



Electronic Communications of the EASST
Volume 86 (1) Year 2026

deRSE26 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany (Part 1)

Edited by Bernd Flemisch, René Caspart, Bernadette Fritzsch, Florian Goth, Jean-Noël Grad, Florian Thiery

A Community-Based Crosswalk Analysis of Research Software Metadata Schemas

Maryam Sefidbari, Aida Jafarbigloo, Lu Gan, Ulrik Stervbo, Stephan Druskat, Florian Thiery, Brigitte Mathiak, Astrid Nieße, Oliver Werth, Stephan Ferenz, Leyla Jael Castro

DOI: 10.14279/eceasst.v86.1.2800

License: This article is licensed under a CC-BY 4.0 License.

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

A Community-Based Crosswalk Analysis of Research Software Metadata Schemas

Maryam Sefidbari [1], Aida Jafarbigloo [2,3], Lu Gan [4], Ulrik Stervbo [5], Stephan Druskat [6], Florian Thiery [7], Brigitte Mathiak [4], Astrid Nieße [2], Oliver Werth [3], Stephan Ferenz [2,3], Leyla Jael Castro [1]

[1] ZBMED - Information Centre for Life Sciences
[2] Carl von Ossietzky Universität Oldenburg
[3] OFFIS - Institute for Information Technology
[4] GESIS - Leibniz Institute for the Social Sciences
[5] Ruhr University Bochum
[6] Technische Universität Braunschweig
[7] Leibniz-Zentrum für Archäologie (LEIZA)

Abstract: High-quality research software metadata is crucial to achieve Findable, Accessible, Interoperable, and Reusable research software (FAIR4RS). Currently, software metadata is created using a wide range of schemas and vocabularies that differ in structure and scope. This heterogeneity limits interoperability and reuse, leading to inconsistencies across systems. Existing initiatives such as Schema.org[1] and CodeMeta[2] provide an important foundation for research software metadata, but they do not fully capture the diversity of metadata concepts used across the broader ecosystem. In this work, we present a systematic crosswalk analysis of 21 metadata resources including research software *metadata schemas*, *vocabularies*, and *ontologies*, using a reference set of properties from CodeMeta and Schema.org. From this analysis, we identify overlaps and inconsistencies across existing metadata approaches, and assess the extent to which CodeMeta properties are represented across the analyzed schemas. Our analysis identifies distinct areas of gaps within CodeMeta, revealing that dimensions such as research context, operational behavior, modern software infrastructure, and accessibility and ethical considerations are currently missing or underrepresented. Also, our findings provide a consolidated, community-informed view of the current research software metadata landscape, highlighting areas of convergence and divergence among the analyzed resources. This work contributes to ongoing efforts toward implementing the FAIR4RS principles and thus improving the interoperability and reuse of research software.

Keywords: Software Metadata; Schema.org; CodeMeta; Crosswalk; Research Software; Metadata Schema; FAIR4RS

[1] https://schema.org/; accessed 27/07/2026
[2] https://codemeta.github.io/; accessed 27/07/2026

1 Introduction

Research software plays a central role in modern science, enabling data analysis, simulation, and reproducibility across disciplines [HMS+09]. To make such software discoverable, reusable and citable, rich and (semantically) structured metadata is essential [LGK+20]. However, the description of research software is currently fragmented across multiple metadata schemas, formats, and community practices. This fragmentation limits the findability and interoperability of software artifacts and complicates their integration into and comparison across research infrastructures. Addressing these challenges is a key requirement for implementing the Findable, Accessible, Interoperable, and Reusable principles for research software (FAIR4RS) [CKB+22]. FAIR4RS corresponds to a community-driven adaptation of the original FAIR principles to the specific needs of research software, emphasizing practices that improve discoverability, accessibility, interoperability, and reuse of software artifacts such as source code, workflows, and executable software [BHK+22, CKB+22]. Since its inception, FAIR4RS has catalyzed policy statements, case studies, and implementation guidance that promote software as a reusable asset across many disciplines [BHK+22, WAB+25, GWB+25].

A number of metadata schemas and standards have been developed to describe research software, including CodeMeta [JBC+], Citation File Format (CFF) [DSC+21], and ERSmeta [Fer25]. Although these schemas share common concepts, such as authorship, versioning, and licensing, they differ in structure, terminology, and scope. As a result, achieving interoperability between systems that rely on different metadata standards remains challenging [LGK+20]. CodeMeta has been proposed as a unifying schema to facilitate interoperability by providing a common vocabulary and crosswalks between existing standards [JBC+]. Crosswalks are a long-established instrument in metadata practice for bridging heterogeneous schemas without forcing every system onto a single standard [WRV+23]. The mapping specifies how the elements (properties or fields) of one metadata schema correspond to those of another: as an exact equivalence, a broader or narrower relation, or a partial overlap, allowing metadata to be translated, harvested, and reused across systems that would otherwise be incompatible [WRV+23, CFKS23].

To achieve a better understanding of the diverse landscape of metadata approaches for research software, we mapped different metadata approaches together with contributors from the NFDI Section (Meta)Data Working Group on Research Software Metadata (NFDI-RSM-WG) [CFF+26]. In total, we created crosswalks from 21 distinct resources, including 5 ontologies, 3 vocabularies, 12 metadata schemas, and one knowledge graph, to a reference set taken from CodeMeta and Schema.org. These mappings capture correspondences between heterogeneous metadata approaches and provide a consolidated view of shared and divergent concepts. Based on the crosswalks, we conducted a workshop titled *Towards a Community-based Extension for Research Software Metadata from the German Research Community*[3] to evaluate the existing CodeMeta properties and to discuss a set of potential additional elements for a unified metadata schema for research software. The workshop was part of the deRSE26 Conference for Research Software Engineering in Germany[4].

[3] https://events.hifis.net/event/2945/contributions/21102/; accessed 27/07/2026
[4] https://events.hifis.net/event/2945/; accessed 27/07/2026

In this paper, we present (i) a systematic crosswalk analysis of 21 software-related resources including metadata schemas, vocabularies, and ontologies, all mapped to a reference set of CodeMeta and Schema.org properties, (ii) an analysis of the coverage vs. underrepresented dimensions identified in CodeMeta through these mappings, and (iii) identification of the common metadata concepts vs. inconsistencies across the analysed metadata approaches. The property mapping process was conducted collaboratively within the NFDI-RSM-WG, and was informed by feedback from the workshop at the deRSE26 Conference. Ultimately, our work contributes to ongoing efforts toward interoperable and FAIR research software metadata by linking large-scale schema mapping with community-based schema refinement.

2 Background

In this section, we provide a review of the landscape of metadata approaches used to describe research software. We focus on their conceptual foundation, the heterogeneity of existing approaches, and the resulting interoperability challenges. Building on this, we discuss schema alignment strategies, particularly crosswalks centered around CodeMeta, and highlight remaining gaps in the current ecosystem.

2.1 Metadata Schemas and Standards for Research Software

Metadata schemas provide structured representations of information about research software, enabling its description, discovery, and reuse. These schemas vary in scope, ranging from general-purpose vocabularies to domain-specific and ontology-based approaches.

At the foundation, Schema.org has emerged as a widely adopted vocabulary for describing digital objects on the web. It enables machine-actionable metadata through formats such as Resource Description Framework (RDF)[5] and its serializations such as JSON-LD[6], supporting integration into search engines and knowledge graphs. Its extensibility through domain-specific profiles (e.g., Bioschemas [GGGD17] for life science content, and CodeMeta for software) has facilitated interoperability across repositories, registries, and search engines [WRV+23, WAB+25, IIR+21]. CodeMeta provides domain-specific terms that improve semantic alignment for software descriptions, while crosswalks between CodeMeta and other standards (e.g., DataCite [Dat] and RO-Crate[7]) enable broader integration with data and workflow registries [WRV+23, LGK+20, SSC+22].

