Formalization and Implementation of Type-based FAIR Digital Object Operations for Machine-actionable Data

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Nicolas Fabian Blumenröhr

Tag der mündlichen Prüfung: 12. Mai 2026

1. Referent: Prof. Dr. Achim Streit

2. Referent: Prof. Dr. Philipp Wieder



This document is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0): https://creativecommons.org/licenses/by/4.0/deed.en

Disclosure

During the preparation of this work, I used ChatGPT (GPT series 4 and 5) and GitHub Copilot for writing support (style, grammar, and spelling) and for software development support of the testbed (debugging suggestions, code review, and refactoring). All tool outputs were reviewed and edited by me as needed and I take full responsibility for the final content and implementation.

Abstract

Modern research increasingly depends on data from many sources, disciplines, and infrastructures. Yet in practice, data reuse is often slowed down by missing context, incompatible formats, and manual, project-specific integration work. The FAIR principles address this problem by providing guidelines for data and metadata that are findable, accessible, interoperable, and reusable—not only for humans, but in ways that also enable automated processing by machines.

FAIR Digital Objects (FDOs) are a concrete approach to putting these principles into practice by treating data as well-defined digital entities with persistent identifiers, structured metadata, and predictable interfaces. Existing FDO-related initiatives have made strong progress on identifier schemes, metadata conventions, and packaging mechanisms. However, they often handle two key aspects only implicitly or inconsistently: how FDOs are typed and how executable operations are systematically associated with those types. As a result, there is still no uniform, technology-agnostic framework that clearly specifies how operations are declared, discovered, and executed for FDOs. To overcome these limitations, this thesis develops a solution for type-based FDO operations to enable machine-actionable data.

First, a rigorously formalized FDO core data model is defined that specifies what an FDO is, which components it contains, and which rules it must follow. The model is designed to be domain-independent and is demonstrated in two different areas, i.e., energy research and the digital humanities, along with a comparison to other FDO modeling approaches.

Building on this foundation, a systematic method for linking operations to FDOs through a type system is introduced. Three complementary typing mechanisms are defined that turn FDO typing into a rule-based scheme for associating the right operations with the right objects, and their strengths and trade-offs are evaluated.

Finally, the dissertation combines these ideas into an implementation approach that makes operations discoverable and executable in practice. Operations are modeled as operating FDOs that can be associated to target FDOs using explicit rules, and are integrated as part of a technology-agnostic framework that enables metadata- and data-level processing. A testbed implementation demonstrates the approach in both application domains considered earlier and shows how it improves interoperability and reuse in line with the FAIR principles.

Overall, the dissertation provides a coherent specification of FDO structure, typing, and operation execution, showing how FDOs can support automated processing and enable more interoperable, reusable research data across diverse research domains.

Zusammenfassung

Die moderne Forschung stützt sich zunehmend auf Daten aus vielen Quellen, Disziplinen und Infrastrukturen. In der Praxis wird die Wiederverwendung von Daten jedoch oft durch fehlenden Kontext, inkompatible Formate und manuelle, projektspezifische Integrationsarbeiten verlangsamt. Die FAIR-Prinzipien gehen dieses Problem an, indem sie Richtlinien für Daten und Metadaten bereitstellen, die auffindbar, zugänglich, interoperabel und wiederverwendbar sind – nicht nur für Menschen, sondern auch in einer Weise, die eine automatisierte Verarbeitung durch Maschinen ermöglicht.

FAIR Digital Objects (FDOs) sind ein konkreter Ansatz zur Umsetzung dieser Prinzipien, indem Daten als klar definierte digitale Einheiten mit persistenten Identifikatoren, strukturierten Metadaten und vorhersehbaren Schnittstellen behandelt werden. Bestehende FDO-bezogene Initiativen haben große Fortschritte bei Identifikationsschemata, Metadatenkonventionen und Datenverpackungsmechanismen erzielt. Allerdings behandeln sie zwei wichtige Aspekte oft nur implizit oder inkonsistent: wie FDOs typisiert werden und wie ausführbare Operationen systematisch mit diesen Typen verknüpft werden. Infolgedessen gibt es noch immer kein einheitliches, technologieunabhängiges Framework, das klar festlegt, wie Operationen für FDOs deklariert, entdeckt und ausgeführt werden. Um diese Einschränkungen zu überwinden, wird in dieser Dissertation eine Lösung für typbasierte FDO-Operationen zur Ermöglichung maschinenverarbeitbarer Daten entwickelt.

Zunächst wird ein streng formalisiertes FDO-Kerndatenmodell definiert, das festlegt, was ein FDO ist, welche Komponenten es enthält und welchen Regeln es folgen muss. Das Modell ist domänenunabhängig konzipiert und wird in zwei verschiedenen Bereichen, nämlich der Energieforschung und den digitalen Geisteswissenschaften, demonstriert und mit anderen FDO Modellierungsansätzen verglichen.

Auf dieser Grundlage wird eine systematische Methode zur Verknüpfung von Operationen mit FDOs über ein Typsystem eingeführt. Es werden drei sich ergänzende Typisierungsmechanismen definiert, welche die FDO-Typisierung in ein regelbasiertes Schema zur Zuordnung der richtigen Operationen zu den richtigen Objekten umwandeln und die Stärken und Kompromisse der Mechanismen werden evaluiert.

Schließlich kombiniert die Dissertation diese Ideen zu einem Implementierungsansatz, der Operationen in der Praxis auffindbar und ausführbar macht. Operationen werden als operierende FDOs modelliert, die mithilfe expliziter Regeln mit Ziel-FDOs assoziiert werden können, und als Teil eines technologieagnostischen Frameworks integriert, das die Verarbeitung auf Metadaten- und Datenebene ermöglicht. Eine Testbed-Implementierung demonstriert den Ansatz in den beiden vorher betrachteten Anwendungsbereichen und

zeigt, wie dieser die Interoperabilität und Wiederverwendbarkeit im Einklang mit den FAIR-Prinzipien verbessert.

Insgesamt liefert die Dissertation eine kohärente Spezifikation der FDO-Struktur, Typisierung und Operationsausführung und zeigt, wie FDOs die automatisierte Verarbeitung unterstützen und interoperablere, wiederverwendbare Forschungsdaten über verschiedene Forschungsbereiche hinweg ermöglichen können.

Acknowledgements

This dissertation was made possible by the support of several people, to whom I would like to express my sincere gratitude.

First and foremost, I would like to express my deep gratitude to my supervisor, Prof. Achim Streit, who gave me the opportunity to do my doctorate at his chair. Over the years, he has supported me with professional and strategic advice and given me space to develop. I would also like to thank my second reviewer, Prof. Philipp Wieder, for the great interest in my work.

I would like to express my special thanks to Rossella Aversa, Thomas Jejkal, and Rainer Stotzka, who supervised my work and supported me tirelessly in different stages of my research.

Thanks also to the entire Data Exploitation Methods department that supported me throughout the years and gave me the opportunity to develop, present and discuss my ideas in a constructive and comfortable environment.

Special thanks go to Philipp Ost, Maximilian Inckmann, Felix Kraus, Andreas Pfeil and Germaine Götzelmann who supported me to refine my work in many fruitful discussions and proof readings.

I would also like to thank those people with whom I had the opportunity to participate in common research projects, especially Markus Götz, Jana Böhm, Sven Bingert and Peter Wittenburg.

Last but not least, I want to thank my family who supported and backed me all the time, especially my dear Ola, who stood close by me from the beginning to the end of this journey, with all its ups and downs, thank you.

List of Publications

During my doctoral studies, the following papers with me as the first author were published (or accepted with pending publication) in peer-reviewed venues:

- **Nicolas Blumenröhr**, Neil MacKinnon, and Rossella Aversa. "FAIR Data Management Workflow for MRI Data". In: Proceedings of the 3rd Workshop on Metadata and Research (objects) Management for Linked Open Science (DaMaLOS 2023) - Co-located with Extended Semantic Web Conference (ESWC). May 2023, p. 5349. DOI: 10.4126/FRL01-006444992.
- **Nicolas Blumenröhr** and Rossella Aversa. "From implementation to application: FAIR digital objects for training data composition". In: Research Ideas and Outcomes Vol 9, e108706, 2023. ISSN: 2367-7163. DOI: 10.3897/rio.9.e108706.
- **Nicolas Blumenröhr**, Philipp-Joachim Ost, Felix Kraus and Achim Streit, "FAIR Digital Objects for the Realization of Globally Aligned Data Spaces". In: 2024 IEEE International Conference on Big Data (BigData). Dec. 2024, pp. 374–383. DOI: 10.1109/BigData62323.2024.10825796.
- **Nicolas Blumenröhr**, Jana Böhm, Philipp Ost, Marco Kulüke, Peter Wittenburg, Christophe Blanchi, Sven Bingert and Ulrich Schwardmann, "A Comparative Analysis of Modeling Approaches for the Association of FAIR Digital Objects Operations". In: Data Science Journal Vol 24, p. 22, 2025. ISSN: 1683-1470. DOI: 10.5334/dsj-2025-022.
- **Nicolas Blumenröhr** and Felix Kraus. "Leveraging Large Language Models for Processing and Evaluating FAIR Digital Objects". In: Proceedings of the 2nd International Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2025) - Co-located with the 22nd Extended Semantic Web Conference (ESWC 2025). Ed. by Georg Rehm et al. Vol. 3977. CEUR Workshop Proceedings. Portorož, Slovenia: CEUR, Jun. 2025. DOI: 10.5445/IR/1000183869.
- **Nicolas Blumenröhr**, Philipp Ost, Rossella Aversa, Maximilian Inckmann, Felix Kraus, and Achim Streit, "FDO-Ops: A Model for Machine-actionable FAIR Digital Objects to Enhance Interoperability" Data Intelligence Journal, 2026. DOI: 10.3724/2096-7004.di.2025.0345. Print ISSN: 2096-7004, Online ISSN: 2641-435X.

I have also contributed to the following works during my studies that are related to the dissertation:

- Maximilian Inckmann, **Nicolas Blumenröhr**, and Rossella Aversa, "Towards Machine-actionable FAIR Digital Objects with a Typing Model that Enables Operations", in 2025 IEEE International Conference on eScience (eScience), Chicago, IL, USA: IEEE, Sep. 2025, pp. 111–120, ISBN: 979-8-3315-9145-8. DOI: 10.1109/eScience65000.2025.00022.
- Zoe Mayer, James Kahn, Markus Götz, Yu Hou, Tobias Beiersdörfer, **Nicolas Blumenröhr**, Rebekka Volk, Achim Streit and Frank Schultmann, "Thermal Bridges on Building Rooftops", Scientific Data, vol. 10, no. 1, p. 268, 2023, ISSN: 2052-4463. DOI: 10.1038/s41597-023-02140-z.
- Felix Kraus, **Nicolas Blumenröhr**, Germaine Götzelmann, Danah Tonne, and Achim Streit, "A Gold Standard Benchmark Dataset for Digital Humanities". In: Proceedings of the 19th International Workshop on Ontology Matching - Co-located with the 23rd International Semantic Web Conference (ISWC 2024). Ed. by Ernesto Jiménez-Ruiz et al. Vol. 3897. CEUR Workshop Proceedings. Baltimore, USA: CEUR, Nov. 2024, pp. 1–17. DOI: 10.5445/IR/1000178023.

Contents

Disclosure . . . . . **ii**

Abstract . . . . . **iii**

Zusammenfassung . . . . . **iv**

Acknowledgements . . . . . **vi**

List of Publications . . . . . **vii**

List of Figures . . . . . **xii**

List of Tables . . . . . **xiii**

List of Algorithms . . . . . **xiv**

Listings . . . . . **xv**

List of Acronyms and Notation Style . . . . . **xvi**

1. Introduction **1**
1.1. Motivation 1
1.2. Research Questions 2
1.3. Contributions and Outline 2

2. Background **4**
2.1. Object-Oriented Design and Type Systems 4
2.1.1. Abstraction and Encapsulation 5
2.1.2. Subtyping, Polymorphism and Inheritance 5
2.1.3. Forms of Type Systems 5
2.2. Knowledge Representation 6
2.2.1. Entity-Relationship Model 6
2.2.2. Knowledge Organization Systems 7
2.2.3. Semantic Web 8
2.3. Research (Meta)data Management 9
2.3.1. Metadata 9
2.3.2. FAIR principles 10
2.3.3. Interoperability 12

2.3.4. Data Spaces . . . 13
2.4. Digital Objects . . . 14
2.4.1. Digital Object Architecture . . . 14
2.4.2. Digital Objects Core Data Model . . . 15
2.4.3. Digital Object Typing and Kernel Information . . . 17
2.4.4. From Digital Objects to FAIR Digital Objects . . . 20

3. Formalizing a FAIR Digital Object Core Data Model . . . 21
3.1. Introduction . . . 21
3.2. Related Work . . . 22
3.3. Methodology . . . 23
3.3.1. Alignment of Data Spaces through FAIRification . . . 23
3.3.2. Introduction of a Meta-level . . . 25
3.3.3. FDO Data Model Formalization . . . 26
3.3.4. Base Services and Infrastructure . . . 34
3.3.5. Example Datasets . . . 35
3.4. Evaluation and Results . . . 35
3.4.1. Fulfillment of Meta-FAIRification Characteristics . . . 36
3.4.2. Model Implementation and Proof of Concept . . . 38
3.4.3. Comparative Analysis with Related Works . . . 42
3.5. Discussion . . . 46

4. Typing Mechanisms for Associating Operations with FAIR Digital Objects . . . 48
4.1. Introduction . . . 48
4.2. Related Work . . . 49
4.3. Methodology . . . 50
4.3.1. Typing Mechanisms in Analogy to OOP . . . 50
4.3.2. Modeling Associations between FDOs and Operations . . . 52
4.3.3. Quality Indicators and Measures . . . 55
4.4. Evaluation and Results . . . 56
4.4.1. Assessment of Typing Mechanisms . . . 56
4.4.2. Quantification of Quality Measures . . . 63
4.4.3. Comparison of Measures . . . 66
4.5. Discussion . . . 69

5. The FDO-Ops Framework for Machine-Actionable FAIR Digital Objects . . . 71
5.1. Introduction . . . 71
5.2. Related Work . . . 72
5.2.1. Assessment of Interoperability . . . 72
5.2.2. Knowledge Organization Systems and Executable Specifications . . . 73
5.2.3. Machine-actionable FAIR Digital Objects . . . 74
5.3. Methodology . . . 74
5.3.1. Specifying Operations . . . 74
5.3.2. Embedding Typing Mechanisms in DTR Model . . . 76
5.3.3. Representing Operations Using FDOs . . . 77

5.3.4. Cross-Walk Between DTR Model and Operation FDOs 83
5.3.5. Process Model 85
5.4. Model Implementation 87
5.4.1. Employment of Next-generation DTR Prototype 88
5.4.2. Digital Object Interface Protocol 88
5.4.3. FDO-Ops Framework Architecture 89
5.4.4. FDO-Ops Framework Sequence 91
5.4.5. Example Data 92
5.5. Evaluation and Results 94
5.5.1. IDORIS Graph Model 94
5.5.2. Applied Use Cases for FDO-Ops Framework 96
5.5.3. Quantification of Impact by FDO-Ops 100
5.5.4. Enhancement of Interoperability Levels by FDO-Ops 102
5.6. Discussion 103
5.6.1. Model and Implementation Design Decisions 103
5.6.2. Evaluation and Results Findings 105

6. Conclusions and Future Work 107
6.1. Summary 107
6.2. Future Work 108

Bibliography 110

A. Appendix 129
A.1. Background Chapter Appendix 129
A.2. Chapter 3 Appendix 131
A.2.1. FDO Graph for Example Data 131
A.2.2. Original Handle Record of Example Data FDO 132
A.2.3. Original Handle Records Used in the Comparative Analysis . . . 132
A.3. Chapter 5 Appendix 133
A.3.1. Testbed Code and Records Availability 133
A.3.2. Algorithm Examples 134
A.3.3. DOIP/HTTP Requests 135
A.3.4. Conflict Rules in the FDO-Graph 136
A.3.5. IDORIS' Neo4j Graph 137
A.3.6. Cross-walk Example for Get Related Terms Operation 140
A.3.7. Sandbox Information Record of Operation FDO "Get Related Terms" 142
A.3.8. Applied Use Case Examples Listings 143
A.4. Proof Trees 144
A.4.1. Basic Type System 144
A.4.2. Record Typing 147
A.4.3. Profile Typing 149
A.4.4. Attribute Typing 152

List of Figures

2.1. A minimal ER-diagram. 7
2.2. A minimal conceptual RDF graph of two nodes connected by an edge. 8
2.3. Core Data Model of Digital Objects. 15
2.4. PID record concept for Digital Objects. 16
2.5. The conceptual property-type-profile model of PITs. 18
2.6. The high-level architecture for PITs. 18

3.1. Venn diagram for FAIRified data spaces. 24
3.2. Meta-level of FAIRified data spaces. 25
3.3. The conceptual FDO core data model. 26
3.4. Model semantics validation activity diagram. 31
3.5. ER principle in FDO-FDO relationships for data and metadata. 33
3.6. Example of a PID hypergraph from which an FDO digraph is constructed. . . 33
3.7. The conceptual model of PID triples based on the FDO's ER modeling. 34
3.8. Exemplary FDOs represented according to the formalized core data model. . 41

4.1. The conceptual typing mechanism for FDOs. 51
4.2. ER and exemplary graph representations of association models. 54

5.1. Advanced DTR model with FDO typing mechanisms. 77
5.2. Example of a PID hypergraph from which an Operation-Target FDO digraph is constructed. 82
5.3. The FDO-Ops Process Model for a FAIR Digital Object. 86
5.4. The FDO-Ops service architecture depicted as UML component diagram. . . 89
5.5. UML sequence diagram of the FDO-Ops framework. 91
5.6. Data flow diagram using the FDO-Ops framework. 92
5.7. The Neo4j graph of the TBBR dataset. 93
5.8. The Neo4j graph of the Operation- and Target FDOs. 94
5.9. Remodeled extract from IDORIS' Neo4j graph. 95

A.1. The original Handle Record of FDO f_3. 132
A.2. The original PIDINST Handle Record. 132
A.3. The original DARIAH Handle Record. 133
A.4. The original DES Handle Record. 133
A.5. Original Neo4j graph in IDORIS I. 138
A.6. Original Neo4j graph in IDORIS II. 139

List of Tables

1. General notation and mathematical symbols used throughout the dissertation. xviii

2.1. General overview of metadata types. 10

3.1. Typed properties of the extended (*) Helmholtz KIP. 31
3.2. Connecting Meta-FAIRification Characteristics (MFC) to FAIR criteria. 37
3.3. Comparison of existing FDO modeling approaches based on our FDO core data model traits. 42

4.1. Overview of measures between RT, PT, and AT approaches and corresponding metrics. 67

5.1. FDO operation phases and categories. 76

A.1. Overview of FAIR principles criteria. 129
A.2. Overview of FAIR4RS Principles criteria. 130
A.3. Overview of updated FAIR principles criteria. 130

List of Algorithms

1. Schema Extraction Algorithm. 19
2. Algorithm for asserting the association between an operation and its Target FDO. 80
3. Algorithm for execution protocol mapping. 84

Listings

5.1. Search_FDO_Records Operation Overview. 97
5.2. List_ops Operation Overview. 97
5.3. Get_Related_FDOs Operation Overview. 97
5.4. Get_Related_Terms Operation Overview. 98
5.5. Convert_Numpy_to_PNG Operation overview. 100

A.1. Mapping of condition object C_o to record set R_f. 134
A.2. Mapping of execution-protocol e_o and FDO record R_{f_t} to an execution map M. 134
A.3. Operation FDO Information Record for "Get Related Terms." 142
A.4. Validate_SKOS_RDF Operation overview. 143

List of Acronyms and Notation Style

API Application Programming Interface

AT Attribute Typing

CRUD Create, Read, Update, Delete

DO Digital Object

DOA Digital Object Architecture

DOI Digital Object Identifier

DOIP Digital Object Interface Protocol

DTR Data Type Registry

EOSC IF EOSC Interoperability Framework

ER Entity-Relationship

FAIR Findable, Accessible, Interoperable, and Reusable

FDO FAIR Digital Object

FDOF FDO Framework

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

KIP Kernel Information Profile

KOS Knowledge Organization System

KR Knowledge Representation

LCIM Levels of Conceptual Interoperability Model

LD Linked Data

MFC Meta-FAIRification Characteristics

OOP Object-oriented programming

PID Persistent Identifier

PIT Persistent Identifier (PID) Information Type

PT Profile Typing

RDA Research Data Alliance

RDF Resource Description Framework

RT Record Typing

TBBR Thermal Bridges on Building Rooftops

TPM Typed PID Maker

UML Unified Modeling Language

XML Extensible Markup Language

Symbol / Form	Meaning	Notes
$S, A, \ldots$	sets	Capital letters for sets
$s, a, \ldots$	elements	Lowercase letters for elements of sets
$\lvert S \rvert$	cardinality	Number of elements of S
$A \subseteq S$, $A \subset S$	(proper) subset	Both variants appear
$P = \{S_1, \ldots, S_n\}$, $\mathcal{P}(S)$	set of sets / power set	Used when sets contain sets
A_I, A_i	indexed sets	Set A relates to set I or element $i \in I$
$S_{\text{Annotation}}$	annotated set	$S_{\text{Annotation}}$ that relates to set S
$\tilde{S}$	corresponding set	Corresponding set with same cardinality/index relation
$S_1 \times S_2 \times \ldots$	Cartesian product	Basis for relations
$R \subseteq (S_1 \times S_2 \times \ldots)$	n-ary relation	General relation notation
$\alpha \models \beta \Leftrightarrow (\alpha, \beta) \in R_{\models}$	named relation	Index names the specific relation
$f : A \rightharpoonup B$, $g : A \to B$	(partial) mapping function	Maps elements of A to B
$a \mapsto b$	element image	Image of a under the (implicit) map
$G = (V, E)$	directed graph	V nodes, $E \subseteq V \times V$ edges
$G = (V, E)$, $E \subseteq V^n$	n-ary graph	For higher-arity associations
$(x, y) \in E$ or $\{x, y\} \in E$	edge	Edge between vertices x and y
$\forall, \exists, \exists!$	quantifiers	For all, existential, exists exactly one
$\forall s \in S\ \exists A_s\ s.t.\ \ldots$	quantification	For all s that are element in S, it exists at least one set A_s such that ...
$\Rightarrow, \Leftrightarrow, \neg, \wedge, \vee$	logic connectives	It follows that, logical equivalence, negation, and, or
$x \coloneqq y$, $(x ::= y)$	new definition	x is defined as y, (in type syntax)
$x \mathrel{\hat{=}} y$	corresponds to symbol	x corresponds to y (no equality)
$\mathcal{O}(\cdot), \lesssim, \gtrsim$	big-O notation, order of	For asymptotic complexity and $\leq, \geq$
■(non-conventional) □	Definition & Proof	Indicates end, respectively
$\Gamma \vdash T$ $\quad$ $\Gamma \vdash M : T$ (generalized as $\Gamma \vdash \mathfrak{J}$)	typing judgment	Under well formed (w.f.) typing environment Γ, type T is w.f; and the w.f. term M has type T
$\Gamma \vdash T' <: T$	subtyping judgment	Under w.f. typing environment Γ, type T' is subtype of T
$\frac{\text{premises}}{\text{conclusion}}$	inference rule	Syntactic rules that transform premises (statements) into new, logically consistent conclusions.
(Rule Name)[Meta-conditions] $\frac{\Gamma_1 \vdash \mathfrak{J}_1 \ldots \Gamma_n \vdash \mathfrak{J}_n}{\Gamma \vdash \mathfrak{J}}$	general type rule form	Assert the validity of new typing judgments on the basis of valid judgments

Table 1.: General notation and mathematical symbols used throughout the dissertation.

1. Introduction

1.1. Motivation

The rapid growth of research data and the increasing importance of interdisciplinary collaboration challenge researchers, scholars, and other data users to manage, store, and share their data in ways that ultimately enable reuse [1]. The *FAIR principles* are internationally recognized guidelines for Findable, Accessible, Interoperable, and Reusable (FAIR) data that address these challenges. However, these guidelines are interpreted and implemented in various ways [2]. As a result, much of the research data is still difficult to find and access, poorly interoperable, and often reusable only with a limited scope or through time-consuming manual processing. Therefore, it is necessary to converge on the practices for handling data of different types, structures, domains, and storage locations [3]. The overall aim thereby is to optimize automated procedures to analyze and employ data in a machine-actionable way.

Promising solutions come from the concepts of data spaces [4] and FAIR Digital Objects (FDOs) [5]. The vision is to establish an ecosystem of machine-actionable FDOs across data spaces that enables use-case-specific application of the corresponding digital resources by different stakeholders. Developments towards this vision are currently fostered in international organizations such as the Research Data Alliance (RDA) [6], the FDO Forum [7], and the European Open Science Cloud (EOSC) [8]. However, until now, the precise definition of FDOs, their functionalities, and the underlying core mechanisms have only been partially addressed within limited and specific scopes. Whilst basic typing and trivial, standardized operations for Digital Objects already exist, the FDO context requires advancements to address the capabilities of machine-actionable data according to FAIR principles. A crucial aspect is how to model and implement FDO operations based on a formalized type system, which ensures the reliable and unambiguous applicability of these operations in a consistent and interoperable way. In addition, the corresponding base service infrastructure must support these type system-based mechanisms and requires advanced tooling to support agnosticism towards the underlying technologies for data and executable software.

This dissertation investigates the development of a solution for formalized modeling and applying operations for FDOs by means of typing. For this, principles from Object-oriented programming (OOP) and Knowledge Representation (KR) are combined with methodologies from (meta)data management and the Semantic Web. These modeling approaches are implemented and systematically investigated to show how FDOs can realize a domain-independent, machine-actionable ecosystem of digital resources.

1.2. Research Questions

The main focus of this dissertation is to explore how FDOs enable machine-actionable processing of research data by building on existing solutions and to determine if this approach can improve the (re)use of data independently of their domains. This main question can be further broken down into the following sub-questions:

1. How can FDOs *conceptually* align data spaces by FAIRification?
2. How can FDOs be consistently modeled using OOP and KR principles?
3. How can a type system for FDOs be formalized?
4. How can operations be associated with FDOs based on their types?
5. How can FDO operations be categorized and in which contexts are they applicable?
6. What technologies are suitable for the implementation and execution of FDO operations?
7. How do FDO operations enable machine-actionability?
8. Which layers of interoperability are improved by machine-actionable FDOs to ultimately facilitate data reuse?

