019), p. 923. DOI: 10.3390/app9050923.

[201] Alexandro Steinert, Stephan Ferenz, and Astrid Nieße. "Choosing the Right Ontology to Describe Research Data in the Energy Domain". In: *SIGENERGY Energy Inform. Rev.* 4.4 (Feb. 2025), pp. 33–48. DOI: 10.1145/3717413.3717417.

[202] Frankie Stevens. *Understanding How Researchers Find Research Software for Research Practice*. Tech. rep. Zenodo, Dec. 2022. DOI: 10.5281/zenodo.7340034.

[203] Markus Stocker, Allard Oelen, Mohamad Yaser Jaradeh, Muhammad Haris, Omar Arab Oghli, Golsa Heidari, Hassan Hussein, Anna-Lena Lorenz, Salomon Kabenamualu, Kheir Eddine Farfar, Manuel Prinz, Oliver Karras, Jennifer D'Souza, Lars Vogt, and Sören Auer. "FAIR Scientific Information with the Open Research Knowledge Graph". In: *FAIR Connect* 1.1 (Apr. 2023), pp. 19–21. ISSN: 2949-799X. DOI: 10.3233/FC-221513.

[204] Victoria Stodden, Marcia McNutt, David H. Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A. Heroux, John P.A. Ioannidis, and Michela Taufer. "Enhancing Reproducibility for Computational Methods". In: *Science* 354.6317 (Dec. 2016), pp. 1240–1241. DOI: 10.1126/science.aah6168.

[205] Victoria Stodden and Sheila Miguez. "Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research". In: *Journal of open research software* 2.1 (July 2014), e21. ISSN: 2049-9647. URL: https://openresearchsoftware.metajnl.com/articles/10.5334/jors.ay (last accessed 2026-03-04).

[206] Anselm L. Strauss and Juliet M. Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Thousand Oaks, CA, US: Sage Publications, Inc, 1990, p. 270. ISBN: 978-0-8039-3250-0.

[207] Alexander Struck. "Research Software Discovery: An Overview". In: *2018 IEEE 14th International Conference on E-Science (e-Science)*. Oct. 2018, pp. 33–37. DOI: 10.1109/eScience.2018.00016.

[208] Morgan Taschuk and Greg Wilson. "Ten Simple Rules for Making Research Software More Robust". In: *PLOS Computational Biology* 13.4 (Apr. 2017), e1005412. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005412.

[209] Leon Thurner, Alexander Scheidler, Florian Schäfer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. "Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems". In: *IEEE Transactions on Power Systems* 33.6 (Nov. 2018), pp. 6510–6521. ISSN: 1558-0679. DOI: 10.1109/TPWRS.2018.2829021.

[210] Ana Trisovic, Matthew K. Lau, Thomas Pasquier, and Mercè Crosas. "A Large-Scale Study on Research Code Quality and Execution". In: *Scientific Data* 9.1 (Feb. 2022), p. 60. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01143-6.

[211] Thomas S Tullis and Jacqueline N Stetson. "A Comparison of Questionnaires for Assessing Website Usability". In: *Usability Professional Association Conference*. 2004. URL: https://dgs.ie/wp-content/uploads/2016/12/A-Comparison-of-Questionnaires-for-Assessing-Website-Usability-.pdf (last accessed 2026-03-04).

[212] Viswanath Venkatesh and Hillol Bala. "Technology Acceptance Model 3 and a Research Agenda on Interventions". In: *Decision Sciences* 39 (May 2008), pp. 273–315. DOI: 10.1111/j.1540-5915.2008.00192.x.

[213] Viswanath Venkatesh and Fred D. Davis. "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies". In: *Management Science* 46.2 (Feb. 2000), pp. 186–204. ISSN: 0025-1909. DOI: 10.1287/mnsc.46.2.186.11926.

[214] Viswanath Venkatesh, Michael G. Morris, Gordon B. Davis, and Fred D. Davis. "User Acceptance of Information Technology: Toward a Unified View". In: *MIS Quarterly* 27.3 (2003), pp. 425–478. ISSN: 0276-7783. DOI: 10.2307/30036540.

[215] Mike Vogt, Frank Marten, and Martin Braun. "A Survey and Statistical Analysis of Smart Grid Co-Simulations". In: *Applied Energy* 222 (July 2018), pp. 67–78. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2018.03.123.

[216] Erica L. Wagner, Susan V. Scott, and Robert D. Galliers. "The Creation of 'Best Practice' Software: Myth, Reality and Ethics". In: *Information and Organization* 16.3 (Sept. 2006), pp. 251–275. ISSN: 1471-7727. DOI: 10.1016/j.infoandorg.2006.04.001.

[217] Oliver Werth, Stephan Ferenz, and Astrid Nieße. "Requirements for an Open Digital Platform for Interdisciplinary Energy Research and Practice". In: *Wirtschaftsinformatik 2022 Proceedings*. Jan. 2022. URL: https://aisel.aisnet.org/wi2022/sustainable_it/sustainable_it/2 (last accessed 2026-03-04).

[218] *What Is a Research Software Engineer? A Definition by the Netherlands eScience Center*. Tech. rep. The Netherlands eScience Center, June 2023. DOI: 10.5281/zenodo.7994286.

[219] Ole Wieners. *Untersuchung und Charakterisierung von Energieforschungssoftware anhand der FAIR-Kriterien*. Bachelor Thesis. 2021.

[220] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. "The FAIR Guiding Principles for Scientific Data Management and Stewardship". In: *Scientific Data* 3.1 (Mar. 2016), pp. 1–9. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18.

[221] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. "Best Practices for Scientific Computing". In: *PLOS Biology* 12.1 (Jan. 2014), e1001745. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.1001745.

[222] Thomas Wolgast, Stephan Ferenz, and Astrid Nieße. "Reactive Power Markets: A Review". In: *IEEE Access* 10 (2022), pp. 28397–28410. ISSN: 2169-3536. DOI: 10.1109/access.2022.3141235.

[223] W. Wu, J. S. Partridge, and R. W. G. Bucknall. "Simulation of a Stabilised Control Strategy for PEM Fuel Cell and Supercapacitor Hybrid Propulsion System for a City Bus". In: *International Journal of Hydrogen Energy* 43.42 (Oct. 2018), pp. 19763–19777. ISSN: 0360-3199. DOI: 10.1016/j.ijhydene.2018.09.004.

[224] Yo Yehudi, Mikaela Cashman, Michael Goedicke, Wilhelm Hasselbring, Daniel S. Katz, Sebastian Müller, Carole Goble, and Caroline Jay. "Research Software Lifecycles and Stages". In: *Electronic Communications of the EASST* 85 (Dec. 2025). ISSN: 1863-2122. DOI: 10.14279/eceasst.v85.2693.

[225] Marcia Lei Zeng and Jian Qin. *Metadata*. Third edition. London: Facet Publishing, 2022. ISBN: 978-1-78330-588-9.

[226] Yang Zhang, Tao Huang, and Ettore Francesco Bompard. "Big Data Analytics in Smart Grids: A Review". In: *Energy Informatics* 1.1 (Aug. 2018), p. 8. ISSN: 2520-8942. DOI: 10.1186/s42162-018-0007-5.

[227] Christian Zirkelbach, Alexander Krause, and Wilhelm Hasselbring. *Modularization of Research Software for Collaborative Open Source Development*. Preprint. July 2019. DOI: 10.48550/arXiv.1907.05663.