Building on Schema.org, CodeMeta has emerged as a general-purpose vocabulary for software description [IIR+21, Hab19]. It provides a machine-readable JSON-LD encoding for software metadata that supports core FAIR4RS goals; namely, enabling precise identification, citation, and discovery of software artifacts [IIR+21, KGK+24]. The vocabulary covers a broad range of metadata, from descriptive information such as authors, licenses, identifiers, and repository links to selected technical aspects such as operating systems, runtime platforms, software requirements, and build instructions. These properties support discovery, citation, metadata ex-

[5] https://www.w3.org/TR/rdf12-primer/; accessed 27/07/2026
[6] https://json-ld.org/; accessed 27/07/2026
[7] https://www.researchobject.org/ro-crate/; accessed 27/07/2026

change, and reuse across repositories and registries. Multiple analyses emphasize CodeMeta's role as a practical "*Rosetta Stone*" that maps diverse software descriptions into a unified frame and as a basis for crosswalks that translate content between dialects used by different platforms [IIR+21, Hab19, PSMH23], as well as its ongoing integration with Schema.org terms to maximize machine-actionable interoperability [Hab19].

Beyond these general approaches, several complementary schemas exist that differ in expressiveness and intended use. Formats such as Citation File Format (CFF) [Dru17, DSC+21] are designed to be both human- and machine-readable and focus on citation-related metadata for research software, provided in CITATION.cff files. On the other hand, ontology-based approaches such as Software Description Ontology (SDO) [GRGK], OntoSoft [Ont, GRG15], and Software Ontology (SWO) [MBL+14] provide richer semantic representations. These enable more expressive descriptions of software functionality, provenance, and relationships. OntoSoft organizes software metadata around six reuse-orientated dimensions [GRG15], while SDO is presented as an evolution of OntoSoft [GRGK]. SWO offers detailed representations of software types, tasks, versions, licensing, provenance, and associated data [MBL+14].

Domain-specific metadata schemas further extend this landscape by providing depth within particular scientific areas. For example, biotoolsSchema [IIR+21] offers a formalized XML[8] (Extensible Markup Language) schema for bioinformatics software, anchored to the biotools registry and using EDAM[9] as a value vocabulary to describe function, inputs, and outputs; it is designed to align with CodeMeta, Schema.org, and other initiatives to foster interoperability among life-science software descriptions [IIR+21]. Bioschemas [GGJ17], built on Schema.org, provides domain-tailored markup profiles for life-science resources and supports a wide range of discovery use cases for biological data and tools. Additional crosswalks and extensions facilitate integration with schemas such as DataCite, Dublin Core, and Data Catalogue Vocabulary (DCAT) [W3C], reflecting a broader ecosystem in which software metadata is embedded within catalogs, repositories, and registries [KGK+24, Hab18].

Taken together, these approaches illustrate a diverse ecosystem in which different schemas address overlapping but not identical aspects of research software metadata. Therefore, an overarching metadata framework or alignment strategy is needed to enable interoperability, reduce fragmentation, and support consistent, machine-actionable descriptions of research software across platforms.

2.2 Heterogeneity of Metadata Representations

The key challenge in research software metadata ecosystem is not only that many schemas exist, but that they differ along several dimensions, e.g., in representation format (RDF/XML, OWL, JSON-LD, YAML, XML Schema), in modeling style (record-oriented vs. graph-oriented), and in scope. Metadata descriptions also vary in terminology, granularity, and the balance between descriptive and technical metadata. Similar concepts are often represented differently across schemas, both structurally and semantically. A "dependency", for example, may be represented

[8] https://www.w3.org/TR/xml/; accessed 27/07/2026
[9] The ontology of data analysis and management, https://edamontology.org; accessed 27/07/2026

as softwareRequirements[10] in CodeMeta, as reference[11] in CFF, or modeled through ontology classes and relationships in SWO. Such differences complicate direct alignment and increase the risk of confusion or information loss during integration.

This heterogeneity is further intensified by community specializations. Domain-specific schemas such as biotoolsSchema [IIR+21] are optimized for particular scientific contexts, while more general frameworks such as DataCite serve broader citation and discovery purposes across research outputs, including software and workflows. Similarly, the Description Of A Project (DOAP) [Dum] provides an XML/RDF vocabulary tailored to describe open-source software projects. As a result, the ecosystem exhibits both overlap and asymmetry: while many schemas share core elements such as title, identifier, license, authorship, and version, they diverge in representing runtime, interfaces, inputs and outputs, or domain-specific functionalities.

2.3 Interoperability Challenges and Crosswalks

The heterogeneity of metadata schemas introduces substantial challenges for interoperability. Integrating metadata from different sources requires aligning both semantic meanings and structural representations, which is often non-trivial. Crosswalks address this by making the correspondences between schemas explicit: rather than harmonizing schemas into one model, they document, element by element, how a concept expressed in one schema maps onto another, which relationships are exact, and where meaning is lost or only partially preserved [WRV+23, CFKS23].

The crosswalk methodology used in this study is related to the broader literature on schema matching, where the central task is to identify correspondences between heterogeneous conceptual structures in order to support data integration and interoperability [RB01, ES13]. Unlike many ontology-alignment approaches that aim to generate formal logical mappings between classes, properties, and axioms, our work focuses on metadata-property correspondences that are interpretable by domain experts and directly usable for research software metadata exchange.

The CodeMeta project has established crosswalks[12] to a number of existing metadata standards, providing an important foundation for interoperability across software metadata ecosystems. These mappings illustrate how software metadata can be harmonized to support attribution, discovery, and reuse across platforms [WRV+23, IIR+21]. However, these mappings introduce limitations such as loss of granularity, semantic ambiguity, incomplete alignment between schema elements, or asymmetric information loss due to omission of mapping directionality. In particular, while CodeMeta enhances software findability and citation, it may not fully capture all distinctive traits of software such as dependencies, versioning, and build environments, necessitating complementary schemas and mappings to ensure robust interoperability [VAF+23, KGK+24].

The practical utility of CodeMeta is reinforced by ecosystem tooling, including generators, validators, and converters (e.g., CodeMeta generator/validator[13] and CodeMeta–Zenodo convert-

[10] https://schema.org/softwareRequirements; accessed 27/07/2026
[11] https://w3id.org/cff/1.2.0/#definitionsreference; accessed 27/07/2026
[12] https://codemeta.github.io/crosswalk/; accessed 27/07/2026
[13] https://github.com/codemeta/codemeta-generator; accessed 27/07/2026

ers[14]), which facilitate the creation, validation, transformation, and exchange of codemeta.json metadata while supporting software citation, reproducibility, and FAIR4RS-oriented metadata practices [WRV+23, IIR+21].

Within the FAIR4RS landscape, the CodeMeta schema aligns with the Findable and Interoperable dimensions by enabling persistent identifiers and machine-actionable metadata, while also supporting Reusable metadata practices through clear licensing, versioning, and dependency specifications [VAF+23, IIR+21]. These features contribute to improved interoperability, but do not fully resolve the need for richer and more expressive metadata. Recent analyses highlight the need to complement CodeMeta with richer, domain-specific descriptors to better capture software ecosystems, including build systems, containerization, and computational workflows [KGK+24, MDR+24, GWB+25].

The above-mentioned crosswalk efforts primarily focus on enabling interoperability, but do not systematically identify or evaluate metadata properties that are absent from the current CodeMeta schema. Among the schemas analyzed in our work, 11 approaches, namely OntoSoft, DCAT, SPDX, Wikidata, DataCite, DOAP, SWO, CFF, Java (Maven), Python Packaging Metadata, and R packaging metadata, overlap with existing CodeMeta crosswalks. Our work extends the existing landscape by systematically mapping a total of 21 software-related schemas, vocabularies, and ontologies to CodeMeta. Furthermore, our approach identifies elements not currently covered by CodeMeta. The next section describes the methodology used to perform the crosswalk analysis.

3 Methodology

This section describes the methodology used for the crosswalk analysis of 21 software and research software metadata approaches. The goal of this analysis is to examine how metadata concepts are represented across different approaches, assess the extent to which these concepts are covered by CodeMeta as a reference schema, and identify common concepts and inconsistencies in the current software metadata landscape.