1.3. Contributions and Outline

To answer these research questions, the dissertation is divided into three parts:

The core scientific contribution of the first part covered in Chapter 3 is a mathematically grounded foundation to **formalize an FDO Core Data Model** [9]. Lessons learned from earlier FAIRification and FAIR assessment works are thereby considered [10]. The structural and behavioral components of FDOs are defined using set theory, type theory, and predicate logic, enabling precise specification and validation of FDOs and their typed elements. The model systematically employs principles from OOP and KR, applying functionalities of objects used in computer programs for objects/entities used and connected in data spaces. On this basis, it is shown how FDO traits leverage domain-specific FAIR criteria at an abstract meta-level. The core data model is implemented as a proof of concept on example data from energy research [11] and digital humanities [12], demonstrating domain-agnostic applicability. Finally, the model is compared with other FDO developments, clarifying how it refines, generalizes, and extends existing approaches. This contribution covers research question 1, 2 and partly 3.

In the second part that is covered in Chapter 4, a **modeling approach for the association of operations with FDOs is developed based on an FDO type system** [13]. Adopting the mechanisms in type systems for objects in OOP, three essential typing mechanisms for FDOs are defined to realize the association of operations: Record Typing, Profile Typing, and Attribute Typing. This approach shifts the perspective of assigning an FDO's type

as a particular record property to a rule-based deduction. To evaluate FDO-operation associations that are based on these typing mechanisms, different quality indicators are considered. Indicators of simplicity, efficiency, and flexibility are defined and used for a quantitative analysis. For this, the typing mechanisms and involved FDO components are modeled as graphs (*association models*). Different measures are defined and used to evaluate the connectivity and time complexity of traversing each graph. In addition, purely qualitative aspects, such as the granularity of required client knowledge, and versatility, are defined and analyzed. On this basis, the typing mechanisms are compared and conclusions are drawn with respect to their interoperability, compatibility, and implementation. This contribution addresses the remainder of research question 3, and answers question 4.

The third part covered in Chapter 5 focuses on the **categorization of FDO operations, their implementation, and machine-actionable execution** to enhance data interoperability layers [14, 15, 16]. Three essential categories of operations are defined: Management-, Information Record-, and Data Resource Operations. The typing mechanisms are grounded in a class diagram model, and Attribute Typing is taken up as an implementation-ready approach by defining operations as operating FDOs, which can be mapped from and to class instances of the diagram. This approach results in a more holistic view of the FDO concept to abstract all types of digital resources, including executable software and interfaces. On this basis, a functional process model called FDO-Ops is developed that uses logical statements to express highly granular association conditions, enables polymorphism, and integrates execution protocols for purely object-level information processing. For evaluation, two complementary strands are pursued. First, it is examined how the advanced typing model defined by the class diagram can be implemented using a prototypical next-generation Data Type Registry. It is then demonstrated how this enables inheritance-aware operation discovery and consistency checks in future FDO infrastructures. Second, a service and software architecture is introduced on top of existing FDO base service components to implement FDO-Ops with the established Digital Object Interface Protocol (DOIP) as a testbed framework. Use cases are designed using the example datasets [11, 12] to assess the integration of FDO-Ops with state-of-the art tools, and the overall impact on enhancing different interoperability levels is evaluated. This contribution addresses the remaining research questions 5, 6, 7 and 8.

The dissertation is structured as follows:

Chapter 2 Elaborates on the background and theoretical foundations of objects and typing, the representation of knowledge, metadata management, and (FAIR) Digital Objects.

Chapter 3 presents the work on the development of a formalized FDO core data model, evaluation, proof of concept, and comparison with existing approaches.

Chapter 4 presents the work on the development of typing mechanisms in FDOs for associating operations and a comparative analysis of their quality indicators.

Chapter 5 presents the work on the implementation-related developments for machine-actionable FDOs and the evaluation of how they enhance interoperability layers.

Chapter 6 concludes the dissertation and provides an outlook for future work.

2. Background

This chapter introduces the background information from the relevant disciplines and topics in a bottom-up order: it starts with foundational notions of *object-oriented design and type systems* for operating on structured information, then addresses *knowledge-representation and modeling* mechanisms to express and link those structures, and finally connects both to *research data management* and in particular the field of *(FAIR) Digital Objects* on which the dissertation is based.

2.1. Object-Oriented Design and Type Systems

Objects in the context of information technology constitute a structuring mechanism to model real-world entities or things in a digital environment. By applying core principles of computer science as outlined below, objects can constitute complex systems that become networks of interacting components. These are the foundations for concepts and paradigms that realize information management and exchange through computer programs. [17]

Having its foundations in type theory, a type system is a logical system that classifies components of a program into types. In general, it has the purpose of preventing errors, defining constraints on permissible values and relationships, and providing guaranties about behavior. A type system defines a set of *type rules* based on *typing judgments*, which are formal assertions under a static and *well formed typing environment* expressing how the program components interact. *Well formedness* essentially means that an expression follows the syntactic rules of a language, so it is valid and interpretable within the system. *Type safety* is the property of a system that prevents unnoticed (untrapped) errors that cause arbitrary behavior. As a result, operations are not applied to values of incompatible types, so "type errors" cannot occur during execution or are reliably detected before the operation is performed. *Typechecking* is the process of verifying and enforcing the corresponding rules, e.g., to ensure that method calls, data access, and object interactions are consistent with class and interface definitions. When passing the typechecker, a program, or in general terms a system, is called *well-typed*. *Static typing* performs typechecking before runtime, usually at compile time, whilst *dynamic typing* does it at runtime. [18, 19]

Mostly known in the context of programming languages, types are also used in data modeling and databases, and for KR systems within ontologies. In the context of OOP, by which many developments in this dissertation are inspired, type systems define the mechanisms for classifying values, structuring object hierarchies, and enabling interactions

between objects. They serve as contracts that govern object behavior, enforce abstraction, and enable safe reuse through consistently applied subtyping and polymorphism. [20]

2.1.1. Abstraction and Encapsulation

Abstraction is a fundamental principle in computer science and is one of the pillars of OOP. It separates the behavior of a system from its implementation to hide irrelevant details and highlight essential properties, thereby generalizing the system's functionality [21]. Information hiding behind module interfaces refers to a core modular design principle for building systems that are easier to maintain and evolve [22]. A mathematical foundation for abstraction in software development is provided by [23] through abstract data types, which are defined by their behavior (operations and properties) rather than by their representation. The work of [24, 25] describes abstraction in relation to types and polymorphism, showing how different levels of type abstraction allow flexible yet safe program design.

Alongside abstraction, encapsulation is another pillar of OOP. Encapsulation bundles data with its operations into a single unit to restrict direct access to its internal state through well-defined interfaces [26]. Closely tied to abstraction and rooted in information hiding, it enables developers to reliably reason about components in terms of their specified behavior rather than their implementation [27]. Type systems reinforce encapsulation by statically restricting which operations are permitted on a given data structure [20]. In OOP, classes are used to package abstraction (what an object as an instance of a class does), with encapsulation (what its internal state representation is) [28, 29]. In terms of type theory, a class defines a type, and all its instances are within the type's value range [24].

2.1.2. Subtyping, Polymorphism and Inheritance

The foundational taxonomy of subtypes and polymorphism is provided in [24]. Central to the flexibility in OOP, *subtype polymorphism* is the use of an object of a subclass wherever its superclass is expected. This enables the use of objects of different types through a shared interface or base types. According to the Liskov Substitution Principle [30, 31], subtyping must preserve behavioral correctness such that objects of a subtype must be usable like objects of a supertype without altering essential program characteristics.

Inheritance is a mechanism for defining new classes based on existing ones, thereby reusing attributes and behavior to define more specific extensions [32]. Subtyping works analogously to, and is often realized through inheritance, but inheritance only complies with the Liskov Substitution Principle if a subclass preserves the behavior of its superclass [33].

2.1.3. Forms of Type Systems

To determine *type compatibility* (also called type equivalence), there are two approaches: either the type and subtypes are explicitly declared through relationships between named

types (e.g., through inheritance), or the type compatibility and subtypes are determined by the type structure considering properties and supported operations [24, 20]. Based on these approaches, three main type systems exist in OOP that associate the type of an object with specific system components:

- **Nominal typing**: the type of an object is determined by its class name and inherited superclasses [20].
- **Structural typing**: the type of an object is determined based on its properties and methods, typically at compile time [20].
- **Duck typing**: the type of an object is determined based on its properties and methods, typically at runtime, and is therefore a more dynamic version of structural typing [34].

2.2. Knowledge Representation

The term Knowledge Representation (KR) stems from an area of Artificial Intelligence (AI) that is concerned with the encoding of knowledge about the world into structured formal representations that a computer can use to reason, draw inferences and act upon [35]. In this section, we focus on KR in the context of domain information modeling adopting a structured approach to identify entities, their relationships, and the vocabularies used to describe them.

2.2.1. Entity-Relationship Model

Originally introduced by [36], the Entity-Relationship (ER) model is used to represent data as entities and their relationships. A major aspect is the provision of independence from domain-specific data described through set theory and relation theory. It is originally derived from three models (the network model, the relational model, and the entity set model) and is considered a generalization and extension of them. It is mainly used in database design, but has also been adopted in other branches of computer science, such as software engineering [37]. As pointed out in [38], ER modeling complements object-oriented (OO) models because it provides a disciplined conceptual foundation for modeling entities, relationships, and identification that OO designs can build upon (often by adding behaviors/methods).

The original ER model provides a logical four-level view of data [36]. In the context of this dissertation, we make use of **Level 1: Information Concerning Entities and Relationships**. For convenience, we report the original formal definition given by [36], as we later refer to it.

An entity is defined as a "thing" that can be distinguished from others. A relationship is defined as an association among entities. To test whether an entity e belongs to the set of entities E_i, there exists a predicate associated with each entity set E_i. Entity sets may not

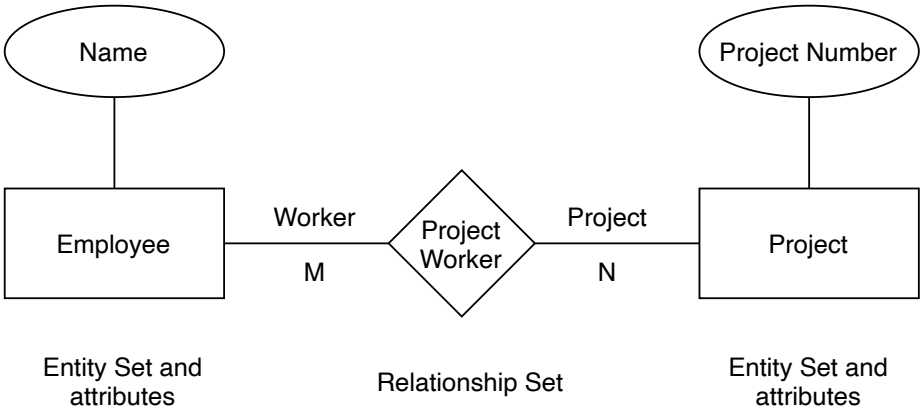


Figure 2.1.: A minimal ER-diagram. Adopted from [36].

be mutually disjoint. Let R_i be a relationship set that is a mathematical relation among n entities, each taken from a set:

$$\{[e_1, e_2, ..., e_n] \mid e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n\}, \tag{2.1}$$

where each tuple of entities $[e_1, e_2, ..., e_n]$ is a relationship. Entity sets E_i in Equation (2.1) may not be mutually disjoint. Entities have roles r_i in the definition of a relationship which are stated as: $r_1/e_1, r_2/e_2, ..., r_n/e_n$ that determine their function in the relationship.

A set of attribute-value pairs expresses information about an entity or a relationship. Values are classified into different value sets V_i. A predicate is associated with each V_i to test whether a value belongs to it. An attribute is formally defined as a function f that maps from an entity set E_i or a relationship set R_i to a value set V_i or a cartesian product of value sets $V_{i_1} \times V_{i_2} \times ... \times V_{i_n}$:

$$f : E_i \textbf{ or } R_i \rightarrow V_i \textbf{ or } V_{i_1} \times V_{i_2} \times ... \times V_{i_n} \tag{2.2}$$

A way of representing the information structure is the ER diagram, for which an example is depicted in Figure 2.1. In this original notation, the entity sets are represented by rectangular boxes, the relationship sets are represented by diamond-shaped boxes, and the attributes are represented by ovals. The roles of the entities in the relationship and the cardinality are provided as line annotations.

2.2.2. Knowledge Organization Systems

Knowledge Organization Systems (KOS) organize and represent information by modeling the semantic structure of a domain and its vocabulary. This comprises multiple types of schemes for organizing information, which can be taxonomically grouped with respect to their structure and function [39], summarized as follows:

- **Term Lists**: are simple vocabularies that harmonize terminology without complex structure.

- **Metadata-like Models**: structure terminology around entities or attributes.
- **Classification and Categorization**: organize items into hierarchically structured and controlled subject groupings.
- **Relationship Models**: are knowledge organization structures, such as thesauri, semantic networks, and ontologies that represent concepts as terms and explicitly define the relationships between them, from simple equivalences and hierarchies to richer, rule-based connections.

2.2.3. Semantic Web

The Semantic Web is an extension of the existing World Wide Web to make machine-interpretable information available for easier exchange and navigation [40]. A central aspect of this vision is that computers can access structured data based on the Resource Description Framework (RDF), and rule sets that enable automated reasoning over (linked) data [41].

2.2.3.1. Resource Description Framework

RDF is a framework for modeling the representation of information and data exchange on the web [42]. The basis of an RDF data model is an abstract syntax with a set of triples consisting of a subject, a predicate, and an object, represented as a node-edge-node link in an RDF graph. A minimal example of such a graph is illustrated in Figure 2.2.

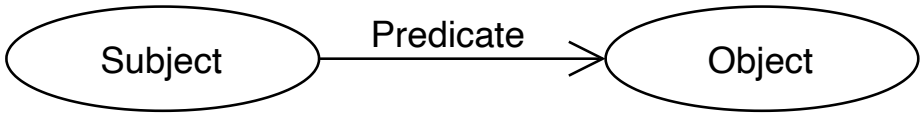


Figure 2.2.: A minimal conceptual RDF graph of two nodes connected by an edge, forming a triple. Adopted from [42].

The term *resource* in RDF is used as a synonym of the term *entity*, which may be anything from a physical thing to an abstract concept. Resources are denoted by Internationalized Resource Identifiers (IRIs) and literals of the graph nodes, which have data types that define the range of possible values. [43]

OWL extends RDF with a description-logic–based ontology language that adds rich class and property constructs (e.g., class expressions, cardinalities, equivalence/disjointness, property characteristics) and formal semantics for automated reasoning [44]. ShEx [45] and SHACL [46] are shape languages to validate RDF graphs against defined constraints to ensure data integrity and provide clear validation feedback to users.

The SPARQL Protocol and RDF Query Language (SPARQL) is the standard query language for RDF data to retrieve, manipulate, and construct data stored in RDF. SPARQL queries

typically match graph patterns against RDF triples and can return results in a serialized format [47].

2.2.3.2. Linked Data

Linked Data (LD), and specifically Linked Open Data (LOD) when using open licenses, aim to interlink data resources on the web by (re)using URIs that are resolvable via the Hypertext Transfer Protocol (HTTP). It consists of 4 principles, originally defined by [48]:

- "Use URIs as names for things".
- "Use HTTP URIs so that people can look up those names".
- "When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)".
- "Include links to other URIs, so that they can discover more things".

With LD, both humans and machines can access data across different servers and have a technical approach to interpret the provided semantics [49]. Given its relationship to RDF, LD models structure data as a graph that spans the web, thus working towards the Semantic Web vision [50].

2.3. Research (Meta)data Management

This section introduces the relevant concepts of metadata- and data management. The FAIR Guiding Principles have become central guidelines in this field, and the increasingly prominent concept of data spaces can provide the technical and governance framework for their implementation. However, the dimension of interoperability is particularly complex as described in Section 2.3.3.

2.3.1. Metadata

Metadata management is the key to maintaining an efficient and thriving digital ecosystem where data can be considered as digital resources that fuel procedures in the information age. Metadata can be defined as "the information we create, store, and share to describe things and that allows us to interact with these things to obtain the knowledge we need" [51]. There are a variety of different types of metadata, with slightly different categorizations between communities, but an often cited overarching classification that provides a good orientation is summarized in Table 2.1 [51].

Metadata are therefore almost everywhere included where data are produced and stored, be it as direct entries in repositories and relational databases records, or as individual file

Table 2.1.: General overview of metadata types according to [51].

Type of Metadata	Purpose
Descriptive metadata	Finding or understanding a resource
Administrative metadata	
- Technical metadata	- Decoding and rendering files
- Preservation metadata	- Long-term management of files
- Rights metadata	- Intellectual property rights attached to content
Structural metadata	Relating parts of resources to one another

formats such as Extensible Markup Language (XML) or JavaScript Object Notation (JSON), serialized with RDF in the context of LD [52]. In that sense, metadata are also digital resources that can be used in different ways depending on the context of the application case.

2.3.2. FAIR principles

The FAIR Guiding Principles for scientific data management and stewardship have been introduced with a particular focus on making digital resources findable, accessible, interoperable, and reusable in a machine-actionable way [1]. In that sense, these principles are quality traits with synergistic effects and the overall aim to re-use data in downstream applications. Each principle has a set of criteria that specify their characteristics of which some are very concrete (e.g., F1: "(meta)data are assigned a globally unique and persistent identifier"), whereas others leave a lot of space for interpretation (e.g., F2: "data are described with rich metadata"). These criteria have been extended and newly interpreted over time in the context of specific types of digital resources and applications, such as research software [53], computational workflows [54], AI [55], and semantic interoperability [56]. The original FAIR criteria, those formulated in FAIR for Research Software (FAIR4RS) [53, 57], and those for the updated FAIR principles (FAIR2.0) [56], are provided in Appendix Tables A.1 to A.3, respectively.

2.3.2.1. Basic Concepts and Technologies

The most relevant and basic concepts and technologies that are used for realizing FAIR principles are:

- **Data Storage Systems:** repositories are central, permanent storage systems where digital resources are managed and accessed [58]. They can be special-purpose, often institutional, as well as open global-scoped and general-purpose repositories [59]. In data management ecosystems, institutional data repositories may be complemented by separate analytical storage or processing architectures such as data lakes, data

warehouses, or a mixture called data lakehouses [60], especially in big data contexts [61]. A data lake is a scalable store that keeps heterogeneous (usually raw) data in original formats and applies some structure at read time (schema-on-read) [62], while a data warehouse integrates, cleans and models data up front (schema-on-write) [63].

- **Persistent Identifier (PID) Systems:** provide permanent stable references (the PIDs) to digital objects or metadata about physical objects (the digital resources). Typically, PIDs, consisting of an authority-specific prefix and a suffix string pattern, are globally unique and aim to provide long-term referencing to their associated resources, independently of changes in their metadata or location. Among the most prominent ones are Handles [64], specifically Digital Object Identifiers (DOIs) [65], and Archival Resource Keys (ARKs) [66].
- **Knowledge Organization Systems:** already described in Section 2.2.2, they comprise multiple concepts to structure, organize, and relate semantics, and are thereby a way to leverage metadata [67].
- **Semantic Web and Linked Data:** already described in Section 2.2.3, they anticipate key ideas of the FAIR principles by promoting machine-readable resources through RDF and controlled vocabularies [68]. Not inherently covered are persistence and fully machine-actionable metadata for reuse [69, 70, 71, 72].
- **Collections:** they are described as location-independent, criterion-defined views on distributed resources [73]. Collection membership is set logically by explicit selection rules and operationally by what the collection's discovery tools can retrieve directly. Conceptually, they fit the definition of Research Objects (ROs) [74] as semantically enriched aggregations of resources that embody a scientific purpose or advance a specific research objective.

2.3.2.2. Machine Abilities

FAIR data emphasize machine-actionability, building on machine-readability and interpretability. A possible definition of these terms is provided by [75], which is adopted and extended in this dissertation:

- **Machine-readability:** the ability of machines to parse and process the syntax of a digital resource. Human intervention is still required to interpret the context.
- **Machine-interpretability:** the ability of machines to interpret the technical, syntactic, and semantic details of a digital resource for content-based validation and reasoning. Human intervention is still needed to act according to the potential conclusions drawn from this interpretation.
- **Machine-actionability:** the ability of machines to operate on a digital resource based on the previous interpretation of its technical, syntactic, and semantic details, using a system with structures and rules against which can be programmed. Human

intervention is not required at the stage of full machine-actionability, but may be necessary at intermediate stages.

2.3.3. Interoperability

There are different definitions and categorizations of interoperability in the literature, sometimes redundant, depending on the domain and subject of interest. The recent work of [76] conducts a tertiary study based on which a classification of types, models, and interoperability frameworks is provided. According to this study, the European Interoperability Framework (EIF) [77] is currently one of the most complete and well-known frameworks. It was also the inspiration for the EOSC Interoperability Framework (EOSC IF) [78] which defines four layers of interoperability in the context of the FAIR principles, namely technical, syntactic/semantic, organizational and legal interoperability.

A work in line with the EIF and the EOSC IF is provided in [79] where three tightly coupled facets are defined, called the interoperability trilogy for Systems of Systems. Each facet is based on different types of interoperability (data, service, and system interoperability) addressing essential components in (meta)data management.

These types of interoperability connect to the Levels of Conceptual Interoperability Model (LCIM) (originally in [80], later extended in [81, 82]), which is also used in the context of the Semantic Web [83]. The LCIM follows a bottom-up approach (L0 to L6) to capture conceptual interoperability between systems. Some of these levels align directly with those of the EIF and EOSC IF, i.e., technical, syntactic, and semantic interoperability.

The LCIM effectively provides descriptions of each level and the requirements to achieve them. This enables interoperability assessment by expertise and judgment using a qualitative measurement mechanism [84].

As also outlined in [76], the interoperability types of technical, syntactic, and semantic interoperability occur the most in different studies that interpret them in the context of different domains and use cases.

The interoperability layers of the EIF and EOSC IF directly aligned with the LCIM levels, up to level 3, can be defined as follows:

- **Level 0 - No Interoperability**: systems are separate. They cannot exchange data in a meaningful way.
- **Level 1 - Technical Interoperability**: systems can connect and pass bits (via networks and protocols) using agreed standards, interfaces, and security mechanisms, but the content is opaque.
- **Level 2 - Syntactic Interoperability**: systems use the same data format/structure, and messages can be parsed.
- **Level 3 - Semantic Interoperability**: systems share the meaning of the data (common vocabulary or model), and parsed data are understood in the same way.

The remaining layers not directly aligned with the LCIM, as well as the interoperability trilogy facets, can be defined as follows:

- **Organizational Interoperability**: processes, roles, and responsibilities across organizations are aligned such that a service actually works end to end.
- **Legal Interoperability**: rules and laws allow data and services to be shared across borders and administrations.
- **Data Interoperability**: the ability to exchange and use heterogeneous (meta)data through standardization and portability, accessibility and openness, as well as privacy and security.
- **Service Interoperability**: the ability to discover and use the services of others through automation and machine-readability, requiring standardized service interfaces.
- **System Interoperability**: the infrastructural features that enable collaborative use of complementary services and technologies across otherwise independent systems.

2.3.4. Data Spaces

The term *data space* is not well-defined and constitutes an umbrella term for multiple related concepts, summarized in [85]. In its first definition introduced by [86], a data space is an abstraction for data management procedures on distributed and decentralized digital sources regardless of their format, location, model, or structure. Another important aspect is the sovereignty of data owners, security, and governance, especially in an industrial context, as proclaimed by the International Data Space Association (IDSA)[1] [87]. From a practical point of view, data spaces provide a framework (technical, organizational, legal, and governance) that supports trusted data sharing and value creation within data ecosystems across (big) data value chains [85, 88]. In a research context, these value chains are used to enable collaboration among diverse, interconnected participants that benefit from each other. Since data spaces provide infrastructure for the integration of heterogeneous data from multiple sources and enable data sharing between organizations, they would be strengthened by FAIR compliance [89]. The process of transforming uncurated data ecosystems into FAIR-compatible data spaces is achieved by *FAIRification* [90, 91].

[1] https://internationaldataspaces.org/

2.4. Digital Objects

In their original work [92] and a later revision [93], the authors propose a uniform framework to identify, access and manage digital information, independent of its location or file format. The framework responds to a key limitation of the World Wide Web as an information infrastructure: resources are distributed across many sites, represented in heterogeneous formats, and referenced by identifiers that are often unstable. To address this issue, the authors separate the informational content, i.e., the Digital Object (DO), from the mechanisms used to store and access it (repositories and access protocols).

In this framework, a DO is an instantiated abstract data type composed of typed data and key metadata. Typed data refers to the object's data representation, and DO subtypes are introduced by hierarchically adding metadata fields. The key metadata ensures uniqueness of the object and contains at least a PID. Additional requirements may vary.

Beyond the object model itself, the authors ([92, 93]) specify the foundational concepts of the system, including mutability properties, a Repository Access Protocol for interacting with DOs and infrastructure components such as handle servers. Together, these elements form the basis for a distributed digital information service in which DOs can be stored, accessed, disseminated, and managed in heterogeneous repositories.

Compared to other works on distributed objects like CORBA or RMI-IIOP [94], which assume controlled, homogeneous middleware-managed environments, DOs are designed for open, heterogeneous cross-domain settings. They rely on globally resolvable identifiers and metadata descriptions instead of middleware-specific bindings.

2.4.1. Digital Object Architecture

The Digital Object Architecture (DOA) is a technical expression of the conceptual DO framework [93]. It is an extension of the Internet architecture that supports information management beyond simply transmitting digital information from one point to another [95]. Part of the DOA is the Digital Object Interface Protocol (DOIP) that defines how clients interact with DOs and their services [96]. In DOIP, every DO must have a defined serialization format, typically JSON, to ensure interoperability between systems. In the native DOA, each DO has a type that describes the semantics of its content.

DOIP defines a set of basic and extended operations. Basic operations must be implemented in all DOIP services, i.e., `hello` (to retrieve service information), `create` (to register and deposit a new DO), `retrieve/access` (to obtain information or elements of a DO), `update` (to update information or elements of a DO), `delete` (to permanently remove a DO or point to a tombstone), `search` (to discover DOs based on metadata), and `listOperations` (to see which operations are supported for a given object). Extended operations are meant to provide additional manipulations for the metadata of a DO or its bit sequence. In this paradigm, an extended operation is itself a DO which trivially has an identifier. These operations are either executable as part of the service implementation or within a runtime environment

provided by the service. Interactions between clients and DOIP services must follow a secure request/response model. In the DOA, everything (such as services, targeted DOs, or operations) has an identifier and is addressed by it [96].

A prime example of the application of the wider DOA is the Handle System [64] that serves as the resolution component of the DOI System [97] which is used, for example, for scholarly publications and related outputs.

2.4.2. Digital Objects Core Data Model

Building on the original definition of [93], the RDA Data Foundation & Terminology (DFT) Working Group introduced a core data model for DOs [98]. Here, a DO is defined as the central organization unit of information, as depicted in Figure 2.3, whose content is represented by a bit sequence stored in a repository. In contrast to the original description, the PID, as a generalization of a Handle, is separated from the key metadata, and the metadata itself is treated as a DO. The model further introduces collections as explicit aggregation constructs, i.e. a collection that groups DOs and is itself represented as one. In the FAIR lifecycle architectures proposed by [99], DOs can serve as the packaging and access unit that binds data and all related artifacts (metadata, methods, workflows, integration results) together by their PIDs to be found, accessed, and reused consistently. This "core data model" is best understood as a metamodel [100] for the conceptual and technology-agnostic representation of the entities and their relationships. However, the model exhibits cyclic dependencies: DOs are described by metadata objects and aggregated by collections, both of which are also modeled as DOs.

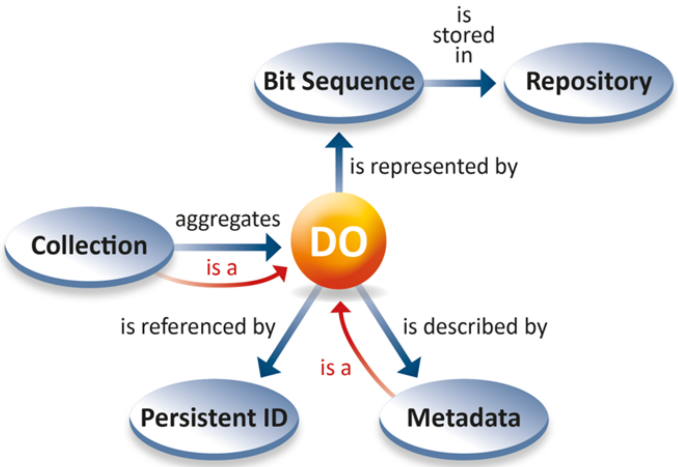


Figure 2.3.: The original Core Data Model of Digital Objects of the RDA DFT group, taken from [3].

A further conceptual refinement concerns the functional role of the PID, as illustrated in Figure 2.4. In earlier descriptions, the PID primarily serves as a reference to the typed data that the DO is composed of. In the DFT model, the PID resolves to a PID record, a dedicated metadata record that provides the information required to locate and access

the associated bit sequence (or an equivalent copy) and to reference additional metadata. Within this framework, also individual metadata attributes of the PID record must be explicitly typed so that they can be interpreted and processed in a machine-actionable manner. [3]

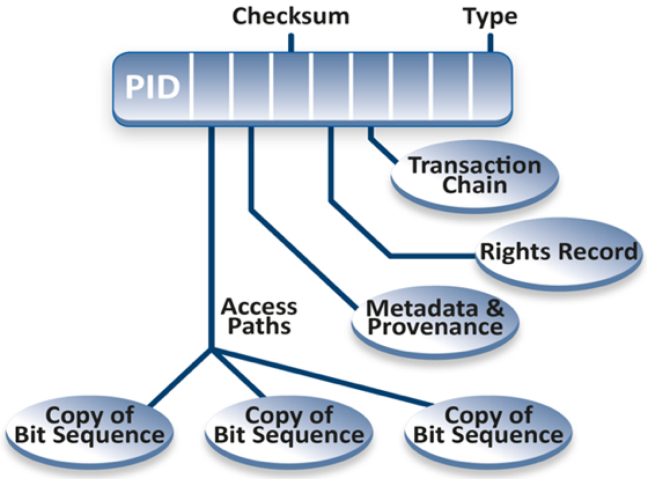


Figure 2.4.: The PID record concept for Digital Objects of the RDA DFT group, taken from [3].

This perspective on DOs emphasizes encapsulation, abstraction, and persistent identification of elements, and is naturally related to established concepts such as object orientation, abstract data types, and object stores [3]. The intended paradigm is that users do not directly engage with a DO's internal bit sequence, but interact through well-defined operations and metadata. DOs thus provide a consistent high-level view of digital entities that is independent of their internal representation. PIDs of these DOs can be interlinked, forming an identifier graph that preserves essential context and provenance pathways across evolving infrastructures, even if individual resources later move or vanish [101]. By pairing PIDs with abstract data types (e.g., lists, sets, maps), *actionable PID collections* of DOs have standard operations (e.g., add, remove, lookup) through a universal interface layer, making them interoperable across PID systems [102]. This requires infrastructures that support not only resolving identifiers but also precise PID typing and DO relationships. Closely related, in [103, 104] the authors introduce the notion of a PID graph, which amplifies the benefits of PIDs by connecting them via their metadata.

The view on DOs as proposed in the RDA's DFT core model connects to an earlier adoption of the DO concept within digital libraries: in [105], Digital Object Prototypes (DOPs) are introduced as explicit object-oriented type definitions that specify an object's parts and behaviors, with runtime checks for conformance and behavior binding [105]. A later implementation highlights encapsulation, metadata constraints and mappings, and repository automation such as file conversions [106]. Overall, they argue that making types explicit and machine-enforceable via DOPs turns heterogeneous digital library content into uniformly manipulable objects, improving reuse, configurability, and library service development.

2.4.3. Digital Object Typing and Kernel Information

This section provides an overview of the key concepts and technologies used to define and type DO content.

2.4.3.1. Data Type Registries

Data Type Registrys (DTRs) support the standardization of type definitions through community-driven registries rather than the slower international consensus process. They store machine-readable and PID-referenced type definitions, which can be validated. The RDA DTR Working Group [107] addressed the challenge of enabling automated data processing and cross-domain interoperability by defining how these data types can be registered, validated, standardized and reused. The conclusion is that no single global registry could serve all domains. Instead, a *federated* registry system is needed, allowing communities to define and manage their own types while maintaining interoperability through PIDs, shared Application Programming Interfaces (APIs), and governance principles. The use cases for DTRs identified by [107] include enabling automated validation of data, supporting dataset discovery by types, linking types to services for type-specific operations, and clarifying semantics to improve reuse and reproducibility. Their data model proposes that each type has an identifier, description, provenance, semantics, standards references, and relationships to other types.

Overall, DTRs complement PIDs by declaratively associating typed elements in DOs with valid operations, emphasizing encapsulation. This makes the infrastructure flexible and extensible, allowing new operations to be added without altering the internal organization of the DO. The envisioned result is a virtualization layer that allows users to interact with logical representations of DOs rather than physical storage.

2.4.3.2. PID-Information Types

PID-Information Types (PITs), conceived by the RDA PIT working group [108], model types according to the requirements of DTRs. They provide a structured way to associate machine-readable typed metadata directly with PIDs, enabling automated processing and supporting discovery, access, and verification of integrity and authenticity. PITs are modeled through *properties* that define elements of atomic metadata, *types* that aggregate them into coherent groups, and *profiles* that aggregate types for domain-specific needs, as shown in Figure 2.5.

To embed PITs within the broader PID infrastructure in conjunction with DTR and PID services, they also propose a PIT API service, abstracting PID operations for clients and enabling further abstraction via higher-level API adapters, as depicted in Figure 2.6.

To ensure that every registered PIT not only has a definition, but also a machine-interpretable schema to validate instances, the work in [109] introduces a method to automatically derive JSON Schemas from hierarchical PIT descriptions. Such hierarchical descriptions

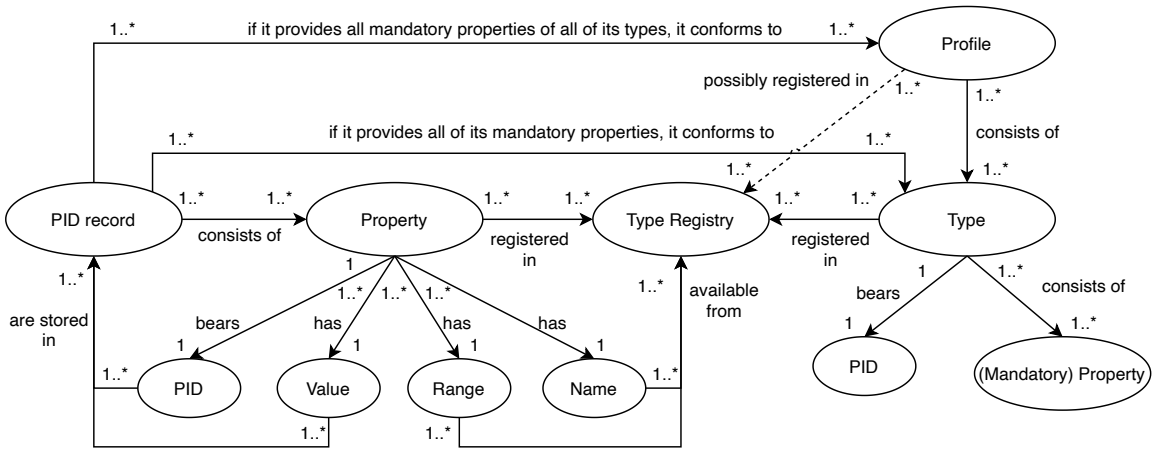


Figure 2.5.: The conceptual property-type-profile model of PITs, derived from the description in [108] with addition of partial formalization using cardinality.

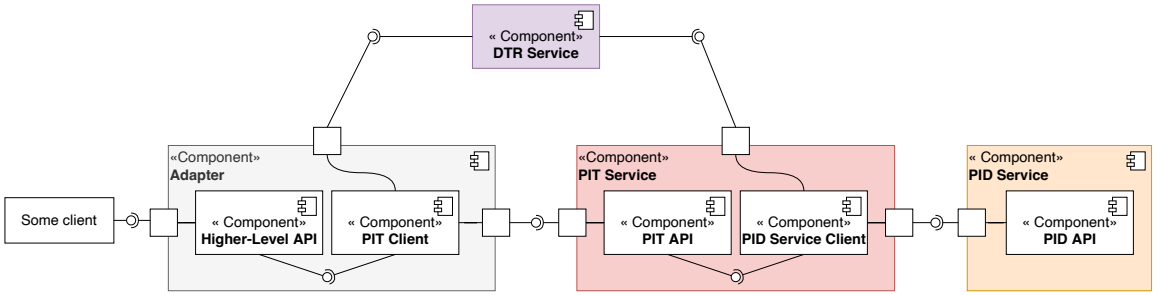


Figure 2.6.: Envisioned high-level architecture for PITs following long-term adoption: PID infrastructure providers would provide PIT API services alongside their existing PID services. Adopted from [108] and adapted to Unified Modeling Language (UML) component diagram.

treat a PIT as a composition of simpler building blocks: at the lowest level are *Basic PITs*, which rely only on primitive JSON datatypes (string, integer, boolean, …) with possible restrictions (regex, ranges, enums, …). *Composite PITs* (originally just referred to as PITs) are hierarchical structured aggregates defined by combining these Basic PITs or other Composite PITs. Each subtype is explicitly registered with its own PID, so the hierarchical structure of a given PIT can be recursively navigated and the corresponding pieces of the JSON Schema can be generated, resulting in a complete JSON Schema that enforces both syntactic validity and structural rules of the type. This method, outlined in Algorithm 1, is used in existing DTR systems, such as the ePIC DTRs [110], to validate PITs in DO PID records.

Algorithm 1 Schema Extraction Algorithm Pseudocode based on the description in [109].

Input: Main PID
Output: JSON Schema

```
1  Retrieve and validate PIT description of the Main PID from the PIT registry
2  Initialize properties section from the main PIT description
3  Initialize empty list definitions // is expanded during schema construction and
     contains definitions of main type, all subtypes and subtypes of subtypes
4  Call BackTracking(Main PID)
5  foreach encountered PIT during backtracking do
6      Create a definitions list entry for the PIT according to the type definition keys with
         the PID as reference value
7      Identify subtype PIDs in the PIT description
8      if subtype PIDs exist then
9          foreach subtype PID do
10             Call BackTracking(subtype PID)
11     else
12         Mark PIT as basic // no recursion
13 Build JSON definitions section for types from definitions list
14 Set JSON properties section to the structure of the (main) PIT from step 1
15 return complete JSON Schema
```

2.4.3.3. PID Kernel Information

PITs address the technical requirements for typing PID record information, whilst the specification of the record content is provided in the RDA recommendations on PID Kernel Information [111]. The authors introduce a minimum, carefully selected set of generic metadata attributes to embed in a PID record, enabling fast processing of the essential information about the corresponding bit sequence even without accessing it. A Kernel Information Profile (KIP) is a registered schema for interpreting PID records, defining allowed attributes, and their representation. Therefore, a PID record is an instantiation of a KIP, analogous to an object being an instance of a class in OOP. A set of guidelines is provided in [111] to define a KIP:

- PID records are machine-actionable, non-authoritative metadata, with kernel information stored directly in the resolver.
- They are owner/delegate controlled and should be updated rarely.
- The kernel and profile attributes are atomic key–value pairs that describe only the identified object, independent of other objects or attributes.

The authors in [111] also define the draft KIP recommended by the RDA for a generic description of digital resources, referred to as RDA KIP.

2.4.4. From Digital Objects to FAIR Digital Objects

DOs provide the foundation for a structured approach to managing digital entities through PIDs and metadata to support automated processing, manage access, and reuse data in a reliable way. Some of these aspects align with the FAIR principles: DOs implement the Findability (F) and Accessibility (A) dimensions almost directly. Their model defines the binding mechanisms between identifiers, metadata, and bit sequences, ensuring stable references and persistence across repositories. Certified repositories, PID infrastructures, and metadata persistence mechanisms make DOs naturally FAIR with respect to stable identification and access [112]. In the dimensions of Interoperability (I) and Reusability (R), DOs can provide facilitation rather than complete solutions. Through their type-based binding concept, DOs offer stable and potentially machine-actionable links that could enable interoperability across repositories regardless of underlying storage or modeling. However, they lack the rich, qualified semantic descriptions, detailed provenance, licenses, and reuse conditions explicitly required by the FAIR Principles [113].

The further extension and alignment of DOs with the FAIR principles gave rise to the concept of FAIR Digital Objects (FDOs) [5], combining the technical framework provided by DOs with the semantic and community-driven requirements contributed by FAIR [114]. The concretization and refinement of this concept is the subject of ongoing research within the FDO community.

3. Formalizing a FAIR Digital Object Core Data Model

This chapter builds on the work published in the proceedings of the IEEE 2024 International Conference on Big Data (BigData) [9], and is related to the works published in the Scientific Data Journal [11], in the proceedings of the 19^{th} International Workshop on Ontology Matching [12], and in the proceedings of the 3^{rd} Workshop on Metadata and Research (objects) Management for Linked Open Science (DaMaLOS) [10]. The main outcome is a formalized core data model that defines FAIR Digital Objects (FDOs) and illustrates how they align data spaces by FAIRification using principles from OOP, KR, and typing (answering to sub-questions 1,2 and 3 in Section 1.2).

3.1. Introduction

The FAIR principles [1] provide guidelines for proper data management and stewardship, and have been implemented in many ways for FAIRification of data [115], depending on the requirements and practices within different communities [2, 116]. On the other hand, compliance with the FAIR principles, so called *FAIRness*, is difficult to measure [117, 118] and is usually assessed in community-specific contexts [119, 120, 121], posing challenges for a direct comparison [122]. Partial and inconsistent FAIRification across and within data spaces results in reduced research efficiency [113]. This is particularly impactful in data-intensive fields that rely on good (meta)data management practices, for example Artificial Intelligence (AI) [123, 124], and is even amplified in the context of big data, especially with respect to interoperability [125].

This issue can be addressed by an abstract, domain independent, and machine-actionable representation of the content of each data space without altering their FAIR-compliant configurations and ensuring full control over their digital resources. FDOs, which enable machine-actionability at the level of the individual digital resource, offer a promising concept for the realization of a global integrated data space [126]. The precursors of this vision are, for example, the Digital Object Abstraction Layer (DOAL) for reusable digital library services [127], or the architectural organization and abstraction of physical implementations by FDOs within data lakes [128]. However, the FDO concept has always been kept as a high-level specification. The lack of a rigorous formalism resulted in divergent interpretations and individual domain-specific solutions that are not directly compatible with each other, missing a holistic approach.

This chapter provides a formalized FDO core data model for abstract meta-level representations of digital resources in data spaces and is structured as follows:

- Definition of the problem in current FAIRification practices in Section 3.3.1.
- Development of the envisioned solution of a meta-level aligned with the FDO concept in Section 3.3.2.
- Developing a formal definition of FDOs that satisfies the requirements of the proposed meta-level in Section 3.3.3.
- Evaluation and implementation of the model as a proof of concept in Sections 3.4.1 and 3.4.2.
- Comparison of the model with other FDO models in Section 3.4.3
- Overall discussion in Section 3.5.

3.2. Related Work

The FDO Forum [7] drives the development of FDOs nowadays and provided a technical definition [129, 130] and a list of **conceptual specifications** [131]. However, key aspects remain open to interpretation, including the precise definition of an FDO, the implementation of typing, the essential technologies involved, and the position of FDOs within the data ecosystem in relation to concepts such as the Semantic Web.

Within the RDA, the development of a conceptual core data model of DOs from the Data Foundation and Terminology (DFT) working group [98], together with the recommendations on PID Information Types (PITs) [108], on PID Kernel Information [111], on Data Type Registries (DTRs) [107], and on Research Data Collections [132], culminated in the introduction of **Handle-based FDOs**. Early adoption attempts are attributed to the Helmholtz Metadata Collaboration (HMC) [133, 134]. Legacy DOs based on these developments were created in the Persistent Identification of Instruments (PIDINST) [135, 136] and in the Digital Research Infrastructure for the Arts and Humanities (DARIAH) [136]. The biodiversity community introduced similar FDO implementations using the Distributed System of Scientific Collections (DiSSCo) research infrastructure [137, 138, 139]. However, their FDO profiles tend to deviate from the RDA recommendations [140].

Other interpretations of FDOs aim to integrate them as part of the **Semantic Web**, and mainly apply the FDO concept using technologies that are focused on Linked Data (LD) [141]. In this context, Research Object Crates (RO-Crates) have been proposed as an FDO implementation strategy [142, 143, 144, 145], also in conjunction with FAIR Signposting called Webby FDOs [146, 147, 148]. An other relevant application to the Semantic Web is the FDO Framework (FDOF) Ontology [149], an OWL implementation of the FDOF conceptual model, which has been used to model FDOs as Nanopublications [150].

Overall, different implementation approaches do not follow a coherent and formal data model, have conceptual gaps, lack clarity, or are not fully aligned in their scopes, as also pointed out in [151].

3.3. Methodology

3.3.1. Alignment of Data Spaces through FAIRification

We describe and formalize the problem and the proposed solution for the alignment of data spaces (s. Section 3.1) by modeling them using set theory. Unlike FAIR assessment tools, this method does not assess FAIRness within a data space, but rather measures- and enables visualization of data space alignment by common FAIRification characteristics.

Definition 1 (FAIRified data space alignment) Let D be the set of digital resources, C be the set of FAIRification characteristics, and $\mathcal{R}_{\models} \subseteq D \times C$ a satisfaction relation with $d \models c \iff (d, c) \in \mathcal{R}_{\models}$, where $d \models c$ means that the resource d satisfies the characteristic c. Let $\mathcal{P}(C)$ be the power set of C and $\mathcal{P}(\mathcal{R}_{\models})$ be the power set of $\mathcal{R}_{\models}$. Let $S \in \mathcal{P}(C)$ be a set for a stack of FAIRification characteristics that are typically domain specific. Each stack constitutes a FAIRified environment.

Let $m : (D \times (\mathcal{P}(C) \setminus \{\varnothing\})) \rightarrow \mathcal{P}(\mathcal{R}_{\models})$ be a mapping function that returns the power set of all satisfaction relations between digital resources and corresponding FAIRification characteristics. All digital resources associated with all elements of a FAIRification stack S_k, with index $k \in \mathbb{N}^+$, are aligned with this FAIRified environment and effectively constitute a data space denoted as a set $\mathfrak{S}_{S_k}$, which is defined as:

$$\mathfrak{S}_{S_k} \coloneqq \bigcup_{d_i \in D} m(d_i, S_k) = \bigcup_{d_i \in D} \left\{ d_i \mid \forall c \in S_k,\ d_i \models c \right\} \subseteq D. \tag{3.1}$$

The set of overlaps between k aligned data spaces that fulfill the same FAIRification characteristics is indicated by:

$$\mathfrak{S}_{Intersec.} \coloneqq \left\{ \mathfrak{S}_{\bigcap_{k \in I} S_k} \;\middle|\; I \subseteq \mathbb{N}^+,\ |I| \geq 2 \right\}. \tag{3.2}$$

■

The output of Equations (3.1) and (3.2) can be visualized using a Venn diagram for an illustration of the concept of FAIRified data spaces.

Figure 3.1 depicts a minimal example, where the sets are:

$$D = \{d_1, d_2, d_3, d_4, d_5, d_6\}, \qquad C = \{c_1, c_2, c_3\},$$

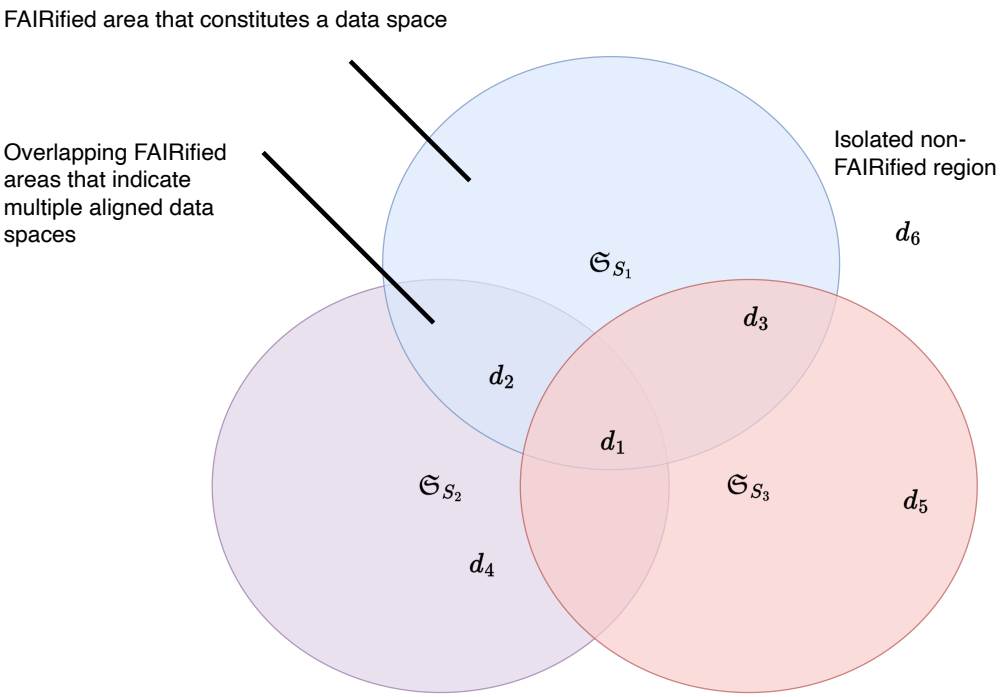


Figure 3.1.: Visualization of partially aligned FAIRified data spaces using a Venn diagram.

with:

c_1 = *digital resources have a license*,
c_2 = *metadata resources are described using the Dublin Core standard*,
c_3 = *digital resources are associated with SKOS-based controlled vocabularies*,

and assuming:

$$(d_1, c_1), (d_1, c_2), (d_1, c_3) \in \mathcal{R}_{\models}, \quad (d_2, c_1), (d_2, c_2) \in \mathcal{R}_{\models},$$
$$(d_3, c_1), (d_3, c_3) \in \mathcal{R}_{\models}, \quad (d_4, c_2) \in \mathcal{R}_{\models},$$
$$(d_5, c_3) \in \mathcal{R}_{\models}, \quad \nexists c \in C : (d_6, c) \in \mathcal{R}_{\models}.$$

The overlapping regions of the Venn diagram, i.e., the degree of alignment between data spaces, strongly depend on the considered stack of FAIRification characteristics and available digital resources. These overlaps represent only a context-dependent snapshot at a given point in time and are typically associated with specific domains. Whilst generic FAIRification characteristics tend to increase overlap, domain-specific, and thus interoperability-impairing characteristics lead to growing divergence. As a result, the shared, interoperable regions across data spaces are typically minimal or even non-existent in large data ecosystems.

3.3.2. Introduction of a Meta-level

We propose achieving alignment between data spaces by implementing a domain-independent meta-level abstraction layer based on FDOs. To introduce this new aspect, the notion of data space alignment (s. Definition 1) is extended as follows:

Definition 2 (Meta-Level) Let $D, C, \mathfrak{S}_{S_k}$ and $\mathcal{R}_{\models}$ be as given in Definition 1.

Let $\mathcal{M}$ be a set of meta-FAIRification characteristics that, taken together, realize one or more FAIRification characteristics in C. $\mathfrak{S}_{\mathcal{M}}$ denotes the meta-level abstraction layer, i.e., the environment that is FAIRified through $\mathcal{M}$. Let $\rho : \mathfrak{S}_{S_k} \rightarrow \mathfrak{S}_{\mathcal{M}}$ be a mapping function that assigns to each digital resource fulfilling one or more FAIRification characteristics its representation in $\mathfrak{S}_{\mathcal{M}}$ such that:

$$\mathfrak{S}_{\mathcal{M}} := \{\rho(d_i) \mid d_i \in \mathfrak{S}_{S_k}\}. \tag{3.3}$$

Let F be the set of all FDOs, which is used to implement $\mathfrak{S}_{\mathcal{M}}$. The satisfaction relation $\mathcal{R}_{\models^{\mathcal{M}'}} \subseteq F \times C$, with $f \models^{\mathcal{M}'} c \iff (f, c) \in \mathcal{R}_{\models^{\mathcal{M}'}}$, states that f satisfies c by means of $\mathcal{M}' \subseteq \mathcal{M}$. Each digital resource within a data space is thus effectively mapped to an FDO in the meta-level abstraction layer:

$$\forall f \in F \; \exists d_i \in \mathfrak{S}_{S_k} \; \exists c \in C \; \exists \mathcal{M}' \; s.t. \; \rho(d_i) = f \wedge f \models^{\mathcal{M}'} c. \tag{3.4}$$

■

This results in an indirect alignment of data spaces through the representation of digital resources and their FAIRification characteristics as FDOs, as illustrated in Figure 3.2.