To evaluate semantic interoperability and identify architectural overlaps, we applied a structured metadata crosswalk approach. A metadata schema crosswalk is a well-established technical procedure used in data science to map equivalent terms across disparate systems, align diverse metadata elements, and identify underlying structural gaps [CFKS23, WRV+23]. Utilizing a baseline or reference schema to analyze correspondences against multiple external schemas is a standardized methodology in the metadata ecosystem for creating unified frameworks and identifying underrepresented dimensions [Sci25].

To support this analysis, we defined a reference set of 68 metadata properties, comprising 60 properties from CodeMeta and 8 additional properties from Schema.org. This set represents a broad range of metadata concepts commonly used to describe research software and serves as a common basis for comparing heterogeneous software metadata approaches.

The crosswalk mappings were used to systematically identify correspondences between metadata concepts across the selected schemas and the reference set of properties. In addition to close matches, partial matches were considered in cases where no direct equivalence existed but

[14] https://github.com/marketplace/actions/codemeta-to-zenodo-converter; accessed 27/07/2026

the semantics were sufficiently similar. This approach enabled a comprehensive comparison of metadata coverage while accounting for differences in terminology and structure.

A direct outcome of the analysis is the distinction between (i) metadata elements that can be directly aligned with the reference set of properties and (ii) metadata elements with no appropriate match within this set. The latter were collected and further analyzed to better understand gaps in the reference schema, particularly with respect to research-specific aspects.

In this section, we discuss the selection of metadata schemas, vocabularies, and ontologies included in the analysis (Section 3.1), the rationale behind the reference schema and the reference set (Section 3.2), the reference property set and its categorization into nine conceptual groups (Section 3.3), and the mapping strategy and collaborative process used to establish correspondences between the 21 metadata resources and the reference set (Section 3.4).

3.1 Schema Selection

Prior to the mapping process, we selected a number of relevant metadata approaches, including 21 metadata schemas, vocabularies, and ontologies. The selected sources represent a diverse set of metadata models used to describe software and research software, including domain-specific approaches, general-purpose schemas, and purpose-specific formats.

Candidates were identified through a community-based process involving members of the NFDI-RSM-WG, who proposed schemas, vocabularies, and ontologies based on their direct experience with metadata standards and research software in their respective domains and communities. The initial pool of candidates included 28 schemas and vocabularies. Next, candidates were collectively discussed within the working group and retained if they addressed research software in a sufficiently general manner; candidates tied to a single proprietary tool or platform rather than to software description as such were excluded. For example, metadata associated with MathWorks was not included, as it is specific to describing MATLAB code rather than research software in general. This filtering step ensured that the final set of schemas is applicable across a broad range of software and research contexts rather than being tied to individual tools, while still allowing schemas with a narrower scientific scope (e.g., domain-specific schemas such as biotoolsSchema or ERSmeta) to be included, since these remain generalizable within their respective research domains rather than being bound to a single proprietary product.

As a result, the final selection of metadata schemas, vocabularies, and ontologies is a broad and representative coverage of metadata models across different domains and use cases, while relying on the collective expertise of the working group. To the best of our knowledge, this work compiles a comprehensive set of software related metadata schemas, vocabularies, and ontologies, while addressing the lack of a systematic overview of research software metadata approaches in the existing literature (as described in Section 2). An overview of the analyzed metadata approaches, including 5 ontologies, 3 vocabularies, 11 metadata schemas, one knowledge graph, and one registry, is provided in Table 1.

3.2 Rationale Behind the Reference Set

We selected CodeMeta as the central target for all mappings for several reasons. First, unlike domain-specific schemas such as biotoolsSchema [IIR+21] and ERSmeta, CodeMeta describes

research software in a domain-agnostic manner, so using it as the reference set avoids biasing the analysis towards any single scientific field. Second, in contrast to general-purpose data and citation schemas such as DataCite or DCAT, CodeMeta is built for software and was explicitly designed as an interoperability vocabulary, providing a common frame and a set of established crosswalks to other standards [JBC+]; this makes it a natural hub in the mediated-schema design. Third, being built on Schema.org and encoded in JSON-LD, CodeMeta inherits web-scale interoperability and is supported by mature tooling for generation and validation, which makes CodeMeta as a reference set directly actionable.

The reference set of properties was defined prior to the crosswalk creation process by the NFDI-RSM-WG, based on a criterion reflecting the close relationship between CodeMeta and Schema.org: a property was included in the reference set if it either originates from CodeMeta or is a software-relevant property defined in Schema.org that had not been incorporated into CodeMeta. Applying this criterion yielded a reference set comprising the 60 properties of CodeMeta version 3 together with 8 additional Schema.org properties considered relevant to describing research software.

Table 1: Overview of metadata schemas, vocabularies, and ontologies mapped to CodeMeta

Name	Type	Scope
AlgoOnto [Mat]	Ontology	Algorithms & computational methods
HITO [JAD+]	Ontology	Health IT ontology
OntoSoft [Ont, GGMR16]	Ontology	Software metadata & provenance
SDO [GOK+19]	Ontology	Software description & classification
Software Ontology [Sof]	Ontology	General software concepts
DCAT [W3C]	Vocabulary	Data catalog metadata
DOAP [Dum]	Vocabulary	Description Of A Project
SPDX [Lin]	Vocabulary	Software licensing & compliance
BARTOC [Led]	Schema	Terminology registry metadata
biotoolsSchema [bio16, IIR+21]	Schema	Bioinformatics tools metadata
Citation File Format [DSC+21]	Schema	Software citation metadata
DataCite [Dat]	Schema	DOI & publication metadata
DataDesc [KGK+, KGK+24]	Schema	Dataset & software description
ERSmeta [Fer25, FWN26]	Schema	Energy research software metadata
Java (Maven POM) [Apa]	Schema	Java build & dependency metadata
maSMP [CGG+24a, CGG+24b]	Schema	Software Management Plans
Python Package Metadata [Pyt]	Schema	Python packaging ecosystem
R Package Metadata [R C]	Schema	R packaging ecosystem
RSD Data Model [NH]	Schema	Research Software Directory metadata
swMath [CDSS24, GS14]	Schema	Mathematical software metadata
Wikidata [Wik, VK14]	Knowledge Graph	Universally valid Linked Open Data

3.3 Mapped Properties

A total number of 68 properties were selected, comprising 60 properties from CodeMeta and 8 additional properties from Schema.org that are not included in CodeMeta. These properties represent a broad range of metadata concepts used to describe research software and serve as the

basis for evaluating the coverage of CodeMeta. These properties include CodeMeta introduced terms (13 properties), as well as properties from Schema.org types, including *schema:Thing* (5 properties), *schema:CreativeWork* (22 properties), *schema:SoftwareSourceCode* (5 properties), *schema:SoftwareApplication* (22 properties), and *schema:WebPage* (1 property). As mentioned above, 60 of these properties are included in CodeMeta. The 8 additional properties from Schema.org include: *codeSampleType*, *applicationSuite*, *availableOnDevice*, *countriesNotSupported*, *countriesSupported*, *featureList*, *screenshot*, and *softwareAddOn*. The complete list of the mapped properties, together with their source (CodeMeta vs. Schema.org), is available in the Appendix A.

To structure this set of properties, we grouped them into 9 categories represented in Table 2. These categories reflect common dimensions of research software metadata and support a structured interpretation of the mapping results. Among these categories, two categories: *Development, Project Status and Management* and *Technical Requirements and Environment* contain the largest number of properties, with 12 and 11 properties, respectively. This indicates that metadata related to these two categories play a central role in describing research software in CodeMeta. Categories such as *Software Classification and Type* and *Publication and Citation*, also show relatively high representation, highlighting the importance of describing both the functional role of software and its scholarly context.