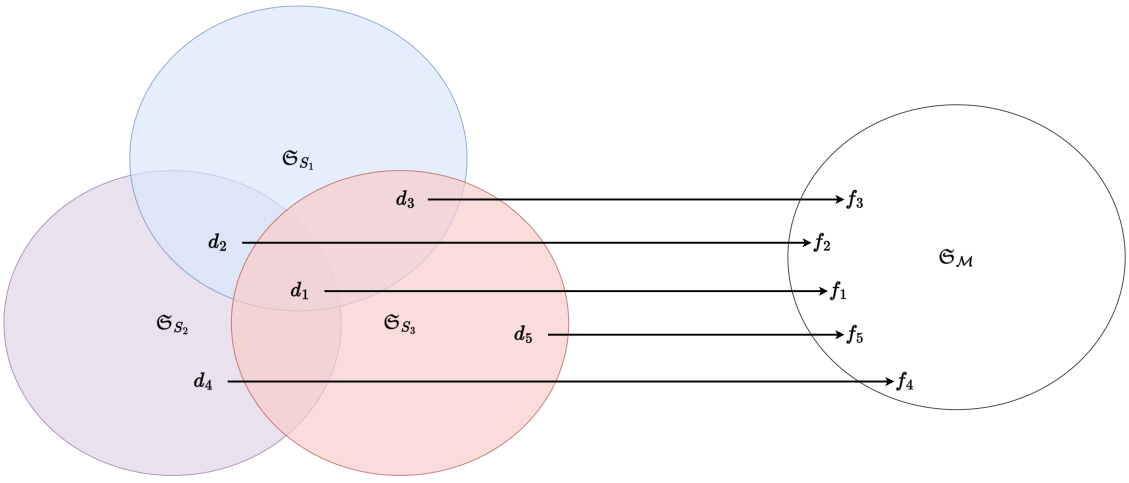


Figure 3.2.: Illustration of fully aligned FAIRified data spaces by the projection to a meta-level.

3.3.3. FDO Data Model Formalization

To realize the meta-level envisioned in Definition 2, we formalize a core data model defining the FDO data structure through a set of assertions and design principles adopted from related disciplines.

As a premise, we assume that one or more digital resources are stored in a permanent (possibly distributed) data storage system that is part of a data space as in Definition 1, from which their bit sequences can be accessed. As a basis for the formalization and overview, the conceptual core data model (non-conventional notation used) illustrated in Figure 3.3 is used, which extends the DO core data model (s. Section 2.4.2).

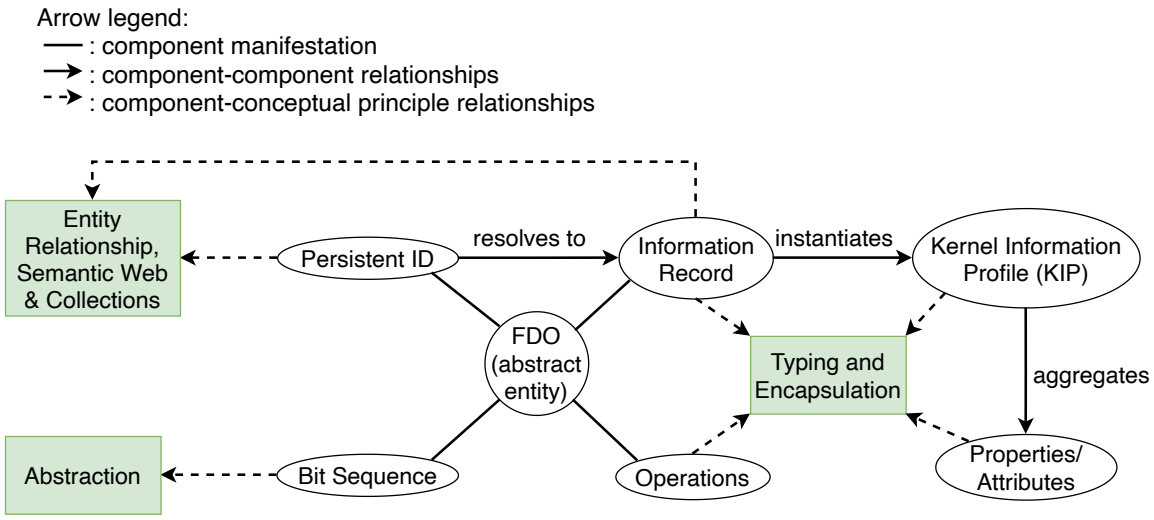


Figure 3.3.: The conceptual FDO core data model.

3.3.3.1. Object-Oriented Programming Principles

In our model specification, FDOs follow a conceptual analogy with OOP, implementing principles such as typing, abstraction, and encapsulation.

Definition 3 (Basic type system) Let Γ be the FDO typing environment with a finite mapping from variable names to their types, i.e., PITs (s. Section 2.4.3.2) that are registered in a DTR (s. Section 2.4.3.1). Γ is a well formed (w.f.) environment, since the collection of variables declared in it, i.e., its domain $dom(\Gamma)$, is finite and there are no duplicate bindings for every unique and valid type $(\tau_1, \tau_2, \ldots)$.

We use the following syntax for the Types (written in capitals) and terms that are assigned to a Type:

Nat ::= Nat Type (for numerals) τ ::= PIT B ::= Generic Type
a ::= labeled KIP property term m ::= Numeral term v ::= PIT value term

We have the following typing judgments, corresponding to the standard judgments and rules defined in [19] and applied to the notion of PIT and Nat Type (s. Table 1 for notation):

$\Gamma \vdash \diamond$ (w.f. environment, implied)
$\Gamma \vdash Nat$ (w.f. Nat Type of natural numbers)
$\Gamma \vdash \tau$ (w.f. PIT)
$\Gamma \vdash Ref(\tau)$ (w.f. Ref of τ)
$\Gamma \vdash Nat \times B$ (w.f. Nat Product Type)
$\Gamma \vdash Nat \rightarrow Ref(\tau)$ (w.f. Nat to Ref Function Type)
$\Gamma \vdash m : Nat$ (w.f. numeral of Nat Type)
$\Gamma \vdash v : \tau$ (w.f. value of PIT)
$\Gamma \vdash Array(\tau) \hat{=} Nat \times (Nat \rightarrow Ref(\tau))$ (w.f. PIT Array Type)
$\Gamma \vdash array(m, v) : Array(\tau)$ (w.f. value of PIT Array Type)

■

A KIP (s. Section 2.4.3.3) is hierarchically structured and, therefore, aggregates a set of PITs for which it specifies the cardinality.

A KIP Type is then a set of properties (equivalent to a Record Type) that are assigned an Array Type:

$$\frac{\Gamma \vdash Array(\tau_1)_1 \ldots \Gamma \vdash Array(\tau_n)_n}{\Gamma \vdash \{a_1 : Array(\tau_1)_1 \ldots a_n : Array(\tau_n)_n\}} \text{(Type KIP) [with } a_i \text{ distinct]}$$

A record can be matched against this KIP Type by the following rule, analogous to the conventional *Type Record* and *Val Record* rules in [19]:

$$\frac{\Gamma \vdash array(m_1, v_1)_1 : Array(\tau_1)_1 \ldots \Gamma \vdash array(m_n, v_n)_n : Array(\tau_n)_n}{\Gamma \vdash \{a_1 = array(m_1, v_1)_1, \ldots, a_n = array(m_n, v_n)_n\} : \{a_1 : Array(\tau_1)_1, \ldots, a_n : Array(\tau_n)_n\}} \text{(KIP Instance Record)}$$

To restrict the cardinality to 1, i.e., when a property is not repeatable, we define a type rule for a Finite Set Type of exactly m elements:

$$\frac{\Gamma \vdash m : Nat}{\Gamma \vdash Fin(m)} \text{(Type Fin Set)}$$

For $m = 1$, we define an Array Type with cardinality 1:

$$Array_{\text{Card.1}} \hat{=} Fin(1) \times Ref(\tau),$$

for which the same rules *Type KIP* and *KIP Instance Record* can be defined with $array(1, v_i)_i$. Similarly, these rules can be defined in mixed form, where the i^{th} element in $a_1, \ldots, a_n$ can be typed either by $Array(\tau_i)_i$ or $Array_{\text{Card.1}}(\tau_i)_i$.

Proof trees for the entire derivation are provided in Appendix Section A.4.1. Note that this is an abstract formalization of how KIP Types are defined in general for any set of n PITs. A concrete FDO type system would then define all KIP Types that are valid under Γ by the specification of the corresponding DTR. For example, a concrete KIP Type may be of the following form, where:

τ_1 ::= PIT for license which is obligatory and non-repeatable,

τ_2 ::= PIT for checksum which is obligatory and non-repeatable,

τ_3 ::= PIT for contact which is non-obligatory and repeatable,

$\Gamma \vdash \{license : Array_{\text{Card.1}}(\tau_1), checksum : Array_{\text{Card.1}}(\tau_2), contact : Array(\tau_3)\}$.

The record of the form $\{license = array(m_1, \tau_1)_1, checksum = array(m_2, \tau_2)_2, contact = array(m_3, \tau_3)_3\}$[1] can then be checked against this KIP Type. To cover all possible valid constellations of a given KIP that is registered in a DTR, 2^N KIP Types have to be defined for N non-obligatory properties. For example, since the *contact* property is non-obligatory, another KIP Type must be defined where this property is absent, such that a record without it is also valid in the type system.

Similar to a type system in a typed programming language, the defined typing judgments and rules for the FDO components provide a typed metadata language that can be used to validate well-typed FDO information records. This supports syntactic well formedness and potentially semantic correctness of FDO attributes that are typed by KIP Type properties, ensuring a consistent overall information-record structure for reliable handling and reducing the likelihood of errors, similar to typechecking. The type system is applied to the FDO information record (independent of its concrete serialization format) when the KIP is instantiated, and is enforced by services that create and validate FDO instances.

Definition 4 (KIP instantiation) Let F be the set of all FDOs, where each $f \in F$ depends on an information record $(R_f)_{f \in F}$, $\mathcal{I}$ be the set of all unique PIDs, and P be the set of unique KIPs. We define $R_{all} := \{R_f \mid f \in F\}$ as the set of all R_f and $\mathsf{instantiate} : \mathcal{I} \times P \to R_{all}$ as a mapping function. A KIP is effectively instantiated by an information record if:

$$\forall f \in F \; \exists! i \in \mathcal{I} \; \exists! R_f \in R_{\text{all}} \; s.t. \; \exists p \in P \; \big(\mathsf{instantiate}(i, p) = R_f\big). \tag{3.5}$$

By design, each FDO can only be associated with one information record that instantiates exactly one KIP:

$$\forall i \in \mathcal{I} \; \forall R_f \in R_{\text{all}} \; \forall p, p' \in P \; \big(\mathsf{instantiate}(i, p) = R_f \wedge \mathsf{instantiate}(i, p') = R_f\big) \implies p = p'. \tag{3.6}$$

■

KIPs and their properties are therefore connecting the FDO to the type system, allowing to determine if its contents are well-typed. The instantiated KIP realizes abstraction, which provides a uniform representation and hides the complexity of bit sequence handling, and encapsulation by bundling a set of common FDO attributes through aggregation.

[1] In an actual record, properties are represented by the PID of their PIT instead of a human-readable name.

Definition 5 (Validation of information record) Let A_p be the set of typed properties in a KIP $p \in P$ and let $A_m \subseteq A_p$ be the set of mandatory kernel information properties, which contains the elements $\{a_1, \ldots, a_6\}$ that are semantically equivalent to: {*Kernel Information Profile Reference, License, Checksum, Digital Object Location, Creation Date, Digital Object Type*}, respectively. Let there be the mapping function $\mathrm{id} : A_p \rightarrow \mathcal{I}$ that maps an $a \in A_p$ to its PID $\mathrm{id}(a) \in \mathcal{I}$. Denote $\tilde{A}_{R_f} \subseteq K \times V_{arr}$ as the set of attributes in an FDO information record forming key-value pairs from the set K of all record keys and V_{arr} of all value arrays that can be assigned to a typed property in a KIP. The following mapping function yields the result of typechecking:

$$\mathrm{typed} : \tilde{A}_{R_f} \times A_p \rightarrow \{true, false\}.$$

For each $R_f \in R_{all}$ and $p \in P$ for which Equations (3.5) and (3.6) hold, the information record of an FDO is well-typed if:

$$\forall (k,v) \in \tilde{A}_{R_f}\ \exists a \in A_p\ \exists! i \in \mathcal{I}\ \text{ s.t. } k = i\ \wedge \mathrm{id}(a) = i \wedge \mathrm{typed}\big((k,v), a\big) = true, \tag{3.7}$$

$$\forall a \in A_m\ \exists (k,v) \in \tilde{A}_{R_f}\ \exists! i \in \mathcal{I}\ \text{ s.t. } k = i \wedge \mathrm{id}(a) = i \wedge \mathrm{typed}\big((k,v), a\big) = true. \tag{3.8}$$

■

With Equations (3.5) to (3.8), the type system is validated before the unique information record is permanently registered, which completes object instantiation. The function typed thereby checks the type rule *KIP Instance Record* (s. Definition 3) in combination with the schema extraction algorithm for PITs (s. Algorithm 1 in Section 2.4.3.2), ensuring syntactic correctness of the record. On the other hand, the semantic correctness of the content can be only partially granted by type validation, as it mainly depends on the PIT specification granularity (e.g., the use of *enumeration* or *regex* patterns).

Definition 6 (Data access, FDO Type and operability.) Let $\mathcal{P}(\tilde{A}_{R_f})\backslash\{\varnothing\}$ be the set of all non-empty subsets of $\tilde{A}_{R_f}$ and B the set of all finite bit sequences. Then:

$$\mathcal{A}_f \coloneqq \mathcal{P}(\tilde{A}_{R_f})\backslash\{\varnothing\} \qquad B \coloneqq \bigcup_{k \in \mathbb{N}^+} \{0,1\}^k.$$

Given at least $DigitalObjectLocation \mathrel{\hat{=}} a_4 \in A_m$ (s. Definition 5), the bit sequence can be accessed by the partial mapping function:

$$\mathrm{access} : \bigcup_{f \in F} \tilde{A}_{R_f} \rightharpoonup B.$$

Then we have the following:

$$\forall f \in F\ \exists! a_4 \in \tilde{A}_{R_f}\ \exists! b_f \in B\ \ s.t.\ \mathrm{access}(a_4) = b_f. \tag{3.9}$$

Denote T as the set of FDO Types that characterize FDOs as a whole, and O as the set of non-trivial operations besides Create, Read, Update, and Delete (CRUD). For each $f \in F$,

let $O_f \subseteq O$ be the set of operations that can be performed on f. We define the partial mapping functions:

$$\text{determineFDOType} : \bigcup_{f \in F} \mathcal{A}_f \rightharpoonup T, \qquad \text{inferOperations} : T \rightharpoonup \mathcal{P}(O).$$

Then, the set of all possible operations for a given FDO f, depending on its type, is:

$$O_f = \bigcup_{S \in \mathcal{A}_f} \text{inferOperations}\big(\text{determineFDOType}(S)\big). \tag{3.10}$$

Operations may or may not be applicable directly to the FDO bit sequence, depending on its accessibility condition as per Equation (3.9). ■

Additional encapsulation aspects (s. also Definition 4) are fulfilled, because each FDO provides an interface to its bit sequence that underlies its own access protocol and introduces a set of non-trivial operations via the FDO Type(s). The FDO Type itself is a technical, syntactic, and semantic characterization of an FDO based on the kernel metadata and the bit sequence. Since an FDO as a whole may have more than one characterization, multiple FDO Types can in principle exist for the same object (s. Section 3.4.2.3). To infer these FDO Types, varying kernel metadata may be considered, possibly including the bit sequence.

3.3.3.2. Validation of Model Semantics

Whilst the FDO core data model enforces technical and syntactical constraints through typing and assertions, it does not by itself ensure the intended model semantics. In particular, the type system by itself does not define the criteria for the properties to be in- or excluded from the KIP. Adding use-case and domain-specific properties $a \notin A_p$ to a KIP $p \in P$ increases the expressive power of the information record, but conflicts with the goal of keeping it minimal and generic. Governance mainly relates to the set A_m, and a validation mechanism is needed during the design of an FDO type system to determine which information should be included directly in the record and which should be represented by an individual FDO and referenced. Thus, we define an activity diagram-based model semantic validation step as depicted in Figure 3.4.

Paradigm: besides the RDA kernel metadata, an FDO information record contains only attributes for the technical and syntactic information given by sets A_p and A_m needed to read and interpret the bit sequence of the digital resource x, or to reference additional specific information. This use-case-specific information is contained in FDOs that represent additional metadata resources y, and can be returned by associated operations.

The *Helmholtz KIP* [133], which is based on the RDA KIP, satisfies the requirements of the mandatory kernel information property set A_m. This profile is adopted and extended according to the procedure defined in Figure 3.4, i.e., with $\mathit{hasSchema} \in A_p \setminus A_m$, to ensure a correct syntactic interpretation of the represented digital resource. An implemented version of the Helmholtz KIP has been registered at the Handle Registry[2]. The entire

[2] https://hdl.handle.net/21.T11148/b9b76f887845e32d29f7?locatt=view:ui

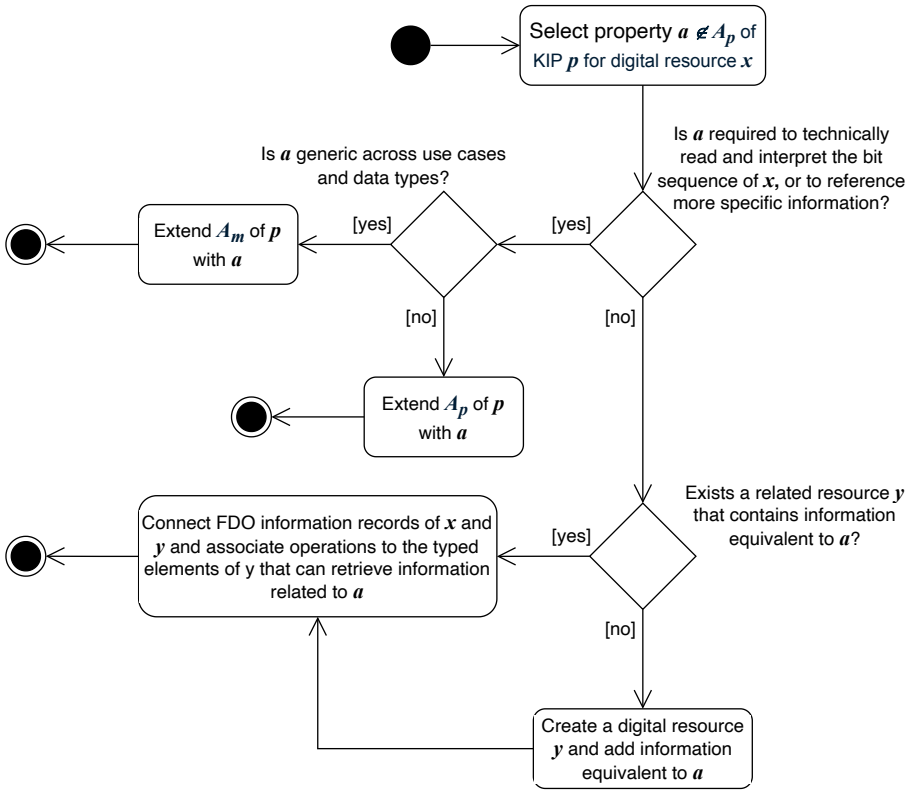


Figure 3.4.: Activity diagram for validating the FDO core data model intention, i.e., model semantics.

Helmholtz KIP and all its properties used in the examples of the subsequent sections are summarized in Table 3.1.

Table 3.1.: Typed properties of the extended (*) Helmholtz KIP.

Name	Obligatory	Cardinality	Basic PID-Information Type
kernelInformationProfile	true	1	Handle-Identifier-ASCII
digitalObjectLocation	true	1..*	URL
dateCreated	true	1	date-time-rfc3339
dateModified	false	0..1	date-time-rfc3339
license	true	1	URL
digitalObjectType	true	1	Handle-Identifier-ASCII or media-type-IANA
checksum	true	1	checksum-string
version	false	0..1	version-number
hasMetadata	false	0..*	Handle-Identifier-ASCII
isMetadataFor	false	0..*	Handle-Identifier-ASCII
topic	false	0..*	URL
contact	false	0..*	URL
hasSchema*	false	0..1	URL

3.3.3.3. Knowledge Representation Principles

To model FDOs as an interconnected graph on the web, we incorporate principles from the Entity–Relationship (ER) Model, Linked Data (LD), and collections. This is also a crucial aspect to realize the model semantics mechanism described in Section 3.3.3.2.

All FDOs satisfy the principles of **ER modeling**:

- **Distinctive identity**: by Equations (3.5) to (3.7), every FDO is assigned a globally unique PID.
- **Typed attributes**: information records provide attribute–value pairs, corresponding to ER attributes as formalized in Equation (2.2).
- **Relationships**: typed attribute keys whose values are other PIDs induce sets of n-ary relationships among entities as defined in Equation (2.1). The role semantics, e.g., "hasMetadata", are given by the PIT definitions of such keys.

Definition 7 (FDO–FDO entity relationships) Let $\mathcal{E}$ be the set of all entities on the web that have some identifier, which could be a PID in the set of PIDs $\mathcal{I}$. The set of FDOs F is a subset $F \subset \mathcal{E}$. Given that some elements in the set V of values that occur in an element of the set V_{arr} of value arrays are equivalent (on the string level) to the PIDs in $\mathcal{I}$, assume $V \cap \mathcal{I} \neq \varnothing$ and define $V_{\mathcal{I}} \coloneqq \mathcal{I} \cap V$.

We consider the following partial mappings:

$$\mathsf{identifier} : \mathcal{E} \rightharpoonup \mathcal{I}, \qquad \mathsf{hasValue} : \bigcup_{f \in F} \tilde{A}_{R_f} \rightharpoonup V_{\mathcal{I}} \qquad (\tilde{A}_{R_f} \mathrel{\hat{=}} \text{record attributes}).$$

Every $f \in F$ is (at least) related to those entities $e \in \mathcal{E}$ whose identifiers occur as values of mandatory attributes in A_m. Whenever the PID of an FDO $f' \in F$ appears as the value of some attribute of $f \in F$, we assert a relationship between f and f':

$$\forall f, f' \in F \; \exists a \in \tilde{A}_{R_f} \; s.t. \Big(\mathsf{hasValue}(a) = \mathsf{identifier}(f') \Big) \;\Rightarrow\; f \; \mathsf{relatesTo} \; f'. \tag{3.11}$$

■

Consequently, FDOs can be used to perform typical ER operations, such as navigation across related entities and composition of higher-level views in distributed environments. As an example, the FDO-FDO relationships according to Equation (3.11) can be applied to data and metadata FDOs, as depicted in Figure 3.5.

The FDO-FDO relationships, being effectively PID triples representing web data, can be used to construct a graph from the plain record structure. This works analogously to **LD**, where RDF triples interlink resources via URLs as shown below.



Figure 3.5.: ER principle in FDO-FDO relationships for data and metadata.

Definition 8 (Graph structure) Let $\tilde{F} \subset \mathcal{I}$ be the set of PIDs that identify FDOs in the set F, $\tilde{K} \subset \mathcal{I}$ be the set of PIDs that identify typed attributes, $\mathcal{T} \subseteq \tilde{F} \times \tilde{K} \times \tilde{F}$ be the set of PID triples and L be a set of labels. The mapping function

$$\ell : \mathcal{T} \to L,$$

maps each triple $t \in \mathcal{T}$ to a label $l \in L$.

Define the PID hypergraph

$$G_H := (V_H, E_H), \tag{3.12}$$

by $\quad V_H := \tilde{F} \cup \tilde{K}, \quad E_H := \left\{ \{\tilde{f}, \tilde{k}, \tilde{f}'\} \;\middle|\; (\tilde{f}, \tilde{k}, \tilde{f}') \in \mathcal{T} \right\},$

and define the FDO digraph

$$G_D := (V_D, E_D), \tag{3.13}$$

by

$$V_D := \tilde{F}, \quad E_D := \left\{ (\tilde{f}, l, \tilde{f}') \in \tilde{F} \times L \times \tilde{F} \;\middle|\; \exists \tilde{k} \in \tilde{K}\ s.t. (\tilde{f}, \tilde{k}, \tilde{f}') \in \mathcal{T} \wedge \ell\big((\tilde{f}, \tilde{k}, \tilde{f}')\big) = l \right\}.$$

■

Based on a PID hypergraph G_H, an FDO digraph G_D can be derived as depicted in Figure 3.6.

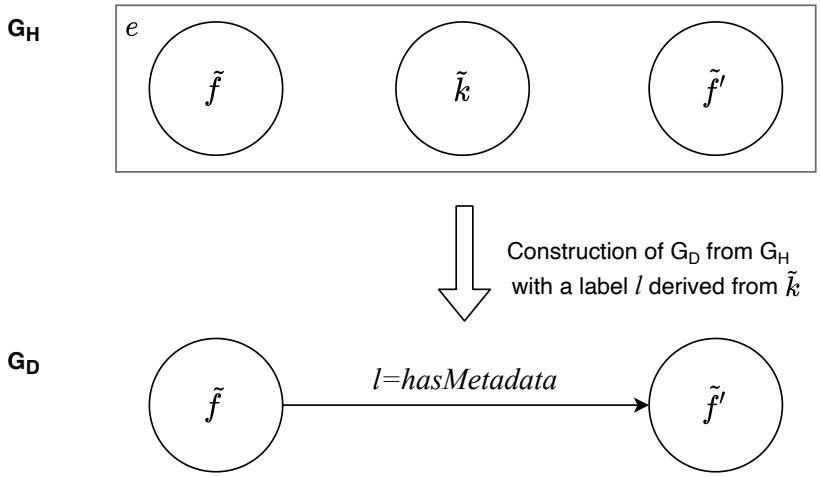


Figure 3.6.: Example of a PID hypergraph G_H with $V_H = \{\tilde{f}, \tilde{k}, \tilde{f}'\}$ and $E_H = \{e\} = \{\{\tilde{f}, \tilde{k}, \tilde{f}'\}\}$, from which an FDO digraph G_D with $V_D = \{\tilde{f}, f'\}$ and $E_D = \{(\tilde{f}, hasMetadata, \tilde{f}')$ is constructed.