Table 2: Categories of mapped properties

Category	**Description**	**Count**
Core Software Identity	Basic identification and general metadata	7
Source Code and Repository	Where the code lives	4
Distribution and Packaging	How the software is packaged and delivered	5
Authors, Contributors and Maintenance	People and organizations responsible for the software	8
Development, Project Status and Management	Project life cycle and management	12
Technical Requirements and Environment	Dependencies and runtime environment	11
Software Classification and Type	Nature and purpose of the software	9
Documentation, Help and Support	Documentation and supporting resources	4
Publication and Citation	Academic and citation-related metadata	8

3.4 Mapping Strategy

The mappings were developed through a collaborative process involving 15 contributors from the NFDI-RSM-WG. To synchronize the development of the mappings between the contributors, a hackathon (focused on creating and discussing crosswalks for software metadata using CodeMeta as a reference set) was organized by ZB MED Information Centre for Life Sciences[15] in July 2025 and conducted at GESIS-Leibniz Institute for the Social Sciences[16] in July 2025. This effort was followed by regular weekly meetings to continue collaboration and refinement.

[15] https://www.zbmed.de; accessed 20/08/2026
[16] https://www.gesis.org/home; accessed 27/07/2026

For every crosswalk, an initial mapping from the selected schema to the set of 68 reference properties was created by one contributor and subsequently reviewed by three other contributors. Following these reviews, all four contributors jointly analyzed and refined the mappings based on the reviews to ensure consistency and completeness. In cases where reviewers disagreed on a mapping, the discrepancy was first discussed among the four contributors to reach a consensus. If no consensus could be reached, the mapping was escalated to the working group for a final decision. This process ensured that ambiguous or contested mappings were resolved through collective judgement rather than individual interpretation.

Mappings were established on the basis of semantic equivalence between metadata properties. A one-to-one correspondence between the schema properties and the CodeMeta properties was preferred. Where multiple properties from a source schema corresponded to a single CodeMeta property, many-to-one mappings were accepted. In contrast, mappings from a single source property to multiple CodeMeta properties were avoided to preserve semantic clarity.

The crosswalk analysis identified three groups of mappings: **close matches**, **partial matches**, and **unmapped** properties. In addition to close matches, partial mappings were introduced in cases where no fully equivalent CodeMeta property existed but the semantics were sufficiently similar. Three types of partial mapping were distinguished. First, a property from one of the analyzed schemas may be *broader* than the corresponding CodeMeta property, covering a wider range of concepts than CodeMeta intends. Second, a property may be *narrower* than its CodeMeta counterpart, capturing only a specific subset of the concept that CodeMeta defines more generally. Third, two properties may *overlap* without either fully subsuming the other, meaning each captures some aspect of the concept that the other does not.

An example of a broader partial mapping is *platform* (P400) in Wikidata – defined as the platform for which a work was developed or released, or the specific platform version of a software product – partially mapped to *runtimePlatform* in CodeMeta, which is more narrowly defined as a runtime platform or script interpreter dependency. While both properties concern the technical environment of the software, the Wikidata property is broader in scope, encompassing hardware platforms and operating systems beyond runtime environments.

An example of an overlapping partial mapping is *supportingData* in CodeMeta – defined as supporting data for a software application with a range of *DataFeed* – partially mapped to *hasRelevantDataSources* in OntoSoft, defined as any relevant data catalogs that can be used with the software, with a range of *TextEntity*. While both properties concern data associated with the software, they differ in two respects: structurally, CodeMeta expects a machine-readable *DataFeed* object while OntoSoft accepts a plain text string; and semantically, CodeMeta refers specifically to data that supports or belongs to the software, while OntoSoft refers more broadly to any data catalogs that can be used alongside it. Neither property fully subsumes the other, making this a case of overlapping partial mapping.

In all three cases, these mappings were retained to maximize coverage while acknowledging minor differences in meaning. Properties for which no suitable CodeMeta property could be identified were classified as unmapped and subsequently analyzed to identify gaps in CodeMeta's coverage of research software metadata concepts. To better illustrate the mapping process, one representative crosswalk is provided in Appendix A. In addition to the mapping results, this appendix provides the complete list of the 68 reference properties organized by their categories.

4 Results and Analysis

This section reports the results of the crosswalk analysis of 21 software-related metadata schemas, vocabularies, and ontologies against a reference set of 68 properties derived from CodeMeta and Schema.org. Section 4.1 identifies dimensions that are not yet captured by CodeMeta. Section 4.2 presents the metadata concepts that are broadly recognized and consistently represented across the analyzed schemas, reflecting areas of community consensus. Section 4.3 reports the inconsistencies identified across schemas in how shared metadata concepts are represented, including terminological divergence, conflicting semantics, and structural mismatches.

4.1 Coverage and Gaps in CodeMeta

The crosswalk analysis confirms that, of the 60 properties from CodeMeta in the reference set, CodeMeta provides strong coverage for fundamental aspects such as identification, authorship, licensing, and technical requirements. However, crosswalk analysis also revealed several areas where relevant metadata concepts are not represented in CodeMeta. These gaps concern four broad aspects of research software that go beyond CodeMeta's primarily descriptive focus.

Operational behavior and execution characteristics. These aspects are underrepresented in CodeMeta. Properties such as *hasInput* (links a model configuration to the input types expected by it), *hasOutput* (expresses what are the outputs of a model), and *hasConfiguration* (links a model to one of its configurations) [GRGK], describe how software interacts with data and its environment at runtime, aspects that are essential for reproducibility but not covered by CodeMeta.

Research context and intended use. These aspects are not explicitly modeled in CodeMeta. Properties such as *hasPurpose* (objective or main functionality that can be achieved by running the software) [GRGK], and *domain* (research domain for which the software was created) [Fer25], capture the primary objective of the software and the research domain for which it was developed, both of which are important for discoverability and reuse in scientific contexts, and can be considered for the future metadata schema.

Accessibility and ethical considerations. These aspects are absent from CodeMeta. Properties such as *conditionsOfAccess* (conditions that affect the availability of, or method(s) of access to an item) and *ethicalLegalSocial* (considerations with regard to ethical, legal, and social aspects) [CGG+24a], address aspects that are increasingly relevant in the context of open and responsible research software.

Modern software infrastructure and governance. These aspects are not captured in CodeMeta. Properties such as *hasSoftwareImage* (function to link a software with its corresponding container) [GRGK] and *hasInterface* (provides a method of asserting what type of interactions are possible for the class in question) [Sof], reflect contemporary practices such as containerization and interface specification. Also *contactPoint* (relevant contact information for the cataloged resource) [W3C], *developerStructure* (structure of the development team) [Fer25], and *hasAcknowledgements* (string with the people, organizations and other contributors acknowledged by the authors) [GRGK] capture project-level and governance-related information.

It should also be noted that after the crosswalk analysis, three properties: *encoding*, *provider*, and *producer* were removed from the reference set following feedback from the NFDI-RSM-WG and the deRSE workshop, resulting in 65 properties in the final reference set. *Provider*, which describes who hosts or offers software as a service, was considered not central to most research software contexts. *Producer* originates from media domains and overlaps semantically with *publisher*. Also, *encoding* was considered redundant given that the core distribution surface is already covered by *codeRepository*, *downloadUrl*, and *hasSourceCode* properties.

Collectively, these findings indicate that while CodeMeta provides a solid foundation for research software metadata, it does not fully capture the broader research context and operational characteristics of research software. These gaps highlight the need to extend CodeMeta with additional properties, which will be addressed in our future work as mentioned in Section 6.

4.2 Common Metadata Concepts Across Schemas

The crosswalk analysis also reveals a set of metadata concepts that are broadly recognized and consistently represented across the analyzed schemas. These shared concepts reflect the metadata dimensions that the research software community has converged on as essential for describing software artifacts, regardless of domain or format.

Universally represented concepts. A small core of metadata properties appears across nearly all analyzed schemas. The software *name*, *description*, *identifier*, and *version* are the most widely represented concepts, mapped in almost every schema from domain-specific approaches such as biotoolsSchema and ERSmeta to general-purpose vocabularies such as DCAT and DOAP. Similarly, *license* and *url* (the official homepage or landing page of the software) are consistently present, reflecting the centrality of accessibility and legal clarity in software metadata. The *author* or creator of the software is also universally recognized, appearing across all schema types, although its exact definition and structure varies. These universally represented properties closely align with the core identification and attribution requirements of the FAIR4RS principles, suggesting a broad community consensus on the minimum information needed to make research software findable and citable.

Widely represented concepts. A second tier of properties appears across a substantial majority of the analyzed schemas, indicating broad but not universal recognition. *Keywords* and *programmingLanguage* are present in most schemas, highlighting the importance of classification and technical context for software discovery. Repository location (*codeRepository*) and *downloadUrl* are widely represented, reflecting the practical need to locate and access software arti-

facts. Dependencies (*softwareRequirements*), *operatingSystem*, and *datePublished* also appear frequently, indicating that technical requirements and temporal context are generally considered important dimensions of research software metadata. Funding-related properties (*funder*, *funding*) and *referencePublication* appear across a notable subset of schemas, reflecting the scholarly and institutional context of research software, though they are less universally adopted than identification and attribution properties.

Narrowly represented concepts. In contrast, several properties in the reference set are mapped by only a small number of schemas or appear in only one. Properties such as *embargoDate*, *embargoEndDate*, *softwareSuggestions*, *continuousIntegration*, *isSourceCodeOf*, *availableOnDevice*, *codeSampleType*, and *featureList* are present in very few of the analyzed schemas, suggesting that these concepts are either niche, platform-specific, or not yet widely recognized as standard metadata dimensions across the community. Similarly, *applicationSuite*, *encoding*, *countriesSupported*, *countriesNotSupported*, *permissions*, and *softwareAddOn* received no mappings across the 21 schemas, indicating that these properties, while present in the reference set, do not correspond to concepts actively used in the broader research software metadata ecosystem. This finding directly informed the community feedback process, which led to the removal of *encoding* from the final reference set, as discussed in Section 4.1.

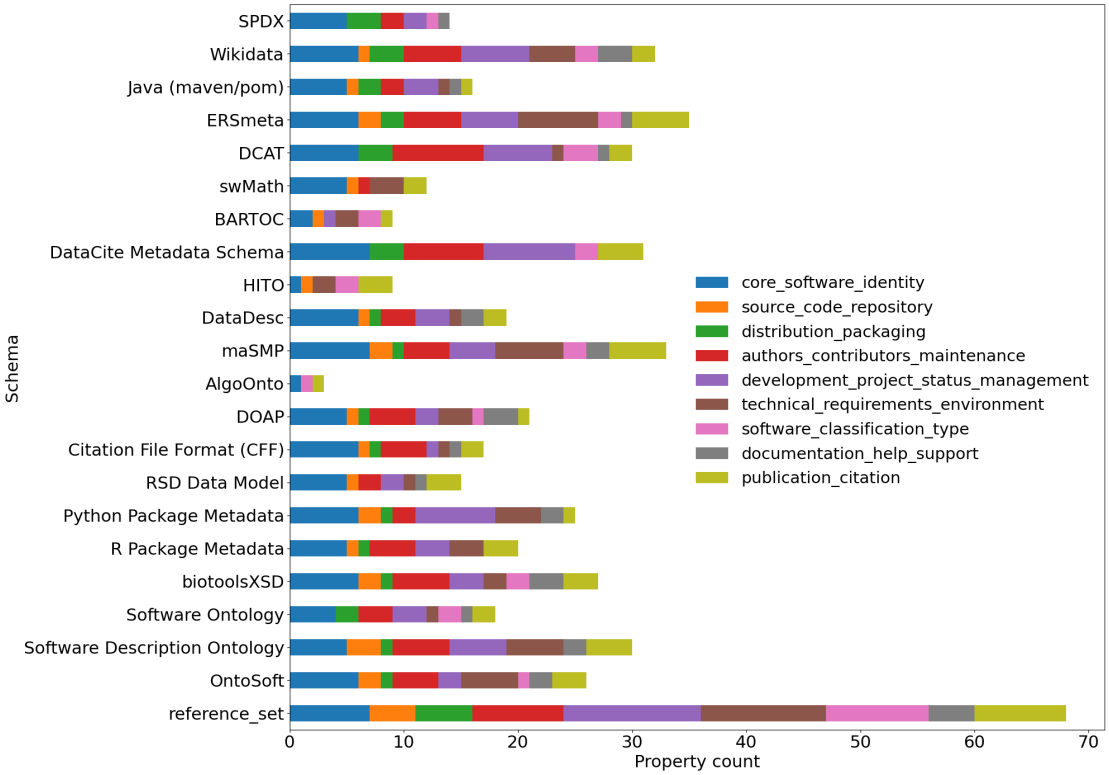


Figure 1: The crosswalk mapping property count information distributed over 9 categories from the 21 metadata approaches in comparison to the reference set.

Distribution across categories. When viewed through the lens of the nine categories defined in Section 3.3, the crosswalk results show that the *Core Software Identity* and *Authors, Contributors and Maintenance* categories achieve the highest average coverage across schemas, confirming that identification and attribution are the most universally addressed dimensions of research software metadata. The *Publication and Citation* and *Technical Requirements and Environment* categories show moderate and consistent coverage, while *Documentation, Help and Support* and *Source Code and Repository* show lower average coverage, suggesting these aspects are addressed more selectively across schemas. The *Development, Project Status and Management* category shows the highest variability, with some properties such as *developmentStatus* and *issueTracker* being widely recognized while others such as *embargoDate* and *continuousIntegration* appear in very few schemas.

Figure 1 further illustrates these findings by showing the per-category property count across all 21 analyzed schemas in comparison to the reference set. The distribution confirms that properties in the *Core Software Identity* category are represented across all 21 metadata approaches, while all other categories are absent in at least one schema, reflecting the uneven coverage landscape described above. These findings provide a consolidated, empirically grounded view of the research software metadata landscape, identifying both the areas of strong community consensus and the dimensions where standardization efforts are still needed.

4.3 Identified Inconsistencies Across Analyzed Schemas

Beyond the common metadata concepts illustrated in the previous section, the crosswalk analysis also revealed a number of inconsistencies in how metadata concepts are represented across the analyzed schemas. These inconsistencies complicate direct alignment and increase the risk of semantic ambiguity or information loss during integration. We identify three main types in this section.

Terminological divergence for equivalent concepts. A recurring pattern across the analyzed schemas is the use of different property names to refer to semantically equivalent or closely related concepts. For example, the concept of a software version, is expressed as *softwareVersion* in CodeMeta, *version* in CFF and biotoolsSchema, *hasVersion* in SDO, *revision* in DOAP, and *hasSoftwareVersion* in OntoSoft. Similarly, the concept of a download location is captured as *downloadUrl* in CodeMeta, *downloadURL* in DCAT, *downloadLocation* in SPDX, and *hasDownloadURL* in SDO. Likewise, the repository location of source code is expressed as *codeRepository* in CodeMeta, *repository* in DOAP, *hasCodeLocation* in OntoSoft, and *source code repository URL* in Wikidata. While these properties are semantically equivalent, their terminological divergence requires explicit mapping rules to enable consistent cross-schema integration.

Conflicting semantics for shared concepts. Several concepts appear across multiple schemas but are defined with subtly different semantics or scope. The concept of software *authorship* is a prominent example: it is represented as *author* in CodeMeta, SDO, ERSmeta and maSMP, *creator* in DCAT, *hasCreator* in OntoSoft, and *developer* in DOAP and Wikidata, yet these properties differ in whether they refer strictly to individuals, organizations, or both. Similarly,

the concept of a *dependency* is captured as *softwareRequirements* in CodeMeta – referring to required software dependencies – as *requires* in DCAT, *hasDependency* in OntoSoft, and *depends on software* in Wikidata, each carrying slightly different semantic scope regarding what constitutes a dependency. The concept of a *contributor* shows similar divergence: CodeMeta uses *contributor*, CFF uses *contributors*, DCAT uses *prov:qualifiedAttribution* with a role qualifier, DataCite uses a structured *contributorType* enumeration, and R Package Metadata encodes contributor roles through a controlled vocabulary within its *Author* field. These differences mean that even apparently equivalent properties cannot always be directly aligned without resolving semantic ambiguities.