Each edge of such a digraph carries semantics of its relationship type and is a resolvable link to another FDO. Since each FDO is associated with an information record, additional machine-interpretable information is available beyond the link itself. PID triples of the form $(\tilde{f}, \tilde{k}, \tilde{f}')$ are used to define the FDO digraph with labeled edges following Equation (3.13). The translation to RDF, illustrated in Figure 3.7, is then:

$$(\tilde{f}, \tilde{k}, \tilde{f}') = \left(\text{PID}(\text{FDO}_{\text{sub}}), \text{PID}(\text{attribute's key}), \text{PID}(\text{FDO}_{\text{obj}})\right) \hat{=} (\text{subject, predicate, object}),$$

where $\tilde{f}$ is the PID of the subject FDO, $\tilde{k}$ is the PID of a semantically defined attribute key from its information record, and $\tilde{f}'$ is the PID of the object FDO. In addition to RDF, FDO digraphs enable resource persistency, strict typing, and machine-actionable processing. In this way, FDOs may be seamlessly mapped to and from Semantic Web environments, profiting from established methods like reasoning and knowledge graph representation.

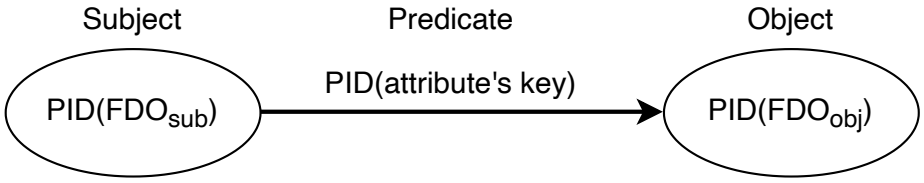


Figure 3.7.: The conceptual model of PID triples based on the FDO's ER modeling in analogy with RDF triples, connecting an FDO_{sub} with an FDO_{obj} by a typed attribute key working as predicate.

Furthermore, FDOs fulfill the requirements for **Research Data Collections** as given by RDA recommendations.

Consider a collection consisting of elements $\epsilon_1, \ldots, \epsilon_n$. If each ϵ_i is represented by an FDO, then the set of these FDOs represent the collection with descriptive metadata and references to its members. An FDO representing ϵ_i can reference an FDO representing ϵ_j (s. Equation (3.11)). If all elements $\epsilon_1, \ldots, \epsilon_n$ are represented in this way and are properly connected, then the induced FDO digraph (s. Equation (3.13)) is a self-contained abstract representation of the entire collection.

3.3.4. Base Services and Infrastructure

For implementing our model, we use an infrastructure consisting of the following existing base services:

- **Handle Registry** [152]: a PID service based on the Handle System that assigns PIDs and registers the corresponding information records that can be resolved via the Handle API protocol.
- **ePIC Data Type Registry (DTR)** [153]: an object store based on the Cordra system [154] that registers and stores PIT and KIP specifications using JSON schemas in accordance with the type system in Definition 3.

- **Typed PID Maker (TPM)** [155]: an implementation of a PIT API service that can be instantiated and configured with the Handle Registry and the ePIC DTR. It abstracts the Handle API interface for registering PIDs for FDO information records and performs a procedure that asserts Equations (3.5) to (3.8), thus effectively creating FDOs by KIP instantiation.

3.3.5. Example Datasets

Two example datasets from different domains are employed for model implementation to demonstrate its domain independent usability.

The first dataset originates from **energy research**. Datasets in this domain cover extensive, high-frequency, or high-resolution data, enabling researchers to analyze patterns, improve efficiency, optimize energy systems, and forecast demand. The “Thermal Bridges on Building Rooftops Dataset” (TBBR) [156] is useful for identifying heat loss points on rooftops and was used for energy conservation studies using AI methods [157]. It is composed of several types of digital resources, comprising drone image sets, Common Objects in Context (COCO) annotation files, metadata files based on the Spatio Temporal Asset Catalogs (STAC), and Frictionless Data standards [11], which are candidates for a data research collection.

The second dataset comes from the domain of **digital humanities (DH)**, where controlled vocabularies (CVs) such as ontologies and thesauri are used to analyze cultural, historical, and linguistic phenomena. CVs usually represent a variety of terminologies, expressed in multiple languages and data models, such as the Simple Knowledge Organization System (SKOS). Therefore, a typical task in this field is ontology matching, which helps align and integrate such heterogeneous CVs. The "Ontology Matching Benchmark Dataset for Digital Humanities" [12], hereafter the OntoMatch, is a multilingual SKOS-based dataset assembled from multiple CVs and is intended to support the advancement of ontology matching systems.

3.4. Evaluation and Results

In this section, we first assess and evaluate how our model fulfills the FAIR principles criteria by meta-FAIRification characteristics as defined in Section 3.3.2. We then provide a proof of concept for the proposed FDO meta-level constitution by implementing our model based on the defined assertions. Finally, a comparative analysis with existing FDO implementation strategies is performed to contextualize our model.

3.4.1. Fulfillment of Meta-FAIRification Characteristics

We evaluate how the FDO core data model formalized in Section 3.3.3 realizes a meta-level $\mathfrak{S}_{\mathcal{M}}$ according to the set $\mathcal{M}$ with **Meta-FAIRification Characteristicss (MFCs)** (s. Equations (3.3) and (3.4)). Thus, we assume that all the model assertions in Equations (3.5) to (3.11) hold and that the intended model semantics have been validated using the activity diagram in Figure 3.4. We assess the MFCs providing conformance arguments for each of the FAIR criteria (s. Appendix Tables A.1 to A.3) as originally formulated (**FAIR**), as (partly) re-defined for research software (**FAIR4RS**), and as extended for semantic interoperability (**FAIR2.0**). The summary is presented in Table 3.2.

Quantitative assessment tools such as F-UJI [118] or FAIRSharing [158], often used to assign a FAIRness score to datasets, are not appropriate in our case: such tools are based on community-specific implementations of FAIR principles and are typically tailored to specific repositories. Apart from that, it is not our attempt to provide here absolute FAIR metric scores anyway, but to assess how the MFCs address FAIR criteria on the meta-level. Some of these FAIR criteria are *explicitly* fulfilled by the implementation of our FDO core data model, while others are *implicitly* addressed by the FAIRification characteristics of the represented digital resources in the community-specific implementations. This distinction is indicated in the **Exp/Imp** and in the **MFC Interpretation** columns of Table 3.2. Under this premise, our model effectively provides a way to abstract and leverage individual community implementations of FAIR criteria at the meta-level $\mathfrak{S}_{\mathcal{M}}$. Therefore, we argue that digital resources can be used according to their native FAIRification characteristics through their uniform FDO representation.

Table 3.2.: Connecting Meta-FAIRification Characteristics (MFC) to FAIR criteria.

MFC	FAIR	FAIR4RS	FAIR2.0	Exp/Imp	MFC Interpretation
MFC1	F1	F1–F1.2	–	explicit	Each FDO has a unique PID; enables findability.
MFC2	F2–F3	F2–F3	–	explicit + implicit	Mandatory set A_m includes access to bit sequence; FDO graph links related data and metadata resources.
MFC3	F4	F4	–	explicit	FDO records are registered in PID services; indexed and searchable.
MFC4	A1–A1.2	A1–A1.2	A1.3	explicit (A1–A1.2), implicit (A1.3)	FDO record retrieval via standard protocols → uniform bit sequence access; supports authn./authz. mechanisms usable in General Data Protection Regulation (GDPR)-compliant processing contexts.
MFC5	A2	A2	–	explicit	Persistent preservation of information records via PID system policies; kernel and referenced metadata remain even if data is gone.
MFC6	I1	I1	–	explicit + implicit	Knowledge via formal PITs in DTR; governed by a typed metadata language; abstraction of knowledge within represented resources.
MFC7	I2	I2	I2.1–I2.3	explicit (I2), implicit (I2.1–I2.3)	Controlled vocabularies in PIT value spaces; domain vocabularies abstracted and linkable as in MFC2.
MFC8	I3	I3	–	explicit	PID-triples relate FDOs and other entities; relationships derivable; aligns with MFC2.
MFC9	–	I1, I2	–	explicit	Software as an FDO related via entity traits, or embedded as type-associated operations.
MFC10	–	–	I4; 4.1–4.4	explicit (I4), implicit (4.1–4.4)	Propositional interoperability when FDOs instantiate same KIP; A_m enables a baseline for minimum interoperability.
MFC11	R1	R1	–	explicit	Profiles and typed attributes via PIT/KIP hierarchies support reuse through shared meaning.
MFC12	R1.1	R1.1	–	explicit	License information recommended in KIPs and included in A_m.
MFC13	R1.2	R1.2	–	explicit + implicit	Provenance in information record per RDA KIP recommendations; or associated via linked resources as in MFC2.
MFC14	R1.3	R1.3	–	explicit + implicit	Community standards in kernel metadata via extended profiles; otherwise linked as associated resources.
MFC15	–	–	R1.4	explicit + implicit	Fulfilled by represented metadata FDO or directly via high-certainty kernel metadata curated by expert groups (e.g., in RDA).

3.4.2. Model Implementation and Proof of Concept

To provide a proof of concept that the FDO core data model formalized in Section 3.3.3 enables the meta-level as in Definition 2, we implement and apply it in a domain-agnostic use case, employing the two datasets described in Section 3.3.5. Both datasets originate from AI-driven optimization of domain-specific problems, (somehow) address FAIR principles, and are potentially valuable for other AI-related tasks. However, due to the different origins, their FAIR-related machine-actionability is not equally usable in their native state, and preliminary (meta)data assessment should be performed. For example, the license of each individual dataset should be verified to determine whether its (re)use is allowed. Our model represents such datasets as FDOs, providing the abstraction needed to ease and automate the assessment.

The original datasets are first described using the simplified data space model (s. Definition 1) and assessed with respect to their FAIRness. Then, each digital resource of both datasets is represented as FDOs according to our FDO core data model, which inherently applies the MFCs. This results in a uniform machine-interpretable representation of both datasets, enabling machine-actionable decision making.

3.4.2.1. Assessment of Data Space Alignment by FAIRification

The TBBR dataset contains 18 individual digital resources, whilst the OntoMatch dataset consists of a single resource containing all files. The elements of the set $D = \{d_1, \ldots, d_{19}\}$ are:

$$
\begin{aligned}
d_1 &= \textit{drone image set 1}, & d_2 &= \textit{annotation file 1},\\
d_3 &= \textit{annotation file 2}, & d_4 &= \textit{drone image set 2},\\
d_5 &= \textit{drone image set 3}, & d_6 &= \textit{drone image set 4},\\
d_7 &= \textit{drone image set 5}, & d_8 &= \textit{drone image set 6},\\
d_9 &= \textit{frictionless data standard file}, & d_{10} &= \textit{STAC collection file},\\
d_{11} &= \textit{STAC feature file 1}, & d_{12} &= \textit{STAC feature file 2},\\
d_{13} &= \textit{STAC feature file 3}, & d_{14} &= \textit{STAC feature file 4},\\
d_{15} &= \textit{STAC feature file 5}, & d_{16} &= \textit{STAC feature file 6},\\
d_{17} &= \textit{STAC camera file 1}, & d_{18} &= \textit{STAC camera file 2},\\
d_{19} &= \textit{VocabularyCollectionFile}.
\end{aligned}
$$

In this simple use case, we can define $D_{TBBRSet} = D \backslash \{d_{\text{VocabularyCollectionFile}}\}$ and $D_{OntoMatchSet} = \{d_{\text{VocabularyCollectionFile}}\}$. We identify the following FAIRification characteristics $C = \{c_1, \ldots, c_8\}$, not necessarily exhaustive:

c_1 = *dataset DOI*,
c_2 = *DataCite-conform record metadata*,
c_3 = *open license CC BY 4.0*,
c_4 = *creators have ORCID*,
c_5 = *data linked to external code repository*,
c_6 = *geospatial image data enriched with standard metadata, i.e., STAC*,
c_7 = *annotations made with COCO-JSON*,
c_8 = *SKOS-based vocabulary modeling*.

We refrain from explicitly associating these characteristics with the FAIR criteria that they address. In principle, the n possibilities to derive FAIRification characteristic stacks correspond to the size of the power set of C, $\mathcal{P}(C) \setminus \{\varnothing\}$. However, here we only focus on those stacks that contain those elements of C that are:

1. basic repository features, applicable to both $D_{TBBRSet}$ and $D_{OntoMatchSet}$ (S_1).
2. domain-specific characteristics only applicable to $D_{TBBRSet}$ (S_2) or $D_{OntoMatchSet}$ (S_3).

Then:

$$S_1 = \{c_1, c_2, c_3, c_4, c_5\},$$
$$S_2 = \{c_6, c_7\},$$
$$S_3 = \{c_8\}.$$

Applying Equation (3.1) to each dataset and stack, with:

$$(d_1, c_1), \ldots, (d_1, c_7), \ldots, (d_{18}, c_1), \ldots, (d_{18}, c_7),$$

$$(d_{19}, c_1), \ldots, (d_{19}, c_5), (d_{19}, c_8) \in \mathcal{R}_{\models},$$

results in:

$$\mathfrak{S}_{S_1} = D_{TBBRSet} \cup D_{OntoMatchSet},$$
$$\mathfrak{S}_{S_2} = D_{TBBRSet},$$
$$\mathfrak{S}_{S_3} = D_{OntoMatchSet}.$$

According to Equation (3.2), the overlaps are:

$$\mathfrak{S}_{S_1 \cap S_2} = D_{TBBRSet}, \qquad \mathfrak{S}_{S_1 \cap S_3} = D_{OntoMatchSet}, \qquad \mathfrak{S}_{S_2 \cap S_3} = \varnothing, \qquad \mathfrak{S}_{S_1 \cap S_2 \cap S_3} = \varnothing,$$

demonstrating that both domain-specific data spaces $\mathfrak{S}_{S_2}$ and $\mathfrak{S}_{S_3}$ are aligned with the generic and domain-agnostic data space $\mathfrak{S}_{S_1}$, but not with each other. Hence, there is a shared interoperable data space $\mathfrak{S}_{S_1}$ in which a client can handle both datasets uniformly at the FAIR repository baseline level, while the domain-specific data spaces $\mathfrak{S}_{S_2}$ and $\mathfrak{S}_{S_3}$ are disjoint and therefore require individual handling of their digital resources.

3.4.2.2. Mapping to FDO Representation

The Helmholtz KIP is instantiated on the basis of JSON records for each digital resource according to Equations (3.5) to (3.9), by employing the TPM instance described in Section 3.3.4.

This yields a total of 19 registered Handle PIDs resolving to the FDO information records (entire PID list in Appendix Section A.2.1 or in [156] for TBBR), with $\mathfrak{S}_{\mathcal{M}} = F = \{f_1, \ldots, f_{19}\}$, where $f_i \in F$ corresponds to $d_i \in D$ described in Section 3.4.2.1: we have obtained a uniform, typed, machine-interpretable representation for all the digital resources of both datasets with harmonized semantics and syntactic constraints.

Based on references to other FDOs in their information records, we can infer the following statements according to Equation (3.11):

$$f_1 \text{ relates to } f_3, \ldots, f_{18} \text{ relates to } f_{10}.$$

Since we know the semantics of these relationships through the involved PITs, all these statements can be expressed as one FDO Metadata ER model. For this example, we use the typed attributes *hasMetadata* and *isMetadataFor*, which resemble Figure 3.5. An exemplary FDO picked from the TBBR dataset, representing a drone image (f_3), and the FDO representing the OntoMatch dataset f_{19} are illustrated in Figure 3.8. The figure depicts those typed attributes that relate these FDOs to common web entities (black arrows) and to other FDOs according to the FDO Metadata ER model (blue arrows). For the original Handle record of f_3, see Appendix Figure A.1.

For the FDO digraph representation that emerges from the FDO Metadata ER model, we have:

$$\tilde{F} = \{\tilde{f_1}, \ldots, \tilde{f_{19}}\}, \qquad \tilde{K} = \{a, b \mid a \mathrel{\hat{=}} \textit{hasMetadata PIT},\ b \mathrel{\hat{=}} \textit{isMetadataFor PIT}\},$$
$$\mathcal{T} = \{(\tilde{f_1}, b, \tilde{f_3}), \ldots, (\tilde{f_{18}}, a, \tilde{f_{10}})\}.$$

Such PID-triples of FDO-FDO relationships only occur within the TBBR resource FDOs that are part of the hypergraph G_H (s. Equation (3.12)), which yields a strongly-connected FDO digraph G_D (s. Equation (3.13)), where each FDO entity has at least one connection to another FDO entity and every FDO node can be reached from any other node. Appendix Section A.2.1 lists the entire triple set and node associations with the original Handle PIDs. The entirety of these related FDOs constitutes a research data collection given that all components of the original dataset are represented in the graph and have clear semantics of their relationships. At the same time, each element of this collection has the same data structure and can be treated equally at this level.

This example shows the flexibility of the FDO core data model regardless of the original data structure: if it is granular, like the TBBR dataset, the resources can be connected on the meta-level and interpreted as a research collection while still remaining individually identifiable. In contrast, if stored in a containerized format, as in the case of the OntoMatch dataset, the contained individual resources are not directly accessible and a limited set of processes can be performed on them. In both cases, however, the uniform FDO representation of datasets enables their machine-interpretability and a path to actionability.

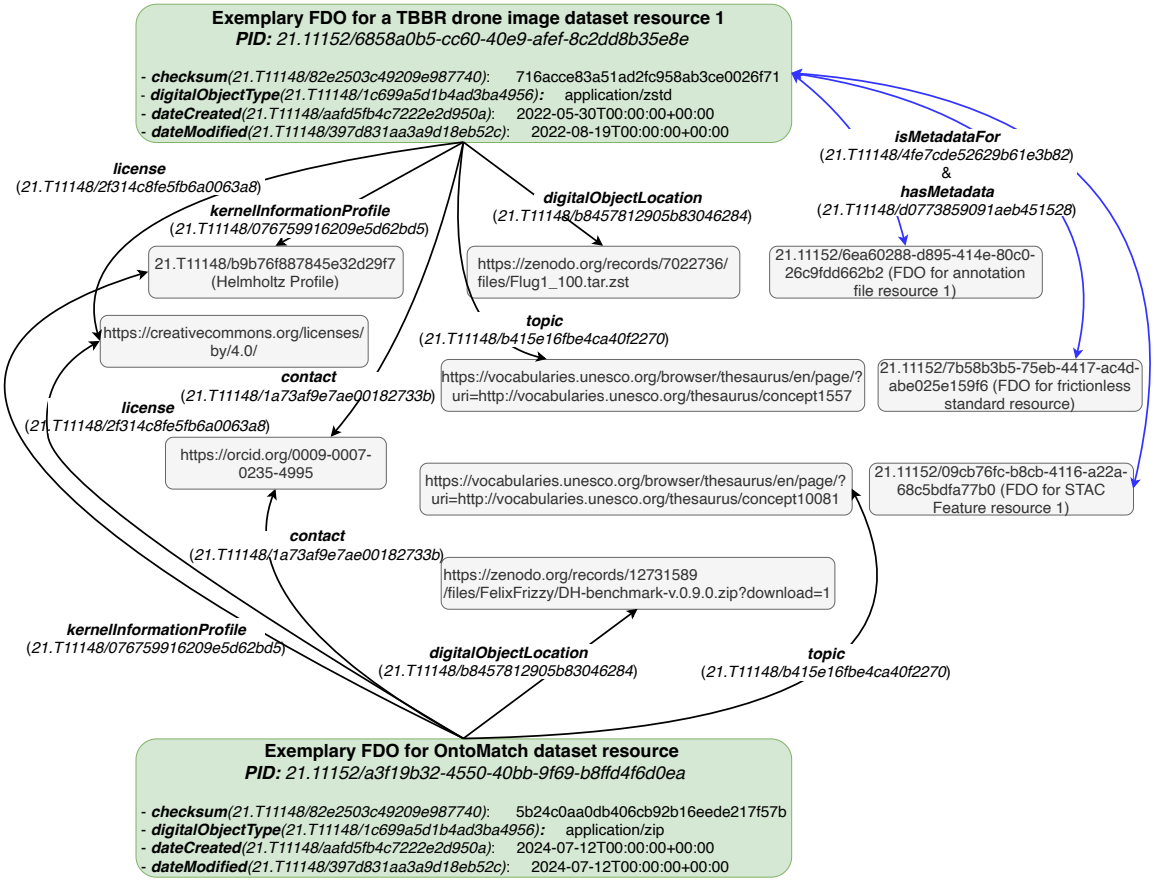


Figure 3.8.: The exemplary FDOs represented according to the formalized core data model. Black arrows: both non-referencing and referencing typed attributes, blue arrows: PID triples of FDO-FDO relationships.

3.4.2.3. Towards FDO Machine-Actionability

By representing all digital resources in a uniform and machine-interpretable structure, the set of operations O_f for each FDO f according to Equation (3.10) enables machine-actionable decisions. Such operations are essential to realize the proposed meta-level abstraction, since they are required to process an FDO and to exploit the knowledge provided by the underlying digital resources. Only through this processing can the FAIRification characteristics of a resource be leveraged by the MFCs of its representing FDO. Operations may target either the metadata in the information record or also the bit sequence, and may be inherently specific to a particular type of digital resource or more generic. A possible set of operations applicable to both example datasets is:

$O_f = \{\text{EVALUATE_LICENSE}, \text{VALIDATE_SCHEMA}, \text{GET_RELATED_FDO}, \text{GET_DIGITAL_RESOURCE}\}$ enabling the following tasks, respectively:

- evaluate the reusability of a digital resource based on its license information
- validate the syntax of a metadata resource based on its schema

- query the FDO graph based on PID-triples to identify use-case-specific metadata resources
- retrieve a digital resource from its storage system within a given data space. This operation is crucial for any subsequent action to be performed on the bit sequence

These operations are associated with elements from the set of FDO Types T that expresses the semantics of one characteristic of f, in this case: $\{$*licensed object, STAC-based schema object, FDO Entity relating object, data accessible object* $\} \subseteq T$, respectively.

In general, an advanced type system based on our model can ensure that all FDOs characterized by the same semantics through their typed attributes support the same set of operations, even across heterogeneous data spaces. For this to work out, well-defined typing mechanisms must be in place such that the FDO Type becomes well formed. These mechanisms must also guarantee that an operation is actually applicable to its target, i.e., that operation input requirements are fulfilled.

3.4.3. Comparative Analysis with Related Works

As anticipated in Section 3.2, the existing implementations of the FDO concept adopted by the different communities are not uniform, resulting in limited interoperability at the FDO level. In this section, we examine these approaches, outlining the differences, compatibility, and scope with respect to our FDO core data model. An overview of the comparative analysis is provided in Table 3.3.

Table 3.3.: Comparison of existing FDO modeling approaches based on our FDO core data model traits.

FDO Model Approach	Env.	PIT-like Typing	Profile	Ops	Bit	PIDG	DA
PIDINST	Handle	Yes	Yes	No	No	No	No
DARIAH	Handle	Yes	No	No	No	No	No
DES	Handle	Yes	Yes	No	No	No	No
RO-Crate	Web/ZIP	Partial	Yes	No	Yes	Partial	Partial
Webby FDOs	Web/HTML	Partial	Yes	Partial	Yes	Partial	Partial
Nanopubs+FDOF	KG/OWL	Partial	Yes	Partial	Yes	Partial	Partial

Legend: Env.=instantiation environment; PIT-like Typing: PIT-like type compliance (Yes/Partial/No); Profile: profile/KIP instantiated (Yes/No); Ops: explicit association of operations (Yes/No); Bit: resolvable access to the bit sequence (Yes/No); PIDG: PID-triple graph compatibility (Yes/Partial/No); DA: digital-resource agnostic intent (Yes/Partial/No).

3.4.3.1. Handle Record Implementations

We examine three Handle-based records:

- **PIDINST**: 21.T11998/0000-001A-3905-1, used for the persistent identification of instruments and described as “almost FDO-compliant” [136].

- **DARIAH**: 21.11113/0000-000B-CA4C-D, used as an FDO profile for "Legacy Repository Records" [136].
- **Digital Extended Specimen (DES)**: 10.3535/G0G-G7D-N5J, used to represent digital specimens [159].

The original Handle records are depicted in the Appendix Figures A.2 to A.4.

Type system: PIDINST uses officially PIT-typed properties and thus aligns with our model at the top level. However, attributes that should be typed by PIT compositions are represented as JSON objects with human-readable keys instead of sub-PIT identifiers. The profile reference is not a PIT identifier, so complete type validation is not possible or depends on mappings.

In the DARIAH record, elements were modeled retrospectively in a type registry, but the Handle record itself does not carry the pointer to these types. No profile is referenced, thus the originally instantiated profile cannot be directly reconstructed.

The DES record references an FDO profile that follows the PIT convention. This allows profile-based validation if the hosting registry is available. Some keys, e.g., `topicCategory`, still violate the allowed value domain, which indicates incomplete validation.

OOP/profile principles: PIDINST and DES are based on a schema that can be instantiated as a profile. The DARIAH record is not. Currently, RDA-mandated attributes were not covered by these profiles, so abstraction is weaker than in our model. None of the records contains a formal model-level definition of associated operations, nor does any of them expose an accessible bit sequence, for which reason encapsulation as required by our model is not achieved.

Entity relationships and Graph Structure: All three records contain typed links to external web entities. Since they are isolated Handle records and have no profile or model that specifies the formation of FDO–FDO PID-triples, no FDO graph emerges.

Model semantics: Both PIDINST and DES records describe physical objects (instrument and specimen, respectively), so they do not resolve to the bit sequence of a digital resource. Each record rather points to a landing page that contains metadata about the physical object, which is typically not machine-readable if no structured metadata is embedded. However, even if this is the case, the rest of the Handle record is then supposed to describe this metadata resource, not the physical object itself. The DARIAH record indeed points to a digital resource, a text file containing RDF with Turtle syntax. However, this is not made explicit in the record metadata. Another problem with all these records is that they contain use-case-specific attributes which are supposed to be contained in separate, schema-based metadata resources that can be represented as FDOs themselves.

Conclusions: All three Handle records show partial compliance with our model in terms of syntactic structure, typing, and basic entity linking. However, they do not fulfill the OOP-oriented parts or the abstraction from domain-specific details. They should therefore be considered as metadata resources about digital or physical objects, rather than FDOs

in the strict sense. An FDO pointing to such a Handle record as the bit sequence indeed remains fully compatible with our model definition.

3.4.3.2. RO-Crate

RO-Crate packages data together with metadata and can be profiled for specific community needs. This approach is based on JSON-LD/RDF and schema.org, a technology stack significantly different from the PIT-based Handle records. A comparison with our FDO core data model is therefore limited to the envisioned meta-level.

Type system: Schema.org is an extensible RDF(S) vocabulary with class hierarchies and typed properties. The typing of attributes is, therefore, native. The main difference from PITs lies in the identifier strategy and validation: schema.org uses HTTP URIs, while our model assumes PID-based, registry-backed PIT identifiers to ensure persistence and controlled value spaces. Consequently, typing in RO-Crate is more flexible, but also less comprehensive than PIT-based typing.