Structural mismatches. In addition to terminological and semantic differences, the analyzed schemas also differ substantially in how they structure the same metadata concept. Licensing information illustrates this well. CodeMeta represents *license* as a URL or CreativeWork, CFF splits it into separate *license* and *license-url* fields with different applications, *license* as an SPDX license identifier and *license-url* as the URL of a custom license text under which the work is licensed and that does not have an SPDX identifier. DataCite encodes it within a *rights* element with a controlled *rightsIdentifier* attribute, and Python Package Metadata represents *License-Expression* as a string with valid SPDX license expression syntax, including one or more SPDX License Identifier(s). Similarly, contributor information is a plain text field in some schemas but it is modeled as a structured object (Organization or Person) in CodeMeta, OntoSoft and DataCite. Funding information is captured as a free-text *funding* field in CodeMeta, while as a structured *fundingReference* element with funder name and identifier in DataCite, and as *Funding Information* object in SDO. Such structural differences mean that even semantically equivalent properties require transformation or data loss during mapping.

Overall, these three types of inconsistency highlight that achieving interoperability across research software metadata approaches requires more than simply identifying equivalent properties. It also requires resolving differences in terminology, semantics, and structure. This underscores the importance of a consolidated reference schema as a shared foundation for consistent metadata practices.

5 Limitations

Two stages of our methodology rely on expert judgement and community consensus rather than on a fully deterministic procedure: the selection of schemas to analyze, which was based on nomination by members of the NFDI-RSM-WG, and the mapping decisions, which rely on assessments of semantic equivalence and on the classification of close and partial mappings. As a consequence, a different community or expert group might select a partially different set of schemas and, in borderline cases, arrive at partially different mappings, and therefore at different conclusions about coverage and gaps. This reflects an inherent characteristic of metadata alignment, where correspondences frequently depend on interpretation rather than on formally decidable equivalence, and it is a known limitation of manual matching more generally [RB01, ES13].

We took several steps to constrain this subjectivity and to support verification. Each crosswalk was produced by one contributor and independently reviewed by three others, with disagreements reconciled collectively and escalated to the full working group when no consensus was reached, so that mappings reflect collective rather than individual judgement. All schemas were mapped against a single shared reference set, and partial mappings were explicitly recorded, making the basis of each decision transparent. The analysis also draws on a broad and diverse set of contributors and schemas, reducing the influence of any single perspective.

We also note that since both stages involved input and feedback from the research software community, various impacts on the crosswalk process have been observed. In addition to the potential bias introduced by the selected expert group, it took extra time to recruit and follow the input of the research community compared to a designated working team. Moreover, achieving consensus in the full working group on escalated properties also induced lengthy discussion and negotiation. Nevertheless, it is undeniable that community feedback makes the research process more relevant and grounded to the community, which drives us to overcome the difficulties of this community-driven approach.

6 Conclusion and Future Work

In this paper, we presented a systematic, community-driven crosswalk analysis of 21 research software metadata resources – including metadata schemas, vocabularies, and ontologies – using 68 properties from CodeMeta and Schema.org as a reference point. The crosswalks were collaboratively developed as part of the activities in the NFDI-RSM-WG and refined through community feedback at the deRSE26 Conference workshop.

Our analysis confirms that CodeMeta, already well-known in the research software engineering community, provides a solid foundation for research software metadata, with strong and consistent coverage of core descriptive dimensions such as software identity, authorship, licensing, and technical requirements. At the same time, the analysis revealed gaps in areas such as research context and intended use, operational behavior and execution characteristics, accessibility and ethical considerations, and modern software infrastructure. These gaps highlight the need for targeted extensions to CodeMeta to better support the description and reuse of research software across domains. In addition, the analysis identified inconsistencies such as terminological divergence, where different property names refer to semantically equivalent concepts; conflicting semantics, where shared concept names carry subtly different definitions; and structural mismatches, where equivalent concepts are represented with incompatible data types or granularity. Indeed, achieving interoperability across research software metadata schemas would require more than identifying equivalent properties, as it would also require resolving differences in terminology, semantics, and structure.

Based on our findings, we have identified two lines of future work. First, the core set will be complemented with additional properties, selected from the unmapped properties identified during the crosswalk analysis, to address the gaps that are currently not covered by CodeMeta. The resulting extension will be released as the ConnOSS metadata schema, a unified research software metadata schema currently under development within the ConnOSS project[17]. Second,

[17] https://connoss-project.github.io/; accessed 27/07/2026

a ConnOSS profile, i.e, usage guidelines on the schema, will be released, including obligation levels, usage examples, and cardinality constraints for properties. This will ease adoption by the community and facilitate validation by automatic means. Such a profile layer is essential to move beyond a list of properties toward a usable and actionable standard supporting consistent metadata quality across research software repositories and registries, in alignment with the FAIR4RS principles.

Acknowledgements: First, we would like to thank the NFDI Section (Meta)Data Research Software Metadata Working Group for their valuable contribution in crosswalks creation. Next, we would like to thank the German Federal Government, the German State Governments, and the Joint Science Conference (GWK) for their funding and support as part of the NFDI4Energy consortium within the German National Research Data Infrastructure (NFDI, www.nfdi.de). This work has been partially funded by the German Research Foundation (DFG) with grants No. 561044496 corresponding to the ConnOSS project, No. 501875662 corresponding to NFDI4Immuno, and No. 501865131 corresponding to NFDI4Energy. SD was supported by the Lower Saxony Digital Science Support Space (DS^3) project as part of Hochschule.digital Niedersachsen, funded by zukunft.niedersachsen.

Author contribution statement: Maryam Sefidbari (sefidbari@zbmed.de) was the main author. Aida Jafarbigloo and Lu Gan helped with the writing process. All authors contributed to the review and revision of the paper. The property mapping process was conducted collaboratively within the NFDI-RSM-WG.

Data availability: To support transparency and reuse, all crosswalks produced in this analysis are openly available on Zenodo [SJG+26].

Declaration on the use of AI: LLMs were used to paraphrase and refine sentences. All conceptual contributions, methodological design, crosswalk analysis, and interpretation of results were carried out by the authors, who take full responsibility for the content of this work.

Bibliography

[Apa] Apache Software Foundation. Apache Maven Project Object Model (POM). Accessed: 2026-08-24.
https://maven.apache.org

[BHK+22] M. Barker, N. Hong, D. Katz, A. Lamprecht, C. Martínez-Ortiz, F. Psomopoulos, J. Harrow, L. Castro et al. Introducing the FAIR Principles for research software. *Scientific Data* 9, 2022.
doi:10.1038/s41597-022-01710-x

[bio16] bio.tools. bio.tools XML Schema (biotoolsXSD). 2016. Accessed: 2026-08-24.
https://github.com/bio-tools/biotoolsSchema

[CDSS24] H. Chrapary, W. Dalitz, W. Sperber, M. Schubotz. swMATH: Publication-based indexing of software. In *90 Years of zbMATH*. Pp. 1–5. EMS Press, 2024.
doi:10.4171/ZBL90/7

[CFF+26] L. J. Castro, S. Ferenz, M. Fuhrmans, J. Göpfert, D. Iglezakis, O. Karras, A. Struck. "Research Software Metadata" - Working Group Charter (NFDI section-metadata). Apr. 2026.
doi:10.5281/zenodo.19732950

[CFKS23] J. Cavanagh, J. C. Fliegner, S. Kopper, A. Sautmann. A metadata schema for data from experiments in the social sciences. Policy research working paper 10296, World Bank Group, 2023.
doi:10.1596/1813-9450-10296

[CGG+24a] L. J. Castro, O. Giraldo, L. Geist, N. Quiñones, D. Solanki, D. Rebholz-Schuhmann. machine-actionable Software Management Plan Ontology (maSMP Ontology). 2024.
doi:10.5281/zenodo.10582073

[CGG+24b] L. J. Castro, O. Giraldo, L. Geist, N. Quiñones, D. Solanki, D. Rebholz-Schuhmann. Usage guidance (aka profiles) for the machine-actionable Software Management Plan Ontology. 2024.
doi:10.5281/zenodo.10582121

[CKB+22] N. P. Chue Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, T. Honeyman, A. Struck, A. Lee, A. Loewe, B. van Werkhoven, C. Jones, D. Garijo et al. FAIR Principles for Research Software (FAIR4RS Principles). Technical report, Research Data Alliance, 2022.
doi:10.15497/RDA00068

[Dat] DataCite Metadata Working Group. DataCite Metadata Schema Documentation. Accessed: 2026-08-24.
https://schema.datacite.org/

[Dru17] S. Druskat. Citation File Format (CFF). 2017.
doi:10.5281/zenodo.1117789

[DSC+21] S. Druskat, J. H. Spaaks, N. Chue Hong, R. Haines et al. Citation File Format. 2021.
doi:10.5281/zenodo.1003149

[Dum] E. Dumbill. Description of a Project (DOAP). Accessed: 2026-08-24.
https://github.com/edumbill/doap

[ES13] J. Euzenat, P. Shvaiko. *Ontology Matching*. Springer, Berlin, Heidelberg, 2nd edition, 2013.
doi:10.1007/978-3-642-38721-0

[Fer25] S. Ferenz. ERSmeta. Oct. 2025.
doi:10.5281/zenodo.17465772

[FWN26] S. Ferenz, O. Werth, A. Nieße. Towards a Metadata Schema for Energy Research Software. 2026.
doi:10.48550/arXiv.2601.09456

[GGGD17] L. Garcia, O. Giraldo, A. Garcia, M. Dumontier. Bioschemas: schema.org for the Life Sciences. In *Proceedings of the 10th International Semantic Web Applications and Tools for Life Sciences Conference (SWAT4LS 2017)*. CEUR Workshop Proceedings 2042. 2017.
https://ceur-ws.org/Vol-2042/paper33.pdf

[GGJ17] A. Gray, C. Goble, R. Jimenez. Bioschemas: From Potato Salad to Protein Annotation. In *ISWC 2017 Posters & Demonstrations and Industry Tracks*. 2017. The 16th International Semantic Web Conference 2017, ISWC 2017.
https://ceur-ws.org/Vol-1963/paper579.pdf

[GGMR16] Y. Gil, D. Garijo, S. Mishra, V. Ratnakar. OntoSoft: A Distributed Semantic Registry for Scientific Software. In *2016 IEEE 12th International Conference on E-Science (e-Science)*. Pp. 331–336. 2016.
doi:10.1109/eScience.2016.7870916

[GOK+19] D. Garijo, M. Osorio, D. Khider, V. Ratnakar, Y. Gil. OKG-Soft: An Open Knowledge Graph with Machine Readable Scientific Software Metadata. In *2019 15th International Conference on eScience (eScience)*. Pp. 349–358. Sept. 2019.
doi:10.1109/eScience.2019.00046

[GRG15] Y. Gil, V. Ratnakar, D. Garijo. OntoSoft: Capturing Scientific Software Metadata. In *Proceedings of the 8th International Conference on Knowledge Capture*. K-CAP 2015, pp. 1–4. Association for Computing Machinery, New York, NY, USA, 2015.
doi:10.1145/2815833.2816955

[GRGK] D. Garijo, V. Ratnakar, Y. Gil, D. Khider. The Software Description Ontology. https://w3id.org/okn/o/sd/1.8.0. Accessed: 2026-08-24.

[GS14] G.-M. Greuel, W. Sperber. swMATH – an Information Service for Mathematical Software. In Hong and Yap (eds.), *Mathematical Software – ICMS 2014*. Pp. 691–701. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. doi:10.1007/978-3-662-44199-2$_1$03

[GWB+25] O. Gustafsson, S. Wilkinson, F. Bacall, S. Soiland-Reyes et al. WorkflowHub: a registry for computational workflows. *Scientific Data* 12, 2025. doi:10.1038/s41597-025-04786-3

[Hab18] T. Habermann. Mapping ISO metadata standards to codemeta. 2018. doi:10.7287/peerj.preprints.27153v1

[Hab19] T. Habermann. Mapping ISO 19115-1 geographic metadata standards to CodeMeta. *Peerj Computer Science* 5:e174, 2019. doi:10.7717/peerj-cs.174

[HMS+09] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, G. Wilson. How Do Scientists Develop and Use Scientific Software? *Proceedings of the ICSE Workshop on Software Engineering for Computational Science and Engineering*, 2009. doi:10.1109/SECSE.2009.5069155

[IIR+21] J. Ison, H. Ienasescu, E. Rydza, P. Chmura, K. Rapacki et al. biotoolsSchema: A Formalized Schema for Bioinformatics Software Description. *GigaScience* 10(1):giaa157, Jan. 2021. doi:10.1093/gigascience/giaa157

[JAD+] F. Jahn, E. Ammenwerth, V. Dornauer, K. Höffner, M. Bindel, T. Karopka, A. Winter. Health IT Ontology (HITO). Accessed: 2026-08-24. https://hitontology.github.io/ontology/

[JBC+] M. B. Jones, C. Boettiger, A. Cabunoc Mayes, A. Smith et al. CodeMeta: an exchange schema for software metadata. Accessed: 2026-08-24. https://w3id.org/codemeta/3.0

[KGK+] P. Kuckertz, J. Göpfert, O. Karras, D. Neuroth, J. Schönau, R. Pueblas, S. Ferenz. DataDesc: Data Description Schema. Accessed: 2026-08-24. https://github.com/FZJ-IEK3-VSA/DataDesc

[KGK+24] P. Kuckertz, J. Göpfert, O. Karras, D. Neuroth, J. Schönau, R. Pueblas, S. Ferenz et al. DataDesc: A Framework for Creating and Sharing Technical Metadata for Research Software Interfaces. *Patterns* 5(11):101064, 2024. doi:10.1016/j.patter.2024.101064

[Led] A. Ledl. BARTOC: Basic Register of Thesauri, Ontologies & Classifications. Accessed: 2026-08-24. https://bartoc.org

[LGK+20] A.-L. Lamprecht, L. Garcia, M. Kuzak et al. Towards FAIR Principles for Research Software. *Data Science* 3(1):37–59, 2020.
doi:10.3233/DS-190026

[Lin] Linux Foundation. SPDX Specification. Accessed: 2026-08-24.
https://spdx.dev/specifications

[Mat] Mathematical Research Data Initiative (MaRDI). AlgoOnto: Algorithm Ontology. Accessed: 2026-08-24.
https://mathalgodb.mardi4nfdi.de/static/widoco/v1/index-en.html

[MBL+14] J. Malone, A. Brown, A. L. Lister, J. Ison, D. Hull, H. Parkinson, R. Stevens. The Software Ontology (SWO): A Resource for Reproducibility in Biomedical Data Analysis, Curation and Digital Preservation. *Journal of Biomedical Semantics* 5(1):25, June 2014.
doi:10.1186/2041-1480-5-25

[MDR+24] M. Moir, M. Dunaiski, C. Riley, A. Winters, V. Naranbhai, F. Made, S. Karim, I. Katz et al. Review of open-source software for developing heterogeneous data management systems for bioinformatics applications. *Bioinformatics Advances* 5, 2024.
doi:10.1093/bioadv/vbaf168

[NH] Netherlands eScience Center, Helmholtz Federation. Research Software Directory Data Model. Accessed: 2026-08-24.
https://research-software-directory.org

[Ont] OntoSoft. OntoSoft Ontology. Accessed: 2026-08-24.
https://www.ontosoft.org/

[PSMH23] B. Patel, S. Soundarajan, H. Menager, Z. Hu. Making Biomedical Research Software FAIR: Actionable Step-by-step Guidelines with a User-support Tool. *Scientific Data* 10, 2023.
doi:10.1038/s41597-023-02463-x

[Pyt] Python Packaging Authority. Python Packaging Metadata. Accessed: 2026-08-24.
https://packaging.python.org/specifications/

[R C] R Core Team. R Package Metadata. Accessed: 2026-08-24.
https://cran.r-project.org

[RB01] E. Rahm, P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal* 10(4):334–350, 2001.
doi:10.1007/s007780100057

[Sci25] SciTePress. Proposal for a common conceptual schema for metadata integration and reuse across diverse scientific domains. In *Proceedings of the International Conference on Data Integration and Metadata Reuse*. Pp. 277–285. 2025.
doi:10.5220/0013291500003929