OOP/profile principles: An RO-crate is a container of data and corresponding JSON-LD metadata that usually follows a profile (e.g., Bioschemas *Computational Workflow*[3]). Even if such profiles are analogous to KIPs, they usually contain a mix of domain-specific and kernel metadata, deviating from the RDA recommendations. Moreover, data and domain-specific metadata are tightly coupled and there is no explicit operation model. Therefore, the abstraction and encapsulation layer between the FDO representation of a digital resource and the interpretation of its bit sequence is weaker than in our model.

Entity relationships and Graph Structure: The JSON-LD in an RO-Crate is fully aligned with Semantic Web practices and can be mapped to RDF graphs. Formalizations such as in [160] add rigor similar to our assertions. However, the focus is on URI-based RDF triples, not on PID-triple graphs between FDOs. This provides less reliability of persistence and no guarantees for machine-interpretable PID record metadata in referenced entities.

Model semantics: RO-Crates profiles tend to be domain-focused and are in fact increasingly used in Electronic Lab Notebooks (ELNs)[4]. Although this approach improves interoperability within the involved communities, it does not realize the intended model semantics of a resource-agnostic FDO abstracting from use-case-specific details.

Conclusions: RO-Crate meets our model at the levels of attribute typing and graph-centric description but diverges on PID strategy, explicit operation association, and resource-agnostic abstraction. It is therefore better viewed as a domain-oriented packaging and metadata enrichment mechanism, rather than as FDO in the strict sense that we assume.

[3] https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE
[4] https://www.researchobject.org/ro-crate/eln

3.4.3.3. Webby FDOs with RO-Crate and Signposting

Webby FDOs result from combining RO-Crate for structured metadata with the FAIR Signposting profile, which provides the kernel information and defines other model components.

Type system: Signposting maps Link relations and attributes to elements of the conceptual FDO model as described by the FDO Forum. Links, and thus also attributes, are typed via IANA Media Types. Hence, attributes are limited to one `type/subtype` shape each, and lack the registry-backed persistency and composite character ensured by PITs.

OOP/profile principles: Instantiating the Signposting profile corresponds to instantiating a KIP. Compared to our core data model, the definition of FDO structure and kernel metadata is simpler and less rigorous. In addition, the persistence and machine-actionable options of the interpretation environment are weaker or even not present, because Webby FDOs are typically resolved and interpreted on landing pages, rather than in PID-service–hosted information records. Nevertheless, an abstraction layer to the digital resource is present, and operations can be declared as part of the kernel metadata.

Entity relationships and Graph Structure: Relying on RO-Crate, URI-based RDF triples are primary for Webby FDOs. Even if PIDs of other Signposting records can be referenced, enabling FDO-FDO relationships, the resulting structure does not fully match a PID-triple-based FDO graph.

Model semantics: Webby FDOs are technically close to the resource-agnostic intention of our model. However, the dependency on RO-Crate limits their scope in practice, as explained in Section 3.4.3.2.

Conclusions: Webby FDOs diverge from our model in terms of PID-level persistency, strict PIT-based typing, and full PID-graph alignment. Yet, they still align on typed attributes in general, explicit operation linkage, and an abstraction layer, even though the coupling to RO-Crate limits the resource-agnostic abstraction targeted by our core data model.

3.4.3.4. Nanopublications and the FDO Framework Ontology

Nanopublications are Semantic Web-native knowledge graph snippets that describe an atomic unit of publishable information. The FDO Framework defines a conceptual model (FDOF-CM) and an OWL implementation (FDOF-OWL), and previous work argues for their alignment with the requirements of the FDO Forum [150].

Type system FDOF-OWL uses OWL classes, properties, and individuals, with hierarchies comparable to PITs. However, OWL targets RDF or knowledge graph instances, while PITs type PID-service record content. OWL follows open-world and no-unique-name assumptions, whereas our PIT-based validation is effectively closed-world with registry-backed identifiers. Constraint languages such as SHACL/ShEx are needed to reach the same level of enforceable profile checks that PITs provide.

OOP/profile principles FDOF-CM does not model PID resolution as an explicit extra layer, so clients can query the graph directly (e.g., via SPARQL). Access to the digital resources is modeled via Globally Unique, Persistent, Resolvable Identifiers (GUPRIs), aligning with our model prerequisites. Types are defined by a constellation of classes that aggregate properties, analogous to our KIP, but the focus is narrower (encoding, content, represented entities) and mandatory attributes are not explicit. The assertion, provenance, and publication-information graphs resemble a structured kernel record, but there is no comprehensive operation-binding layer.

Entity relationships and Graph Structure The links to related resources follow the principles of LD and live within the RDF graph. FDO-FDO relations therefore appear as individual-to-individual RDF triples. PIDs can be embedded as IRIs to improve persistency, but PID-triple semantics and relationships at the global PID-service level are not native.

Model semantics In alignment with our model, both FDOF and nanopublications aim at domain- and data technology-agnostic kernel information and interlinking. However, they model multiple layered FDO components as OWL classes in a rather complex network of relationships, where, for example, metadata FDOs are treated different from data FDOs. This risks the genericity of FDO modeling and imposes a narrow interpretation of metadata beyond FDO-FDO relationships. Due to the grounding in an ontology language, their model assumes that clients can interpret OWL/RDF vocabularies and query knowledge graphs, in contrast to a PID-service environment that can be abstracted behind uniform interfaces (e.g., by the TPM). Although operations are conceptually foreseen, there is currently no detailed mechanism for declaring and executing operations on the typed attributes, so type-driven processing of FDO related metadata information is limited.

Conclusions: Nanopublications and FDOF align with our model on typed descriptions and graph-centric relationships. They diverge on the validation mechanism, genericity of (meta)data representation, PID-triple semantics, and persistence of the instantiation environment.

3.5. Discussion

By formalizing the problem description (the alignment of data spaces through FARIfication) and the solution approach (the introduction of a meta-level), we provided the first rigorous specification of the FDO concept. Until now, the lack of such precision has left room for interpretation, resulting in divergent implementation strategies and inconsistent harmonization attempts.

Our FDO core data model fills this gap: it defines how to realize the envisioned meta-level, using a set of rigorous assertions to relate principles from OOP, typing, ER modeling, and LD. To specify the semantic intention behind the construction of the meta-level, we developed a validation mechanism based on an activity diagram. This fosters appropriate model application, in particular with respect to the FDO abstraction paradigm: in fact, this diagram directs FDO creators to keep digital resources separated from their domain-

and use-case-specific metadata, and to link them through PID-triples. Such a mechanism facilitates the semantic intention of the core data model, but additional governance is required to fully enforce it. The conformance arguments in the assessment of FAIR MFCs explain how our model is capable of leveraging FAIRification characteristics that are implemented within individual data spaces, complying with the abstract meta-level in our premise.

As a proof of concept, we uniformly applied the model to two different domain-specific examples, showing its domain-agnosticism. However, some parts of the implementation examples are not yet fully specified or implemented—especially the consistent association and execution of operations via a uniform interface. This points to a gap in the typing mechanisms that should define the rules for linking operations to typed elements of FDOs. Since machine-actionability depends directly on these operation mechanisms, closing this gap is the next crucial step.

The comparative analysis with other existing FDO models provides multiple insights:

1. It confirms the heterogeneity of the interpretations and implementations of FDOs. Our approach is highly oriented towards the DO concept formulated by RDA and links directly to the original definition by [93]. Some communities went in a similar direction, whilst others interpreted the FDO concept in a pure Semantic Web context.
2. None of the existing implementations is fully aligned with our model: some of them have similar type systems and corresponding technologies but different meta-level intention, while others align with our model semantics but have different type systems and technological approaches.
3. OOP principles and ER traits are present in all models, but their expressions differ from our interpretation of FDOs that combine the characteristics of abstract interacting objects with those of connected and uniformly structured entities on the web.
4. Given their current specifications, none of the analyzed models from related works has all the adequate requirements to constitute the meta-level for data spaces that we defined and realized. However, with the given intersections we identified, some of them could potentially adopt remaining traits of our core model specification to realize higher abstraction that is required for the meta-level. The concrete requirements and details of this are subject to a dedicated study, which goes beyond the scope of this dissertation.

4. Typing Mechanisms for Associating Operations with FAIR Digital Objects

This chapter builds on the work published in the Data Science Journal [13]. We define three mechanisms to type FDOs for associating them with operations, and evaluate these mechanisms quantitatively and qualitatively (answering to sub-questions 3 and 4 in Section 1.2).

4.1. Introduction

The alignment of data spaces by FAIRification would allow one to process large heterogeneous datasets from different domains in a machine-actionable way. The concept of FAIR Digital Objects (FDOs) outlines how such an infrastructure could be realized, i.e., representing digital resources of any type in a way that enables automated processing by operations accompanying an FDO [161].

An operation is generally associated with an FDO by a mechanism related to its type and may be executed on the metadata or on the bit sequence of a digital resource [162]. Operations may range from basic Create, Read, Update and Delete (CRUD) to more advanced ones such as data format conversion, and could be implemented using various technologies. The *FDO Type* is an overall characterization of the entire FDO entity, and different possibilities exist for associating operations with FDOs that finally depend on the adopted notion of this FDO Type. In principle, having multiple approaches is desirable, as there may not be a one-size-fits-all solution. Nevertheless, to ensure an interoperable ecosystem for FDOs, it is important to assess the mutual compatibility of these approaches and to define them in terms of a common notion of a type system [151]. Hence, formalized type specifications and user intentions paired with FDO infrastructure conditions are key to supporting the adoption of FDOs by different communities. A basic type system for FDOs was already defined in Definition 3 of Section 3.3.3.1, but its exact specification and evaluation with respect to the FDO Type and associated operations has not yet been fully scoped.

This chapter proposes a well formed notion of the FDO Type, grounded in a formal type system, enabling the consistent association of operations. The structure is as follows:

- Definition of general typing mechanisms for FDOs using principles from object-oriented programming (OOP) to associate operations with FDOs in Section 4.3.1.

- Specification of the typing mechanisms as Entity-Relationship (ER) models and derivation of corresponding graph models for analysis in Section 4.3.2.
- Formal deduction of each typing mechanism on top of the FDO type system defined in Chapter 3, and assessment with respect to subtyping and polymorphism in Section 4.4.1.
- Evaluation of association mechanisms based on the graph models in Sections 4.4.2 and 4.4.3.
- Overall discussion in Section 4.5.

4.2. Related Work

The definition of the FDO Type originates from its predecessor, the DO Type, which is described as a specific attribute in the DO's record (s. Section 2.4). In general, the FDO Type is referred to as a reference (likely PID resolvable) to a standardized, registered, machine-readable type-specification of how to treat an FDO [162, 163]. In practice, this type-specification is often related to the Media Type of the represented digital resource and is thus a specification of the bit sequence represented by the FDO [164, 130]. In the FDOF model [149], the FDO Type separates the Media Type, i.e., the encoding format, from the content (a label for the digital resource) that aggregates characterizing properties.

Several works on FDO implementations describe the theoretical applicability of FDO operations that a type system could provide, e.g., for Machine Learning [165], Biodiversity Science and Geoscience [166], Materials Science [167], or access control [168]. Specialized setups were implemented to enable the execution of operations in their FDO ecosystem: The Typed PID Maker (TPM) service [155], as well as the FDO Manager [169] that is part of an FDO testbed [170] within the FDO One project[1], are reference implementations to realize basic FDO operations (CRUD). Another FDO Manager is described in [171], which combines FDO modeling approaches based on the FDOF and RO-Crate (s. Section 3.4.3). Operations are explicitly mentioned at a high level for the FDO records within this architecture but are not further detailed. In addition, specialized clients and platforms that act on the typed information in FDO records are described in [172, 173].

However, an FDO Type that is determined by a record attribute and treated primarily as a label for-, or a pointer to external specifications of the bit sequence and its encoding format is limited. Information about how the FDO can be treated in its entirety (also with respect to polymorphism) is then possibly not captured. In addition, interoperability between FDO systems is weakened, as without an explicit and general set of type rules, different systems may interpret FDO Type-related information inconsistently and may not be able to derive operations from it. Hence, there are currently no general typing mechanisms as part of a formal type system for associating FDOs with non-trivial operations beyond

[1] https://www.indiscale.com/de/projekte/fdo-one-2/

CRUD [151]. To the best of our knowledge, also no quantitative or qualitative aspects of typing mechanisms have been addressed, hindering differentiated evaluation of their practical use and reproducibility.

4.3. Methodology

In this section, we define typing mechanisms for FDOs in analogy to OOP typing by which the FDO Type can be deduced in contrast to assigning it as a static record attribute. These mechanisms are then used to model associations between FDOs and their operations.

4.3.1. Typing Mechanisms in Analogy to OOP

As introduced in Section 2.1, operations are the actions that a type system allows on values of a given type, and hence on the objects that fulfill the *operation signatures* (i.e., input/output types).

Similarly, we define a set of typing judgments and rules, integrated as part of an FDO type system to deduce the FDO Type and the allowed operations based on involved FDO components. We call corresponding sets of judgments and rules *typing mechanisms*. This replaces static labels for FDO Types with a rule-based semantic contract, typing an FDO by its structure and behavior. The three typing mechanisms that we define are analogous to the OOP ones described in Section 2.1.3, as depicted in Figure 4.1.

Record Typing (RT) is conceptually similar to *structural typing*, where type compatibility depends on the structure of the object, i.e., which methods and attributes are enforced at compile time. This focuses on what an object of a given type can do rather than what it is. Similarly, FDO typing is defined by its typed attributes and the operations they declare, which are assigned when the FDO is created. Operations are directly associated and listed in the information record with this scheme.

Profile Typing (PT) is conceptually similar to *nominal typing*, where a method is bound to a class name that determines type compatibility, and is thus applicable to its instances (the objects) with access to its attributes. In analogy, FDO typing is determined by the FDO's profile (which corresponds to a KIP Type) that is instantiated by the FDO information record and which is associated with a set of operations.

Attribute Typing (AT) conceptually relates to *duck typing*, a dynamic variant of structural typing, where the behavior of an object, and thus its type compatibility, is determined by the presence of specific attributes and methods at runtime. In FDOs, AT treats operations as external functions rather than as methods: an operation dynamically checks the presence of required attributes in the information record for FDO typing. This works for FDOs because their typed attributes refer to the specification of PITs, meaning that each element is unambiguously identified, has a defined value space, is possibly associated with terms from controlled vocabularies, and can be reused and recognized for all FDOs.

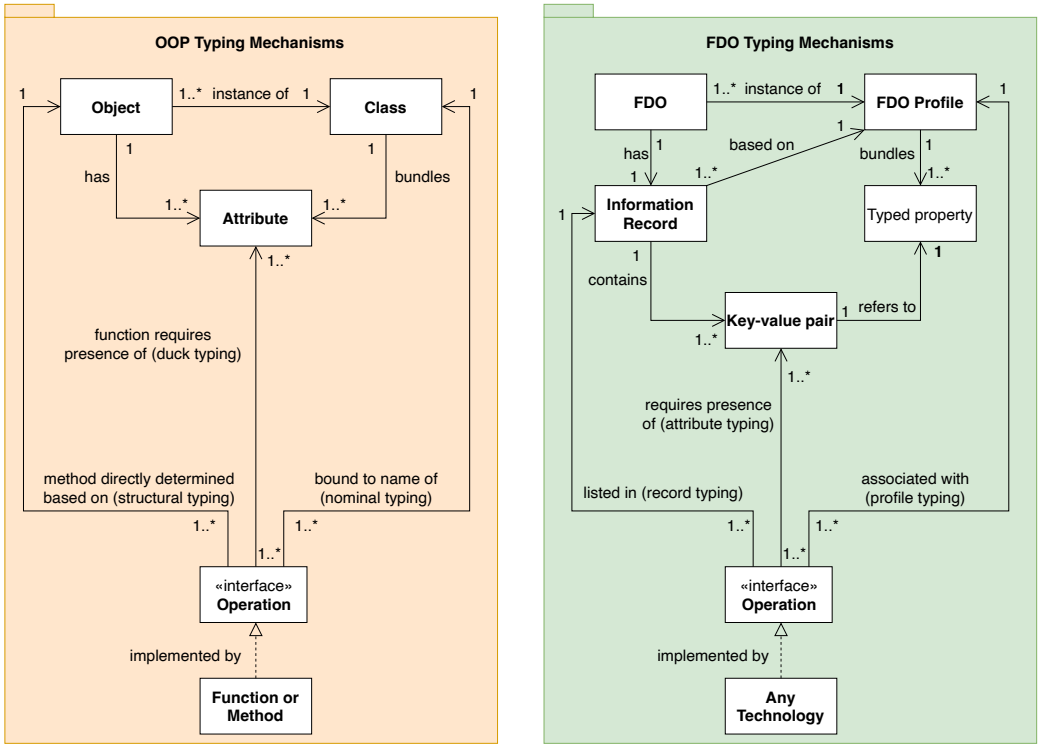


Figure 4.1.: Metamodel of the conceptual typing mechanism to associate FDOs with operations in analogy to OOP as UML class diagram.

4.3.1.1. Typechecking in FDO Infrastructures

Each typing mechanism is assessed with respect to their typechecking characteristics and the resulting implications for associated operations. FDO typing mechanisms can be **static** or **dynamic**, analogous to compile-time vs. run-time checks in OOP. Unlike a program in a single system, an FDO ecosystem is distributed, so typing must account for infrastructure components.

A type registry (e.g., a DTR) defines the types (PITs and profiles), and a proxy service (e.g., a PIT service) validates typed elements before registration of the information record, i.e., before instantiation time. For typechecking, this proxy works similar to a compiler's type checker, enforcing registry-defined rules on the information record (static typing). Any typechecking performed after FDO instantiation, during use, is dynamic typing.

4.3.1.2. FDO Subtyping and Polymorphism

For subtyping in FDOs, we assess two levels:

1. **Subtyping-Level 1 (SL1)**: captures hierarchical refinement relationships between typed FDO components (e.g., PITs) that are involved in a typing mechanism.

2. **Subtyping-Level 2 (SL2)**: captures hierarchical refinement relations between deduced FDO Types as they emerge under a typing mechanism.

With respect to **subtype polymorphism** (s. Section 2.1.2), we assess for each typing mechanism whether all FDOs that are instances of a given FDO Type (and its subtypes) are guaranteed to support the same baseline set of Type-associated operations.

4.3.2. Modeling Associations between FDOs and Operations

The three typing mechanisms described in Section 4.3.1 are the foundation for three association models, of which we define all the components involved as sets. ER Models are then constructed to specify the set relationships. These relationships express, for example, "FDO f contains attribute a in its information record" and "attribute a references operation o". In this way, each ER model semantically specifies the components and structure of an association model generically. The ER model components are then converted into directed graphs such that entities and their attributes (of the ER Attribute class) are represented as vertices, while relationships translate to edges. This enables assessment of computational complexity in terms of nodes (set elements) and edges (set element connections).

All components of the FDO ecosystem are uniquely identified by PIDs, except attributes that are instances of PITs within records. Besides, FDO information records, attributes also occur in records of the operation and profile specification schemes, which we simply refer to as operation and profile records, respectively. Whilst the different attribute categories are associated with their entities in the ER diagrams, we denote all attributes as one single set. Some of the components in the FDO space differ in numbers per association model or do not occur in each model, i.e., the set of profiles, specific attributes, and PITs. Hence, we assign each model its own index $i \in \{1, 2, 3\}$, such that $i = 1$ refers to RT, $i = 2$ to PT, and $i = 3$ to AT. These indices are used wherever differences in quantities for component sets or measures are expected.

Definition 9 (Component sets and graph models) Let $\mathcal{I}$ be the set of all identifiers including PIDs, F be the set of all FDOs, O the set of all operations, P the set of all FDO profiles, $\mathcal{T}$ the set of all PITs that are potentially involved in associations and $A_i \subseteq K \times V$ the set of attributes formed by the sets K of keys and V of values. The PID of any element in a given set $o \in O, p \in P, t \in \mathcal{T}$ is denoted as $\mathrm{id}(o), \mathrm{id}(p), \mathrm{id}(t) \in \mathcal{I}$, respectively, as also denoted in Definition 5. FDOs and operations are considered to be the same sets in all models. Strictly speaking, we assume that there are bijective mappings $\mathcal{M}_{ii'} : F_i \to F_{i'}$ and $\mathcal{M}'_{ii'} : O_i \to O_{i'}$ between FDOs from different models and operations from different models, for $i, i' \in \{1, 2, 3\},\ i \neq i'$. Each attribute $a = (k, v) \in A_i$ is represented by a tuple that consists of a key k and a value v. Two attributes $a_1 = (k_1, v_1)$, $a_2 = (k_2, v_2) \in A_i$ are considered to be the same element (i.e., $a_1 = a_2$) if and only if they have the same key-value pair (i.e., $k_1 = k_2$ and $v_1 = v_2$) and are part of the same record. If those two attributes are part of the same record and have only the same keys (i.e., $k_1 = k_2$), they can be transformed into a tuple $a = (k, \{v_1, v_2\})$. The set $\{v_1, v_2\}$ when being transformed into an array complies with the structure in Definition 5 for the validation of typed attributes.

We denote $G_i = (V_i, E_i)$ as the graph of the i^{th} association model:

- $i = 1$ (RT): each FDO is directly associated with an operation via an attribute within the information record. Hence,

$$\begin{aligned} V_1 &= F \cup A_1 \cup O, \\ E_1 &= \{(f, a) \mid \text{FDO } f \in F \text{ has attribute } a \in A_1\} \\ &\quad \cup \{(a, o) \mid \text{attribute } a \in A_1 \text{ references operation } o \in O\}. \end{aligned}$$

- $i = 2$ (PT): each FDO references a profile via an attribute in the information record. In turn, an attribute in the profile record references an FDO operation. Therefore,

$$\begin{aligned} V_2 &= F \cup A_2 \cup P \cup O, \\ E_2 &= \{(f, a) \mid \text{FDO } f \in F \text{ has attribute } a \in A_2\} \\ &\quad \cup \{(a, p) \mid \text{attribute } a \in A_2 \text{ references profile } p \in P\} \\ &\quad \cup \{(p, a') \mid \text{profile } p \in P \text{ has attribute } a' \in A_2\} \\ &\quad \cup \{(a', o) \mid \text{attribute } a' \in A_2 \text{ references operation } o \in O\}. \end{aligned}$$

- $i = 3$ (AT): each operation references one or more PITs by the attributes in its record. These PITs correspond to one or more attributes in an FDO information record, to which the operation is then implicitly related. Hence,

$$\begin{aligned} V_3 &= F \cup A_3 \cup \mathcal{T} \cup O, \\ E_3 &= \{(f, a) \mid \text{FDO } f \in F \text{ has attribute } a \in A_3\}\} \\ &\quad \cup \{(a, t) \mid \text{attribute } a \in A_3 \text{ references PIT } t \in \mathcal{T}\} \\ &\quad \cup \{(o, a') \mid \text{operation } o \in O \text{ has (requirement) attribute } a' \in A_3\} \\ &\quad \cup \{(a', t) \mid \text{(requirement) attribute } a' \in A_3 \text{ references PIT } t \in \mathcal{T}\}. \end{aligned}$$

■

Edges always point from the originating entity to the referenced entity. Note that edges from attributes to PITs in case $i = 3$ arise from the instantiation itself. Consequently, the edges $\{x, y\} \in E_i$ are naturally ordered and can be modeled as directed edges. For any directed edge from $x \in V_i$ to $y \in V_i$, there will never be another directed edge from y to x due to model definition.

Figure 4.2 depicts the diagrams of the ER model and the corresponding exemplary graphs according to Definition 9 of a potential FDO ecosystem. In this example, all graphs contain the same FDOs $f_1, \ldots, f_4$, the same operations $o_1, \ldots, o_5$, and represent the same set of associations: f_1 is associated with o_1, o_2 and o_3, while f_2 and f_3 are associated with o_3, and f_4 is associated with o_5.

For the RT model, each FDO has several attributes for the association of operations, which contain the same key (i.e., $k_1 = k_2 = k_3 = k_4 = k_5 = k_6$). The attributes directly reference

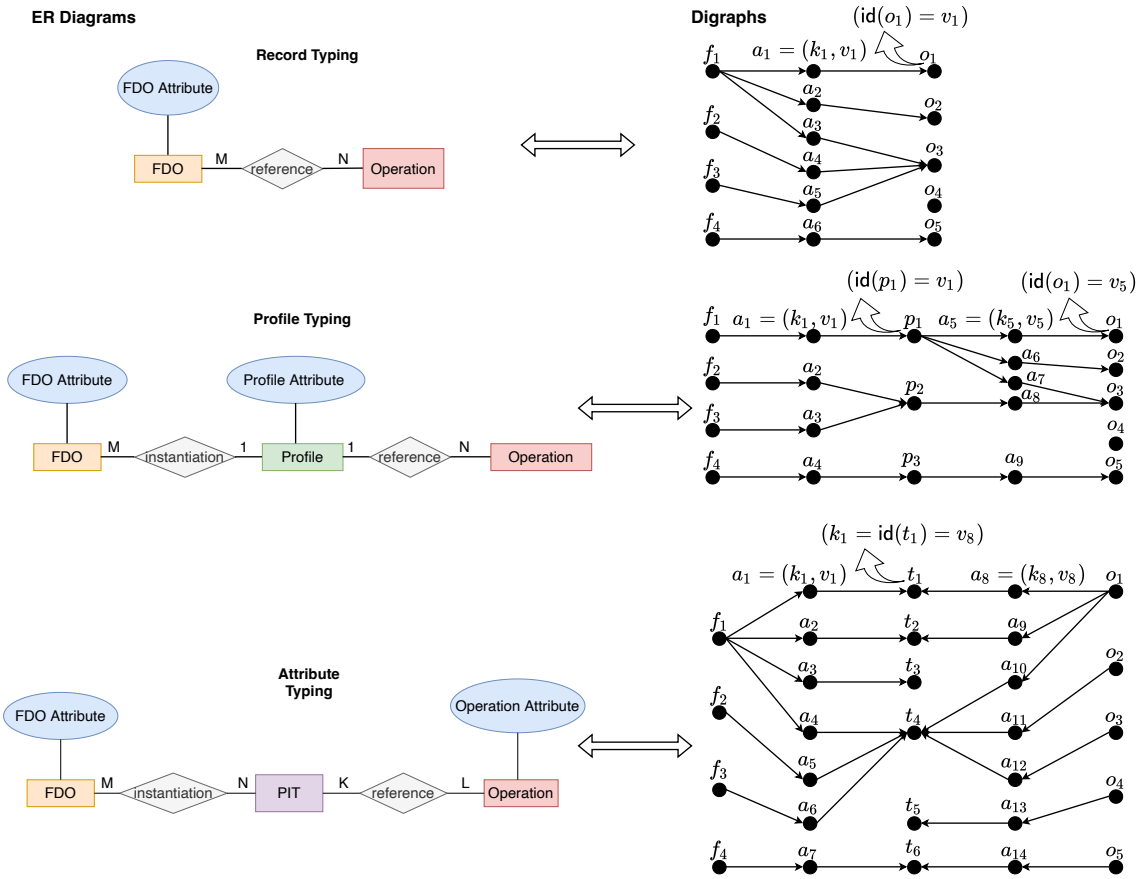


Figure 4.2.: ER and corresponding exemplary graph representations, modeling the three association approaches based on the typing mechanisms. Equivalence of elements that correspond to nodes is only indicated for the first example in a graph, respectively.

an operation via their value. In this example, the attributes a_3, a_4, and a_5 all have the same value (i.e., $v_3 = v_4 = v_5 = \mathrm{id}(o_3)$), because they refer to the same operation o_3. Each path connecting an FDO f_j with an operation o_w represents an FDO-operation-association.