[SJG+26] M. Sefidbari, A. Jafarbigloo, L. Gan, O. Werth, S. Ferenz, L. J. Castro et al. Crosswalks of Research Software Metadata Schemas to CodeMeta. 2026.
doi:10.5281/zenodo.21935688

[Sof] Software Ontology. Software Ontology (SWO). Accessed: 2026-08-24.
https://bioportal.bioontology.org/ontologies/SWO

[SSC+22] S. Soiland-Reyes, P. Sefton, M. Crosas, L. Castro, F. Coppens, J. Fernández, D. Garijo et al. Packaging research artefacts with RO-Crate. *Data Science* 5:97–138, 2022.
doi:10.3233/ds-210053

[VAF+23] T. Vuillaume, M. Al-Turany, M. Füßling, T. Gal, E. García, K. Graf, G. Hughes, M. Kettenis et al. The ESCAPE Open-source Software and Service Repository. *Open Research Europe* 3:46, 2023.
doi:10.12688/openreseurope.15692.2

[VK14] D. Vrandečić, M. Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM* 57(10):78–85, 2014.
doi:10.1145/2629489

[W3C] W3C. Data Catalog Vocabulary (DCAT). Accessed: 2026-08-24.
https://www.w3.org/TR/vocab-dcat-2/

[WAB+25] S. Wilkinson, M. Aloqalaa, K. Belhajjame, M. Crusoe et al. Applying the FAIR Principles to computational workflows. *Scientific Data* 12, 2025.
doi:10.1038/s41597-025-04451-9

[Wik] Wikimedia Foundation. Wikidata. Accessed: 2026-08-24.
https://www.wikidata.org

[WRV+23] M. Wu, S. M. Richard, C. Verhey, L. J. Castro, B. Cecconi, N. Juty. An analysis of crosswalks from research data schemas to Schema.org. *Data Intelligence* 5(1):100–121, 2023.
doi:10.1162/dint_a_00186

A Sample Crosswalk: CodeMeta to OntoSoft

Table 3 presents the crosswalk mapping between the reference set properties and OntoSoft ontology, serving as a representative example of the mapping process. We selected CodeMeta – OntoSoft crosswalk as the representative example because it illustrates the full mapping methodology particularly well, as it contains instances of all three partial mapping types described in Section 3.4.

Table 3: Crosswalk mapping between CodeMeta and OntoSoft

Reference Property	**Source**	**OntoSoft Property**	**Mapping Type**
Core Software Identity			
name	CodeMeta	hasName	Close match
description	CodeMeta	hasShortDescription	Close match
identifier	CodeMeta	hasUniqueId	Close match
softwareVersion	CodeMeta	hasSoftwareVersion	Close match
url	CodeMeta	hasProjectWebsite	Partial match
sameAs	CodeMeta	—	Unmapped
keywords	CodeMeta	hasDomainKeywords	Close match
Source Code and Repository			
codeRepository	CodeMeta	hasCodeLocation	Close match
hasSourceCode	CodeMeta	—	Unmapped
isSourceCodeOf	CodeMeta	—	Unmapped
buildInstructions	CodeMeta	hasInstallationInstructions	Close match
Distribution and Packaging			
downloadUrl	CodeMeta	—	Unmapped
installUrl	CodeMeta	hasExecutableLocation	Close match
fileSize	CodeMeta	—	Unmapped
encoding	CodeMeta	—	Unmapped
encodingFormat	CodeMeta	—	Unmapped
Authors, Contributors and Maintenance			
author	CodeMeta	hasCreator	Close match
contributor	CodeMeta	hasMajorContributor	Close match
maintainer	CodeMeta	hasEmailContact	Partial match
editor	CodeMeta	—	Unmapped
publisher	CodeMeta	hasPublisher	Close match
copyrightHolder	CodeMeta	—	Unmapped
producer	CodeMeta	—	Unmapped
provider	CodeMeta	—	Unmapped
Development, Project Status and Management			
developmentStatus	CodeMeta	—	Unmapped
issueTracker	CodeMeta	—	Unmapped
continuousIntegration	CodeMeta	—	Unmapped
releaseNotes	CodeMeta	—	Unmapped

Reference Property	Source	OntoSoft Property	Mapping Type
dateCreated	CodeMeta	—	Unmapped
dateModified	CodeMeta	—	Unmapped
datePublished	CodeMeta	hasVersionReleaseDate	Close match
embargoDate	CodeMeta	—	Unmapped
embargoEndDate	CodeMeta	—	Unmapped
funding	CodeMeta	hasFundingSources	Close match
funder	CodeMeta	—	Unmapped
sponsor	CodeMeta	—	Unmapped
Technical Requirements and Environment			
programmingLanguage	CodeMeta	hasImplementationLanguage	Close match
runtimePlatform	CodeMeta	—	Unmapped
operatingSystem	CodeMeta	supportsOperatingSystem	Close match
softwareRequirements	CodeMeta	hasDependency	Close match
softwareSuggestions	CodeMeta	—	Unmapped
processorRequirements	CodeMeta	—	Unmapped
memoryRequirements	CodeMeta	requiresAverageMemory	Close match
storageRequirements	CodeMeta	—	Unmapped
permissions	CodeMeta	—	Unmapped
availableOnDevice	Schema.org	—	Unmapped
supportingData	CodeMeta	hasRelevantDataSources	Partial match
Software Classification and Type			
applicationCategory	CodeMeta	hasSoftwareCategory	Close match
applicationSubCategory	CodeMeta	—	Unmapped
applicationSuite	Schema.org	—	Unmapped
targetProduct	CodeMeta	—	Unmapped
codeSampleType	Schema.org	—	Unmapped
featureList	Schema.org	—	Unmapped
softwareAddOn	Schema.org	—	Unmapped
hasPart	CodeMeta	—	Unmapped
isPartOf	CodeMeta	—	Unmapped
Documentation, Help and Support			
softwareHelp	CodeMeta	hasSoftwareSupport	Close match
readme	CodeMeta	hasDocumentation	Partial match
relatedLink	CodeMeta	—	Unmapped
screenshot	Schema.org	—	Unmapped
Publication and Citation			
citation	CodeMeta	hasCitationLocation	Close match
referencePublication	CodeMeta	hasPreferredCitation	Close match
review	CodeMeta	—	Unmapped
license	CodeMeta	hasLicense	Close match
copyrightYear	CodeMeta	—	Unmapped
countriesSupported	Schema.org	—	Unmapped
countriesNotSupported	Schema.org	—	Unmapped

Reference Property	Source	OntoSoft Property	Mapping Type
isAccessibleForFree	CodeMeta	—	Unmapped
OntoSoft properties without a mapping from CodeMeta			
—	—	hasAdopter	Unmapped
—	—	hasAverageRunTime	Unmapped
—	—	hasBenchmarkInformation	Unmapped
—	—	hasCitationText	Unmapped
—	—	hasCommitmentOfSupport	Unmapped
—	—	hasCompatibleSoftware	Unmapped
—	—	hasCompositionDescription	Unmapped
—	—	hasFileId	Unmapped
—	—	hasFileType	Unmapped
—	—	hasInput	Unmapped
—	—	hasInputParameter	Unmapped
—	—	hasInteroperableSoftware	Unmapped
—	—	hasOtherImplementationDetails	Unmapped
—	—	hasOutput	Unmapped
—	—	hasPersonIdentifier	Unmapped
—	—	hasRating	Unmapped
—	—	hasSalientQualities	Unmapped
—	—	hasSoftwareCommunity	Unmapped
—	—	hasTestData	Unmapped
—	—	hasTestDataDescription	Unmapped
—	—	hasTestDataLocation	Unmapped
—	—	hasTestInstructions	Unmapped
—	—	hasUsageInformation	Unmapped
—	—	hasUseLimitations	Unmapped
—	—	hasUseStatistics	Unmapped
—	—	hasUsesAndAssumptions	Unmapped
—	—	hasWorkflowDescription	Unmapped
—	—	similarSoftware	Unmapped
—	—	supersededBy	Unmapped
—	—	supersedes	Unmapped