In the PT model, each FDO has exactly one attribute that contains the profile reference. Those attributes have the same keys (i.e., $k_1 = k_2 = k_3 = k_4$). If two FDOs have the same profile, their attributes point to the same profile p_l in the graph (i.e., $v_2 = v_3 = \mathrm{id}(p_2)$). Each profile may contain one or more attributes to specify a list of operations. Similarly to RT, each path connecting an FDO f_j with an operation o_w represents an FDO-operation-association. In this example, o_4 is isolated because no profile attribute points to it.

For the AT model, each targeted FDO f_j may contain multiple attributes. Similarly, each operation o_w may contain multiple attribute requirements, with keys $k_8 = k_9 = k_{10} = k_{11} = k_{12} = k_{13} = k_{14}$. The attributes in the operation record reference a PIT t_m that is instantiated by attributes in a targeted FDO information record (i.e., in this example, we have $k_1 = \mathrm{id}(t_1) = v_8$, and so on). For PITs for which no matching information record exists (e.g., t_5), the operation cannot be associated (e.g., o_4).

4.3.3. Quality Indicators and Measures

For the evaluation of the different association models based on the introduced typing mechanisms, we define a set of quality indicators that are inspired by the methods used to evaluate conceptual models as described in [174]. These quality indicators finally serve the purpose of putting the different association models in relation to each other and evaluating their advantages, disadvantages, and compatibility. For those indicators that can be quantified (i.e., simplicity, efficiency, flexibility), we define measures that are evaluated separately on each graph model under the assumption that the whole FDO ecosystem purely relies on a single association approach. In addition, we will also consider purely qualitative aspects, i.e., granularity, required client knowledge, and versatility.

Quantitative Quality Indicators

Simplicity refers to how complex it is for a client to handle an FDO ecosystem that applies a given association model with respect to its structure. This is measured by considering the number of components involved and the number of their attribute-based relationships.

Efficiency takes into account how complex it will be to find all operations that are associated with an FDO or to assess whether a certain FDO is associated with a given operation. The measures are related to the traversal steps of edges in the graph that make up an association and are quantified using the big O notation.

Flexibility considers how many active modifications are required when new components are added to an existing FDO ecosystem that applies a particular association model. The measures relate to the number of updates that must be performed when a new association between an FDO and an operation is made and is also quantified using big O notation.

Definition 10 (Measures for quantitative quality indicators) We define the following measures to assess the quality indicators:

1. ϑ is the total number of identifiable components in the FDO ecosystem that are part of each association mechanism in a given model (higher Simplicity with lower ϑ).
2. λ is the sum of all attributes instantiated in all FDO records, profile records, and operation records that are part of the associations in a given model (higher Simplicity with lower λ).
3. Φ is an upper bound on the time complexity of an algorithm that decides whether an FDO is associated with an operation (higher Efficiency with lower Φ).
4. Ψ is an upper bound on the time complexity of an algorithm that finds all FDOs that are associated with a single operation. (higher Efficiency with lower Ψ).
5. φ is an upper bound on the time complexity of an algorithm that finds all operations associated with a single FDO (higher Efficiency with lower φ).
6. Υ is an upper bound on the time complexity of an algorithm that performs all required updates in the FDO ecosystem to associate a new operation with a set of FDOs (higher Flexibility with lower Υ).

7. Ξ is an upper bound on the time complexity of an algorithm that performs all required updates to associate a new FDO with a set of operations (higher Flexibility with lower Ξ).

■

Qualitative Aspects

The qualitative aspects we consider are the *Granularity* of the association models in comparison to the *Extent of Client Knowledge* that is required in order to make the association between an operation and an FDO (client hereby corresponds to operation and/or FDO creator or maintainer). In addition, the *Versatility* of the models is discussed, which considers the possible processing options of an FDO through its operations in relation to the effort that must be conducted for their association.

4.4. Evaluation and Results

The typing mechanisms are formally deduced to assess their typechecking, subtyping and polymorphism schemes, and the corresponding association models are evaluated with respect to quality indicators based on a comparative analysis.

4.4.1. Assessment of Typing Mechanisms

We formally deduce each typing mechanism based on the earlier defined FDO type system in Definition 3 using conventions from [19], and assess how it enables typechecking with an FDO infrastructure (s. Section 4.3.1.1), and subtype polymorphism with respect to SL1 and SL2 (s. Section 4.3.1.2). For all typing mechanisms, we assume the well formed (w.f.) FDO typing environment that relates to the types in a DTR, i.e., the domain $dom(\Gamma)$. For each typing approach, the aim is to derive the type rules from typing judgments that finally allow us to infer the FDO Type T for any FDO and consequently its set of operations. All judgments are only defined and considered under a well formed context; i.e., $\Gamma \vdash \diamond$. The deductions follow a proof-tree-*like* hierarchical approach where we derive conclusions from premises by ongoing application of inference rules (s. Table 1 for notation). Note that we use the term profile as a generic description of a KIP, and *KIP Type* as the corresponding notion at the type system level, focusing on its syntactic characteristics. The complete proof trees for each typing mechanism are provided in Sections A.4.2 to A.4.4, respectively.

4.4.1.1. Record Typing

For RT, we define the following syntax for Types (written in capitals) and their assignable terms:

Definition 11 (Record Typing syntax and judgments)

T_{rec} ::= Record-dependent FDO Type $\quad \tau$ ::= PIT

$\xi \hat{=} Array(\tau)$ ::= PIT Array (or Array with Card.=1) Type $\quad a$ ::= KIP property term

m ::= Numeral term $\quad [v] \hat{=} array(m, v)$ ::= array val. term $\quad R$ ::= FDO record term

The foundational judgments are the following:

$$\Gamma \vdash [v] : \xi \text{ (w.f. array value of PIT Array Type)}$$
$$\Gamma \vdash \xi' <: \xi \qquad (\xi' \text{ is a Subtype of } \xi)$$

■

RT Deduction

Recall the *Type KIP* rule where index $j \in \{1, \ldots, n\}$:

$$\frac{\text{(Type KIP) [with } a_j \text{ distinct]} \quad \Gamma \vdash \xi_1 \ \ldots \ \Gamma \vdash \xi_n}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}} \tag{4.1}$$

The set $\{\xi_1, \ldots, \xi_n\}$ may include PITs for ordinary properties as well as PITs for specific operation-reference properties.

A record with attributes that correspond to the properties in the KIP Type is then typed by:

$$\frac{\text{(KIP Instance Record) [with } a_j \text{ distinct]} \quad \Gamma \vdash [v]_1 : \xi_1 \ \ldots \ \Gamma \vdash [v]_n : \xi_n}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\} : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}} \tag{4.2}$$

A set of operations that are associated with T_{rec} is denoted as $O_{rec} = \{o_1, \ldots, o_j\}$, where each $o \in O_{rec}$ has a set of *input type* parameters Δ_o that are part of its operation signature. Since T_{rec} types a record that is typed by the same KIP Type that also types an FDO information record R, T_{rec} also types R. With this and a meta-condition for the fulfillment of an operation signature element by an attribute through the satisfaction relation $a \models \delta \iff (a, \delta) \in \mathcal{R}_{\models} \subseteq R \times \Delta_o$, we derive the following type rule:

$$\frac{\Gamma \vdash R : \{a_1 : \xi_1, \ldots, a_n : \xi_n\} \qquad \Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\} : T_{rec}}{\Gamma \vdash R : T_{rec}} \tag{4.3}$$

(FDO Type by Record)[with a_j distinct and where $\forall o \in O_{rec}\ \forall \delta \in \Delta_o\ \exists a \in R$ s.t. $a \models \delta$]

Typechecking in RT

In RT, the applicable operations are part of the FDO Type determination process itself, as deduced by the *FDO Type by Record* rule 4.3. All applicable operations must be therefore explicitly declared by instances of PITs and are fixed in the information record at the time of typechecking against a valid KIP Type prior to FDO creation/instantiation. Since each operation is individually and directly associated, this conforms to a static typing scheme

and is the most straightforward way of deriving the FDO Type. Since the information record must satisfy the operation input signature, the associated operations also depend on certain attributes besides those that declare them. Hence, the set of operations that are associated by declaration within the record cannot be arbitrary.

Subtyping and Polymorphism in RT

Operations in SL1 are determined by the subtype hierarchies of a PIT Array Type ξ used in their input signatures (with respect to value space and semantics). The same holds for PIT Array Types that occur inside different KIP Types, which are then also subtypes according to the following rule:

$$\frac{\Gamma \vdash \{a_j : \xi'_j\} \qquad \Gamma \vdash \{a_j : \xi_j\} \qquad \Gamma \vdash \xi' <: \xi}{\Gamma \vdash \{a_j : \xi'_j\} <: \{a_j : \xi_j\}} \text{(Subtype KIP) [with } a_j \text{ distinct]} \tag{4.4}$$

However, since operations are themselves declared by PIT arrays in the record, they must not change by introducing PIT Array Subtypes. Hence, KIP Types are only subtypes if they declare all operations of the same set O_{rec}.

Regarding SL2, let A and B denote two FDOs and the corresponding FDO records R_A and R_B, which are sets, and their Record-dependent FDO Types T_{rec_A} and T_{rec_B}, respectively. If R_B extends R_A, i.e., R_B is a superset of R_A, then we can derive a subtyping relation between T_{rec_A} and T_{rec_B} as follows:

$$\frac{\Gamma \vdash R_A : T_{rec_A} \qquad \Gamma \vdash R_B : T_{rec_B}}{\Gamma \vdash T_{rec_B} <: T_{rec_A}} \text{(Subtype FDO by Record Subset)[where } R_A \subseteq R_B] \tag{4.5}$$

Intuitively, T_{rec_B} is a subtype of T_{rec_A} because the set R_B contains at least the same typed attributes as the set R_A and adds additional details, increasing the specificity of the FDO Type. As a simple example, consider the following: a type *Shape* contains the attribute *color* and its subtype *Rectangle* contains additional attributes *width* and *height*.

If one record extends another (by adding further typed attributes), then every operation applicable to the smaller record is also applicable to the larger one. Therefore, the deduction of RT is monotonic. In aspects of polymorphism; the same operation that is associated with a type T_{rec_A} or any subtype $T_{rec_B} <: T_{rec_A}$ is applicable to all corresponding FDOs of these (sub)types according to their record structure. If these FDOs represent different digital resources, the operation will return different instance-specific results.

4.4.1.2. Profile Typing

For PT, we add the following to the syntax and judgments in Definition 11:

Definition 12 (Profile Typing syntax and judgments)

ξ_p ::= Prof.-reference PIT Array Type (Card.=1)

T_{prof} ::= Prof. Name-dependent FDO Type $\quad$ a_p ::= Prof.-reference property term

$[v]_p$ ::= Prof.-reference property value term $\quad$ res ::= resolve func. term

The foundational judgment is as follows:

$$\Gamma \vdash [v]_p : \xi_p \text{ (w.f. Prof.-reference value of Prof.-reference PIT Array Type)}$$

■

PT Deduction

A record that instantiates a KIP Type contains a mandatory property with index $j = p$ and is typed analogously to the *KIP Instance Record* 4.2 type rule. This is essentially just a type-theoretic formalization of the requirement for FDOs defined in Definition 4 that every FDO information record R has a profile-reference:

$$\text{(KIP Instance Record with Explicit Prof.-reference PIT Array) [with } a_j \text{ distinct]} \quad \frac{\Gamma \vdash [v]_1 : \xi_1 \ldots \Gamma \vdash [v]_p : \xi_p \ldots \Gamma \vdash [v]_n : \xi_n}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_p = [v]_p, \ldots, a_n = [v]_n\} : \{a_1 : \xi_1, \ldots, a_p : \xi_p, \ldots, a_n : \xi_n\}} \tag{4.6}$$

To select (denoted as a dot) this dedicated property, we have the following rule:

$$\text{(Select Prof.-reference Property)[with } a_j \text{ distinct]} \quad \frac{\Gamma \vdash R : \{a_1 : \xi_1, \ldots, a_p : \xi_p, \ldots, a_n : \xi_n\}}{\Gamma \vdash R.a_p : \xi_p} \tag{4.7}$$

From ξ_p and T_{prof}, we construct a function type by:

$$\text{(Function Type Prof.-reference PIT Array Type to Prof. Name-dependent FDO Type)} \quad \frac{\Gamma \vdash \xi_p \qquad \Gamma \vdash T_{prof}}{\Gamma \vdash \xi_p \rightarrow T_{prof}} \tag{4.8}$$

By a_p that is assigned ξ_p, the FDO Type is obtained by the resolving function type:

$$\text{(Resolve to Prof. Name-dependent FDO Type)} \quad \frac{\Gamma \vdash res : \xi_p \rightarrow T_{prof} \qquad \Gamma \vdash a_p : \xi_p}{\Gamma \vdash res(a_p) : T_{prof}} \tag{4.9}$$

Similar to RT, a set of operations associated with T_{prof} is denoted as $O_{prof} = \{o_1, \ldots, o_n\}$, again with its own set of input type parameters Δ_o that are satisfied in the following using a meta-condition. T_{prof} is then assigned to an FDO information record R by resolving the Prof.-reference according to:

$$\text{(FDO Type by Dereferenced Prof.)[where } \forall o \in O_{prof}\ \forall \delta \in \Delta_o\ \exists a \in R \text{ s.t. } a \models \delta] \quad \frac{\Gamma \vdash R.a_p : \xi_p \qquad \Gamma \vdash res(a_p) : T_{prof}}{\Gamma \vdash res(R.a_p) : T_{prof}} \tag{4.10}$$

Typechecking in PT

According to the *FDO Type by Dereferenced Prof.* rule 4.10, an FDO Type is fixed by an FDO's profile-reference attribute, of which the scheme is validated against the FDO information record prior to instantiation (corresponding to the *KIP Instance Record with Explicit KIP Name-reference PIT Array* rule 4.6). The consistency with the operation signatures of the operations associated with the profile is also checked at this point. As long as the profile behind the reference in the record does not change, the FDO Type remains stable. This results in a static typing scheme.

Because a profile is maintained independently, its operation set that is specified in its record can evolve. As a result, repeatedly retrieving the operations associated with an FDO may produce different results over time (i.e., the lookup is non-idempotent). For this reason, only operations whose signatures depend exclusively on mandatory properties can be associated with a profile, i.e., the Prof.-Name dependent Type T_{prof}, ensuring type safety at all times.

Subtyping and Polymorphism in PT

Regarding SL1; since profiles technically correspond to KIP Types, they can also be hierarchically structured (s. Equation (4.4)) to extend generic profiles, e.g., by using inheritance analogous to classes in OOP. Hence, we can derive the following rule for Prof. Name-dependent FDO Types:

$$\frac{\Gamma \vdash \xi'_p <: \xi_p}{\Gamma \vdash T'_{prof} <: T_{prof}} \quad \text{(Subtype Prof. Name-dependent FDO Type)} \tag{4.11}$$

To ensure substitutability of the Prof. Name-dependent FDO Type T_{prof} by a subtype T'_{prof}, this requires explicit (inheritance) rules — particularly for the cardinality and obligation of profile properties, so that subtypes may only strengthen constraints (i.e., make them stricter) and not relax them.

Regarding SL2, for every FDO information record R, we can then derive the following rule:

$$\frac{\Gamma \vdash res(R.a_p) : T'_{prof} \qquad \Gamma \vdash T'_{prof} <: T_{prof}}{\Gamma \vdash res(R.a_p) : T_{prof}} \quad \text{(Prof. Name-dependent FDO Subtype Subsumption)} \tag{4.12}$$

Hence, any FDO information records R_A with FDO Type T_{prof} and R_B with FDO Type T'_{prof} have (potentially substitutable) FDO Types, if they instantiate KIP Types of profiles with an inheritance chain. Furthermore, any FDO information records R_A and R_B have the same FDO Type T_{prof}, if they instantiate the same KIP Type.

Since the set O_{prof} of operations associated with T_{prof} is also present in T'_{prof}, any operation declared by a profile that corresponds to T_{prof} can be safely applied to all FDOs that instantiate its KIP Type or any KIP Types that correspond to a subtype T'_{prof}, resulting in polymorphic behavior.

4.4.1.3. Attribute Typing

For AT, the syntax and judgments of Definition 11 are extended as follows:

Definition 13 (Attribute Typing syntax and judgments)

τ_{req} ::= Requirement PIT T_{req} ::= Requirements-dependent FDO Type

r ::= requirement attribute term v ::= (requirement) value term

R_{req} ::= requirement PITs record term $mapReq$ ::= requ. to KIP Type function term

$\{a_1 : \xi_1, \ldots, a_z : \xi_z\}$ ::= KIP Type $\{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}$ ::= Requ. PITs Record Type

The foundational judgments are as follows:

$\Gamma \vdash v : \tau_{req}$ (w.f. value of Requirement PIT)
$\Gamma \vdash \{r_1 = v_1, \ldots, r_n = v_n\} : \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}$ (w.f. record of Requ. PITs Record Type)
$\Gamma \vdash \{a_1 : \xi_1, \ldots, a_z : \xi_z\} <: \{a_1 : \xi_1, \ldots, a_n : \xi_n\}$ KIP Type of size z is Subtype of
KIP Type of size n, given that $\{a_j \mid 1 \leq j \leq n\} \subseteq \{a_j \mid 1 \leq j \leq z\},\ z \geq n$

■

AT Deduction

To obtain all required PIT Array Types from the set of Requirement PITs, we define a function type that maps a Requirement PITs Record Type to its corresponding KIP Type:

(Function Type Requ. PITs Record Type to KIP Type)[with each a_j and r_j distinct]

$$\frac{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\} \qquad \Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\} \rightarrow \{a_1 : \xi_1, \ldots, a_n : \xi_n\}} \tag{4.13}$$

Assigning the function term $mapReq$ to this function type and passing it the requirement PITs record ($R_{req} \hat{=} \{r_1 = v_1, \ldots, r_n = v_n\}$) yields the required record of the KIP Type:

(KIP Type Assignment by Requ. PITs Record)[with each a_j and r_j distinct]

$$\frac{\begin{array}{c}\Gamma \vdash mapReq : \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\} \rightarrow \{a_1 : \xi_1, \ldots, a_n : \xi_n\} \\ \Gamma \vdash \{r_1 = v_1, \ldots, r_n = v_n\} : \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}\end{array}}{\Gamma \vdash mapReq(\{r_1 = v_1, \ldots, r_n = v_n\}) : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}} \tag{4.14}$$

Similar to RT and PT, a set of operations that are associated with T_{req} is denoted as $O_{req} = \{o_1, \ldots, o_n\}$, satisfied in the following using a meta-condition. Here, the elements in Δ_o correspond to those in R_{req}. Given that the FDO information record R is typed by a

KIP Type of size z, and T_{req} types a record of size n that meets the requirements specified in a record R_{req}, T_{req} types R according to the following rule:

$$\frac{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_z : \xi_z\} <: \{a_1 : \xi_1, \ldots, a_n : \xi_n\} \qquad \Gamma \vdash mapReq(R_{req}) : \{a_1 : \xi_1, \ldots, a_n : \xi_n\} \qquad \Gamma \vdash R : \{a_1 : \xi_1, \ldots, a_z : \xi_z\} \qquad \Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\} : T_{req}}{\Gamma \vdash R : T_{req}}$$

(FDO Type by KIP Subtype of mapped Requ. PITs Record)[with a_j distinct, and where $\forall o \in O_{req}\ \forall \delta \in \Delta_o\ \exists a \in R$ s.t. $a \models \delta$] (4.15)

Typechecking in AT Operations are independent entities with their own identity and are not explicitly declared by any element of the FDO they target. Instead, they check type compatibility at runtime when encountering an FDO depending on attribute availability and compliance according to the *FDO Type by KIP Subtype of mapped Requ. PITs Record* rule 4.15. This unambiguously follows a dynamic typing approach.

Subtyping and Polymorphism in AT

Regarding SL1, operations depend on the requirement PITs record and their subtypes. Given that $Array(\tau) \hat{=} \xi$, according to Definition 11, we have:

$$Array(\tau') <: Array(\tau),$$

and the following rule can be derived:

$$\frac{\Gamma \vdash a : Array(\tau') \qquad \Gamma \vdash Array(\tau') <: Array(\tau)}{\Gamma \vdash a : Array(\tau)}$$

(Array PIT Subtype Subsumption) (4.16)

The same holds for any $Array_{\text{Card.1}}(\tau')$ and $Array_{\text{Card.1}}(\tau)$. The judgment of the *Function Type Requ. PITs Record Type to KIP Type* type rule 4.13 (writing $Array(\tau)$ instead of ξ) when modified to consider all τ'_j that are a subtype of τ_j by subsumption looks as follows (note that any element a_j could also be assigned type $Array_{\text{Card.1}}(\tau_j)$):

$$\frac{\Gamma \vdash \{r_j : \tau_{req_j}\} \rightarrow \{a_j : Array(\tau)_j\} \qquad \Gamma \vdash Array(\tau') <: Array(\tau)}{\Gamma \vdash \{r_j : \tau_{req_j}\} \rightarrow \{a_j : Array(\tau')_j\}}$$

(Function Type Requ. PITs Record Type to KIP Type Subsumption)[with each a_j and r_j distinct] (4.17)

With respect to SL2, FDO Type subtyping is already implicitly given in the *Function Type Requ. PITs Record Type to KIP Type (Subsumption)* rules 4.13, and 4.17: two information records R_A and R_B of an FDO A and B, respectively, have the same type T_{req}, if they are subsets of the same record that is mapped from the requirement PITs record (given by the function $mapReq(R_{req})$).

Polymorphism then follows directly: the same operation can safely be applied to all FDOs that correspond to the derived type T_{req} that is effectively a supertype for one or more KIP Types these FDOs instantiate, independent of other FDO record contents or the explicit profile name. Different FDO instances may again produce different results for that operation.

4.4.2. Quantification of Quality Measures

In the following, we provide the quantification of the measures defined in Section 4.3.3 for each association model (s. Section 4.3.2). Here, we concentrate only on the comparison between the models rather than providing absolute numbers for implementation examples, as these are not relevant in the frame of a comparative analysis on the conceptual level. We assume a system where subtypes of types, i.e., KIP Types and PITs, can be used when their supertypes are expected. This follows Liskov's Substitution Principle (s. Section 2.1.2) and the deduction of subtype hierarchies we assessed in Section 4.4.1. Since the PIT and profile set elements $p \in P$ and $t \in \mathcal{T}$ of the graph models correspond to these types, their subtype hierarchies are implicitly captured in any connections of the set elements.

Definition 14 The following notation is introduced to evaluate the quantitative measures based on the sets and their graph-based connections in Definition 9.

1. For any non-empty set $F' \subseteq F$, $O_{F'}$ is the set of all operations that are associated with at least one FDO $f \in F'$. For the set of operations associated with a single element $f \in F'$, we write O_f. For PT ($i = 2$) and a non-empty set of profiles $P' \subseteq P$, we write the set of all operations that are associated with a single element $p \in P'$ as O_p, and the set of attributes in the record of p as $A_p \subseteq A_2$.
2. For any set $O' \subseteq O$, $F_{O'}$ is the set of all FDOs that are associated with at least one operation $o \in O'$. For the set of FDOs associated with a single element $o \in O'$, we write F_o.
3. For an FDO $f \in F'$ and an operation $o \in O'$, let $A_{i,f} \subseteq A_i$ and $A_o \subseteq A_3$ be the sets of all attributes in the records of f and o, respectively.
4. Finally, the following definition holds only if $i = 2$: for sets F' and O', we define $P_{F'}$ as the set of all profiles referenced by at least one FDO $f \in F'$, $P_{O'}$ as the set of profiles associated with at least one operation $o \in O'$, and $P_{F'O'} := P_{F'} \cap P_{O'}$ as the set of profiles that occur in at least one FDO–operation association. For the set of all profiles associated with a single element $o \in O'$, we write P_o.

■

Note that the total number of FDO-operation-associations is represented by $\sum_{f \in F} |O_f| = \sum_{o \in O} |F_o|$ irrespective of the association model.

The measures specified in Definition 10 have the following quantities under the conservative consideration that record elements must be traversed initially to find an entry:

Proposition 1. *The total number of components ϑ_i is, respectively:*

$$\begin{aligned}
\vartheta_1 &= |F_{O'}| + |O_{F'}| + 1, \\
\vartheta_2 &= |F_{O'}| + |O_{F'}| + |P_{F'O'}| + 2, \\
\vartheta_3 &= |F_{O'}| + |O_{F'}| + |\mathcal{T}|.
\end{aligned}$$

Proof. We count those components that are part of an association mechanism. For $i = 1$, this is the set of FDOs, the set of operations and a single PIT (assuming that all FDOs reference their operations via attributes that instantiate the same PIT). For $i = 2$, there are two PITs involved in the association mechanism, i.e., one to reference an FDO profile in all FDO information records and one to reference a list of operations in all profile records. For $i = 3$, there are no restrictions on the set of FDO- and operation record attributes that are involved in the dynamically checked association requirements. Hence, all PITs $\mathcal{T}$ are potentially participating in the association mechanism. □

Proposition 2. *For $i = 3$, let $b_1, \ldots, b_{|F_{O'}|} \in \mathbb{N}^+$ be the number of required attributes in the FDOs $f_1, \ldots, f_{|F_{O'}|}$, and let $d_1, \ldots, d_{|O_{F'}|} \in \mathbb{N}^+$ be the number of attribute requirements for each operation $o_1, \ldots, o_{|O_{F'}|}$ that are part of the association mechanism. The total number of instantiated attributes λ_i is then, respectively:*

$$\lambda_1 = \sum_{f \in F_{O'}} |O_f|,$$
$$\lambda_2 = |F_{O'}| + \sum_{p \in P_{F'O'}} |A_p|,$$
$$\lambda_3 = \sum_{g=1}^{|F_{O'}|} b_g + \sum_{h=1}^{|O_{F'}|} d_h.$$

Proof. Counting the number of attributes that are part of the association mechanism means to count all edges between the nodes of FDOs and attributes that are part of at least one FDO-operation association. For $i = 1$, each association corresponds to one attribute such that the number of attributes equals the total number of associated operations. For $i = 2$, each FDO contains exactly one attribute to be connected to a profile (totaling $|F_{O'}|$ attributes), and each profile has a set of attributes that connect it to a set of operations (totaling $\sum_{p \in P_{F'O'}} |A_p|$ attributes). For $i = 3$, the equation follows by the definition of b_g and d_h. □

Proposition 3. *The upper bound on the time complexity Φ_i for the given $f \in F_{O'}$, corresponding $p \in P_{F'O'}$, and $o \in O_{F'}$ is, respectively:*

$$\Phi_1 = O(|A_{1,f}|),$$
$$\Phi_2 = O(|A_{2,f}| + |O_p|),$$
$$\Phi_3 = O(|A_{3,f}| + |A_o|).$$

Proof. Let $f \in F_{O'}$ be any FDO and $o \in O_{F'}$ be any operation. For $i = 1$, the attributes that contain the reference to o in the entire FDO information record must be searched, taking time $O(|A_{1,f}|)$. For $i = 2$, the profile p must be found in the FDO information record in time $O(|A_{1,f}|)$. Since accessing the profile highly depends on the implementation but is not directly relevant for the comparative analysis, we assume access in constant time $O(1)$. Finally, the list of operations needs to be searched for references to o, taking time $O(|O_p|)$.

For $i = 3$, all attributes in the operation record must be found that specify the association requirements for the attributes of the FDO information record, which is done in $O(|A_o|)$, adding to $O(|A_{3,f}|)$. □

Proposition 4. *The upper bound on the time complexity Ψ_i for the given $o \in O_{F'}$ is, respectively:*

$$\Psi_1 = O\left(\sum_{f \in F_{O'}} |A_{1,f}|\right),$$
$$\Psi_2 = O\left(\sum_{f \in F_{O'}} |A_{2,f}| + \sum_{p \in P_{F'}} |O_p|\right),$$
$$\Psi_3 = O\left(\sum_{f \in F_{O'}} |A_{3,f}| + |A_o|\right).$$

Proof. For $i = 1$, each FDO information record in the FDO ecosystem must be searched for its operations in time $O(\sum_{f \in F_{O'}} |A_{1,f}|)$. For $i = 2$, a similar time is required to find all profiles $P_{F'}$. We assume that each profile can be accessed in time $O(1)$. The operation is then searched in the operation list for each profile in time $O(\sum_{p \in P_{F'}} |O_p|)$. For $i = 3$, first all requirement attributes of the operation record must be found in time $O(|A_o|)$. Then, all attributes in all FDO information records must be checked against these requirements, taking time $O(\sum_{f \in F_{O'}} |A_{3,f}|)$. □

Proposition 5. *The upper bound on the time complexity φ_i for the given $f \in F_{O'}$ and the corresponding $p \in P_{F'O'}$ is, respectively:*

$$\varphi_1 = O(|A_{1,f}|),$$
$$\varphi_2 = O(|A_{2,f}| + |O_p|),$$
$$\varphi_3 = O\left(|A_{3,f}| + \sum_{o \in O_{F'}} |A_o|\right).$$

Proof. For $i = 1$, it is required to read all attributes in the FDO information record. For $i = 2$, the profile p is accessed within $O(1)$, and reading the list of operations takes time $O(|O_p|)$. For $i = 3$, each operation record of all operations is also verified against the targeted FDO, which requires time $O(\sum_{o \in O_{F'}} |A_o|)$. □

Proposition 6. *The upper bound on the time complexity Υ_i for the given $o \in O$ and set $F' \subseteq F$ is, respectively:*

$$\Upsilon_1 = O(|F'|),$$
$$\Upsilon_2 = O(|P_o|),$$
$$\Upsilon_3 = 0.$$

Proof. For $i = 1$, relating a new operation to the set F' requires adding one attribute to each FDO information record, yielding $O(|F'|)$ updates in total. For $i = 2$, the new operation needs to be added to all profiles which are intended to be associated with it, in time $|P_o|$. For $i = 3$, no updates are necessary because the set F' is implicitly defined by the attributes in the operation record. □

Proposition 7. *The upper bound on the time complexity Ξ_i for the given $f \in F$ and set $O' \subseteq O$ is, respectively:*

$$\Xi_1 = O(|O'|),$$
$$\Xi_2 = 0,$$
$$\Xi_3 = 0.$$

Proof. To associate a new FDO with a set of operations O', $|O'|$ new attributes need to be added to the FDO information record for $i = 1$. In the case of $i = 2$, no updates need to be performed because the new FDO is required to have a profile anyway, and the profile implicitly defines the set O'. For $i = 3$, no updates need to be performed for the same reason as detailed in the *Proof* of *Proposition* 6. □

Note that the sets F' and O' in the *Proofs* of *Proposition* 6 and 7, respectively, are defined by the FDO creator/maintainer called the client in this context (for $i = 1$), or are imposed by the model (for $i = 2$ and $i = 3$). This is the case because the three association mechanisms follow different ideas: for $i = 1$, the client can decide on any association individually, so it will define the set F'. For $i = 2$, F' is partly determined by the model in advance given that the associations between profiles and FDOs were already declared. However, F' is also affected by any modification of the profile, i.e., when a new operation is added. For $i = 3$, the set F' is fully determined by the model in advance, depending on the attributes in the operation record that specify association requirements. Such considerations must be taken into account when evaluating the quality measures in the following section.

4.4.3. Comparison of Measures

We compare the measures to evaluate the quantitative strengths and weaknesses of the different association models with a rather large FDO ecosystem in mind. The overview of all measures is provided in Table 4.1. Afterwards, the quality indicators are evaluated.

- **Simplicity:** If few components are involved, the models are easier to understand for potential users. If in addition few attributes are involved, records (of FDOs, profiles, and operations) can be kept comparatively short. Regarding components, we have $\vartheta_1 < \vartheta_2$ and $\vartheta_1 \leq \vartheta_3$ (since $\mathcal{T} \geq 2$), while for ϑ_2 and ϑ_3, the following cases are possible:

$$\vartheta_2 \begin{cases} < \vartheta_3 & \text{if } |P_{F'O'}| + 2 < |\mathcal{T}| \\ = \vartheta_3 & \text{if } |P_{F'O'}| + 2 = |\mathcal{T}| \\ > \vartheta_3 & \text{if } |P_{F'O'}| + 2 > |\mathcal{T}| \end{cases}$$

Measures	Simplicity	Efficiency	Flexibility	Versatility	Gran. & Requ. Client Knowldg.
RT (i=1)	moderate to high	high	low	low	high and high
PT (i=2)	moderate to high	low to moderate	moderate	moderate	low and low
AT(i=3)	low	low to moderate	high	moderate to high	high and moderate
Metric overview	$\vartheta_1 < \vartheta_2$, $\vartheta_1 < \vartheta_3$ and typically $\vartheta_2 < \vartheta_3$; $\lambda_1 < \lambda_3$ and $\lambda_1 > \lambda_2$ and $\lambda_2 < \lambda_3$ in most cases, depending on profile reuse	Typically $\Phi_1 \lesssim \Phi_3 \lesssim \Phi_2$; $\Psi_1 \lesssim \Psi_3 \lesssim \Psi_2$; $\varphi_1 \lesssim \varphi_2 \lesssim \varphi_3$	$\Upsilon_1 \gtrsim \Upsilon_2 > \Upsilon_3$; $\Xi_1 > \Xi_2 = \Xi_3$	None	None

Table 4.1.: Overview of measures between RT, PT, and AT approaches and corresponding metrics.

In practice $\vartheta_2 < \vartheta_3$ is the most likely scenario, since less profiles are expected than PITs that instantiate attributes for the specification of the association requirements between FDOs and operations.

Given that $\sum_{f \in F_{O'}} |O_f| > |F_{O'}|$, the relationship between λ_1 and λ_2 depends on the number of profiles used to instantiate FDOs and the profile's record size, compared to the record size of each FDO. The following cases are possible:

$$\lambda_1 \begin{cases} < \lambda_2 & \text{if } \sum_{f \in F_{O'}} (|O_f| - 1) < \sum_{p \in P_{F'O'}} |A_p| \\ = \lambda_2 & \text{if } \sum_{f \in F_{O'}} (|O_f| - 1) = \sum_{p \in P_{F'O'}} |A_p| \\ > \lambda_2 & \text{if } \sum_{f \in F_{O'}} (|O_f| - 1) > \sum_{p \in P_{F'O'}} |A_p| \end{cases}$$

In practice, the same profile is expected to be instantiated by multiple FDOs, and many operations will be declared by the same profile for multiple FDOs instead of being declared in their information records. Hence, $\lambda_1 > \lambda_2$ is most likely.

The size of λ_3 is *at least* twice the size of λ_1 since for each operation at least one attribute requirement is needed, such that $\sum_{g=1}^{|F_{O'}|} b_g + \sum_{h=1}^{|O_{F'}|} d_h \geq 2 \sum_{f \in F_{O'}} |O_f|$.

Hence, $\lambda_1 < \lambda_3$, i.e., the association requirements specifications involve more attributes of FDOs and operations than the total number of associations.

Assuming the previous conditions, it trivially follows that $\lambda_2 < \lambda_3$ for most cases.

Overall, this makes RT and PT significantly simpler association models than AT.

- **Efficiency:** For efficiency measures, we have:

$$\Phi_1 \begin{cases} \lesssim \Phi_2 & \text{if } O_p > O_f \\ \gtrsim \Phi_2 & \text{otherwise} \\ \lesssim \Phi_3 & \text{if } A_o > O_f \\ \gtrsim \Phi_3 & \text{otherwise} \end{cases}$$

For Φ_2 and Φ_3, the following cases are possible:

$$\Phi_2 \begin{cases} \lesssim \Phi_3 & \text{if } A_o > O_p \\ \gtrsim \Phi_3 & \text{otherwise} \end{cases}$$

Typically, one profile declares more operations in its record than a single FDO (in $i = 1$) and has more attributes than a single operation record. Also, an FDO record has more attributes than an operation record such that we estimate $\Phi_1 \lesssim \Phi_3 \lesssim \Phi_2$.

For the same reason and the fact that all profiles must be searched, we also estimate $\Psi_1 \lesssim \Psi_3 \lesssim \Psi_2$. For φ_i, a similar estimate holds for φ_1 and φ_2, but φ_3 scales with the considerably larger number of attributes in all operations such that $\varphi_1 \lesssim \varphi_2 \lesssim \varphi_3$.

Overall, this shows that RT is generally the best approach in terms of efficiency. PT is generally less efficient than AT in terms of measures Φ_i and Ψ_i. However, φ_3 exposes the comparatively high cost of AT in specific scenarios: to identify the operations associated with a given FDO, the attributes of every operation record must be checked against the target FDO, which takes time $O\left(\sum_{o \in O_{F'}} |A_o|\right)$.

- **Flexibility:** Assuming that the number of FDOs associated with the new operation is larger than the number of profiles associated with this operation, it trivially follows that $\Upsilon_1 \gtrsim \Upsilon_2 > \Upsilon_3 = 0$. Given that $\Xi_1 > \Xi_2 = \Xi_3 = 0$, AT is the most flexible in terms of required updates, followed by PT. In comparison, RT is relatively inflexible.

- **Granularity and Required Client Knowledge:** For RT, any FDO can be associated with any operation as long as it matches the operation signature. This is the most granular approach since operations can be directly tailored to their FDOs. However, for each new FDO, the client that creates the information record must then decide which operations to include. This requires domain knowledge of the FDO's content, the operation signature, and familiarity with the association mechanism. Hence, higher granularity may come at the cost of careful case-by-case inspection. In addition, typed attributes that declare operations are not yet included in the recommended FDO kernel metadata, which may currently limit the feasibility of RT.

  PT is the least granular approach, since each operation is associated with an entire class of FDOs determined by their KIP. To reach granularity comparable to that of

the other models, many different profiles must therefore be available. The advantage is that the client only needs to choose an appropriate profile which is referenced in the FDO information record by a dedicated PIT. The profile definition then specifies the attributes that are required in the FDO information record. As a result, the client does not need to explicitly consider which operations are associated.

In AT, associations are determined automatically: clients simply include all available information in the FDO information record, which may match the signature, i.e., the association requirements, of an operation. If the operation signature was earlier carefully and properly specified, the association requirements are likely met. However, if a client needs an operation that is not automatically associated because its requirements are not met, it must identify which additional attributes are missing and add them retrospectively to the FDO information record. Therefore, AT is equally granular compared to RT, but may require less client knowledge since operations are ideally already defined such that association requirements match.

- **Versatility:** In contrast to its high granularity, the overall versatility of RT is considered to be the lowest, as each FDO-operation association must be explicitly declared to increase the possible processing options for an FDO.

  PT has much greater versatility compared to RT because a profile is typically reused several times to create a set of FDOs. All of these FDOs then automatically have the possible processing options defined by the operations associated with that profile.

  Compared to PT, the versatility of AT is potentially higher because PITs that refer to an association requirement are typically reused across profiles and may occur in multiple targeted FDOs. These FDOs then automatically have the possible processing options defined by these operations. In this way, an operation can still be associated with any FDO whose profile contains the required set of attributes, and the association is not missed because it was not explicitly declared in the profile. In addition, an operation associated via PT assumes the presence of mandatory PITs in the FDO information records that adhere to the profile. Due to this limitation, it can be expected that fewer operations are associated by PT compared to AT in general, and thus also fewer processing options for an FDO.

4.5. Discussion

The definition and deduction of typing mechanisms, i.e., RT, PT and AT, provide well formedness of the FDO Type within a type system for reliably associating operations to FDOs. The type of the bit sequence, e.g., given as Media Type specification, thereby remains a record attribute that can contribute to the overall FDO Type determination. Together with their respective subtyping relations and polymorphism traits, they establish a refinement structure in which more specific FDO Types guarantee at least the behavior of more generic ones. This makes the characteristics of an FDO machine-actionable in a systematic way. The type system provides a common language that allows different

communities to derive associated operations. In this way, actions such as accessing, reading, or transforming FDOs can be reliably requested by a client (a user or machine), regardless of the specific FDO service. The technical compatibility of an operation's technology is thereby ensured by the operation signature. This enables its consistent interpretation and execution on FDOs across different systems and platforms.

On this basis, association models can be derived as we did with the ER and graph models. Such models may be implemented using various technologies, but are consistent and compatible with respect to a coherent FDO type system. KIPs are obviously central in this context, since they provide a minimal, standardized metadata structure for all FDOs. Furthermore, all association models for FDO operations are expected to work with either a profile, profile attributes (also operations that are specified in the record) or a combination of both, so profiles maintain a basic level of compatibility. Hence, regardless of whether a simple or complex association model is used, the kernel metadata ensures that at least a core set of operations can be applied universally. In principle, these operations can be anything from simple CRUD to complex traversal of FDO-FDO relationships and processing of their corresponding bit sequences.

Our analysis underlines that each typing mechanism has distinct advantages and trade-offs for FDO ecosystems regarding simplicity, efficiency, flexibility, versatility, and granularity in conjunction with the required client knowledge. While RT offers simplicity and high granularity, PT and AT provide enhanced flexibility, versatility, and few required client knowledge. In practice, the storage capabilities of different services must also be taken into account, e.g., the PID services that store FDO information records (most relevant for RT) and the DTRs that store profiles (most relevant for PT). A growing number of attributes that specify operations may scale differently in these systems. In the context of AT, these attributes are distributed over the operation records, so the corresponding storage system must scale for the number of individual records rather than the size of the records. So far, there is no evidence that the three typing mechanisms are mutually exclusive, such that an FDO may be associated with its operations through any combination of them. This is indeed an important aspect in the context of FDO ecosystem interoperability: RT addresses only the use cases that the FDO creator intends and is aware of, because they must be explicitly declared before instantiation time. PT restricts the operations to those that are associated with mandatory kernel information, possibly extended by individual communities. AT overcomes these limitations by enabling the association of operations that do not require to be explicitly declared and are not limited to kernel metadata. This also indicates that interoperability between FDO ecosystems is easier to achieve with AT, since FDOs and operations only depend on the harmonization of PITs and not extended profiles. Considering the higher costs in AT with respect to simplicity and efficiency, one could say that increasing FDO ecosystem interoperability comes with higher costs in requesting available operations. How these aspects can be balanced with respect to implementation strategies is the next challenge to consider. However, what matters the most is that, with typing-based operation associations, clients can uniformly discover and invoke behavior across different FDO services, and hence data spaces that contain the corresponding digital resources.

5. The FDO-Ops Framework for Machine-Actionable FAIR Digital Objects

This chapter builds on the contents of the work published in the Data Intelligence Journal [15], the work published in the RIO Journal [14], and the work published in the proceedings of the IEEE 2025 International Conference on Science (eScience) [16]. The FAIR Digital Objects core data model and the typing mechanisms for associating operations are implemented using base service infrastructure components and advanced tooling for the execution of operations. The main outcome is the FDO-Ops process model and framework as a practical way to make FDOs machine-actionable, which enhances different levels of interoperability (answering to sub-questions 5-8 in Section 1.2).

5.1. Introduction

As described in many works, interoperability is the main bottleneck for efficient information exchange, also in the context of the FAIR principles [1, 56, 125, 175, 176]. Here, the term interoperability is defined as “the ability of data or tools from non-cooperating resources to integrate or work together with minimal effort" [1]. In addition to the conditions for data resources to be findable and accessible, “data becomes interoperable when it is machine-actionable” [175], and reusability of data can be understood as a result of interoperable systems [177]. Machine-actionable decision making would indeed reduce the barriers to efficiently preparing digital resources for a given task. For example, in the context of intelligent building energy management, the work of [178] describes how variations in building operating characteristics and data quality prevent fully-automated data preprocessing for operational data, which could otherwise improve data analysis efficiency. Preprocessing typically depends on data characteristics like the format, evaluation of legal aspects such as the license, or more use-case specific information that is often available in external metadata files.

Knowledge Organization Systems (KOSs) (s. Section 2.2.2) based on standardized metadata schemas [179] provide a way to unify information management and access in repositories, and Semantic Web (s. Section 2.2.3) methods enable web service specifications, e.g., for Web APIs [180, 181]. They improve information assessment for humans, machine interpretability, and some degree of machine-actionable processing. However, alignment to

enable interoperability is difficult due to the lack of a universal solution, the diversity of standards, and the evolving nature of technologies. Therefore, achieving interoperability, e.g., at the technical, syntactic, and semantic levels, remains a challenging task, particularly across domains. [182]

FDOs constitute an approach to a harmonized meta-level representation of various digital resources and KOSs to abstract their technical, syntactic, and semantic complexity. Operations that are associated with an FDO by its type system can process its information and potentially manipulate its content properly depending on the given context. This must be supported by an FDO infrastructure, i.e., base services, communication protocols, and advanced tooling [183]. Of particular relevance in this context are Data Type Registries (DTRs), which contain the syntactic and semantic specifications of typed elements in FDOs. However, for full machine-actionability and resulting interoperability, a robust procedure is required not only to associate and request the operations but also to execute the individual technologies used for their implementation, e.g., Web APIs.

We introduce FDO-Ops, a process model that provides a holistic solution for machine-actionable FDOs. The outline is as follows:

- Development of an approach to categorize and represent FDO operations in compliance with the modeling of typing mechanisms in advanced DTR systems in Sections 5.3.1 to 5.3.4.
- Introduction of the FDO-Ops process model for machine-actionable FDOs in Section 5.3.5.
- Implementation of a framework based on the FDO-Ops model using FDO base service infrastructure, the Digital Object Interface Protocol (DOIP) and advanced tooling in Sections 5.4.1 to 5.4.4.
- Evaluation of advanced DTR system capabilities and the FDO-Ops framework based on example data in Sections 5.4.5, 5.5.1 and 5.5.2.
- Evaluation of how different interoperability layers are enhanced with FDO-Ops in Sections 5.5.3 and 5.5.4.
- Overall discussion in Section 5.6.

5.2. Related Work

5.2.1. Assessment of Interoperability

Various communities are using different interoperability frameworks and models to assess several layers and levels of interoperability (s. Section 2.3.3). For example, the EOSC Interoperability Framework (EOSC IF) [78] is used to assess the compliance of FDOs and Linked Data (LD) with each of its layers in [151]. Their findings indicate that FDOs conceptually align well with the EOSC IF in governance and technical use of PIDs, although

technological maturity still imposes gaps. Another example from [184] describes how the SLICES Research Infrastructure (SLICES-RI) can facilitate data reuse when aligned with the interoperability framework layers of the EOSC IF. The reference architecture for heterogeneous Systems of Systems based on the interoperability trilogy of [79] is used by [56] to motivate requirements around semantic interoperability and FAIR services. The Levels of Conceptual Interoperability Model (LCIM) [82] is used as a framework for conceptual modeling in the Modeling and Simulation discipline [185], the analysis of interoperability between mobile apps cross-platforms [186], the domains of systems, cybernetics, and informatics [187], in heterogeneous multi-agent systems [188], or to examine various interoperability issues in the context of the Industrial Internet of Things [189].

5.2.2. Knowledge Organization Systems and Executable Specifications

Metadata management models and systems such as GEMMS [190] or goldMEDAL [191] provide schema- and vocabulary-based representations to describe datasets and their context. They support uniform description, semantics, and structure across heterogeneous data platforms, including data lakes and data warehouses, often in conjunction with data catalogs and metadata management services such as Amundsen [192, 193]. As described in [194], metadata management models/systems are powerful but difficult to operationalize on an institutional scale because they tend to require administrative overhead and substantial expertise in modeling and data base skills, ontologies, and query-rewriting engines.

Research repositories typically operationalize metadata through established community standards and exchange formats, for example, the DataCite Metadata Schema for dataset description and DOI registration [195], Dublin Core [196], and dissemination profiles such as DCAT [197] and Schema.org/JSON-LD [198, 199], as exemplified by the Zenodo supported metadata formats in [200]. Whilst many repositories adopt accepted metadata standards, heterogeneous resource types and communities frequently rely on specialized schemas [201]. Interoperability can then be supported via (FAIR) schema cross-walks [202] or model unification techniques (e.g., [203]), which typically require domain expertise and are non-trivial due to contextual ambiguities [204].

Ontologies formally define entities, relations, and constraints, underpinning Knowledge Graphs and Linked Data Platforms, and enable semantic interoperability across diverse resources [205]. They are also used to describe the properties and capabilities of web services in a machine-interpretable way, for example, in OWL-S [206] or WSMO [207]. However, their adoption is often constrained by complexity, the required granularity of domain knowledge, and high maintenance costs [208].

Executable specifications built on KOSs support machine-actionable interactions with data and (web) services. RDF-centric approaches such as LDScript [209], ActiveRDF [210], RDF-based OAS [211], Hydra [212], the FAIR Data Point [213], or metadata application profiles [214] specify access and operations using Linked Data (LD). They improve integration and mainly semantic interoperability within RDF-based systems but make it

difficult to work with other data formats. JSON or YAML-based specifications such as the OpenAPI Specification (OAS) [215], HATEOAS [216], or the Common Workflow Language (CWL) [217] for command-line and workflow tools enhance interoperability through standardization, cross-platform portability, and multi-engine support. However, they depend on individual carrier technologies or ad-hoc scripts to enable machine-actionable processes on data resources.

In summary, existing KOSs and executable specifications support the implementation of different (especially semantic and syntactic) interoperability levels, but remain technology dependent, often domain-specific, and entail significant technical overhead, with a limited scope regarding machine-actionability on data and their metadata.

5.2.3. Machine-actionable FAIR Digital Objects

Many works discuss applications for machine-actionable FDOs. For example, [165, 218, 219, 220, 221] describe perspectives for researchers to gain efficiency and stronger FAIRness without adding administrative burden by using FDOs as part of the Canonical Workflow Framework for Research (CWFR) initiative. The works of [128, 194] introduce and assess an FDO-based data-lake architecture approach in which datasets, workflow manifests, and containerized execution environments are treated as typed, persistently identified objects for monitoring provenance information. Predefined functions (e.g., API actions) then operate uniformly on these objects, providing a consistent user interface and abstracting the full life cycle of a research experiment.

However, technical solutions are either realized by individual service-oriented FDO operations in specific setups or limited to basic Create, Read, Update, Delete (CRUD) functionalities of DOIP [96]. Currently, there is no comprehensive specification on how advanced DOIP operations can be implemented in a technology-agnostic way and which use cases they address. Furthermore, existing DTRs currently used for FDO typing comprise three instances — the ePIC test DTR [222], the ePIC production DTR [153] and the EOSC DTR [223] — which lack integration of systematic typing mechanisms to associate FDOs with their operations in a standardized way. Hence, the machine-actionable capabilities of FDOs are still limited to individual ecosystems, which impairs their interoperability in the larger context of data spaces.

5.3. Methodology

5.3.1. Specifying Operations

In this section, we provide a precise specification of different FDO operation categories and use case phases where these categories are typically applied.

5.3.1.1. Operation Categories

Depending on their role, we classify FDO operations into three categories: *Management-*, *Information Record-*, and *Data Resource Operations*, which are described in more detail in the following subsections.

Management Operations: Act on behalf of a particular service that manages a set of FDOs and are used to provide a basic infrastructure to maintain an FDO space, i.e., the proposed meta-level in Chapter 3. Therefore, they enable the core functionalities for FDOs, including basic CRUD operations and a mechanism to request operations from other categories.

Information Record Operations: These operations use and process the metadata contained in the information record of an FDO. They do not access the represented bit sequence and do not modify the content of the information record. However, they may process any set of key-value pairs in the record to gain insight into the characteristics of the represented bit sequence.

Data Resource Operations: These operations act on the bit sequence of the data resource that is represented by the FDO, also taking into account the metadata in the information record, with data access information as a minimum. A copy of the bit sequence itself may be manipulated by these operations, e.g., in the frame of a given application case.

5.3.1.2. Operation Phases

We propose applying these operation categories sequentially in three phases.

In **phase I**, there must be a way to initially search and filter for FDOs. In this phase, FDOs are aggregated by the text-based content of their information records rather than their typed elements (i.e., a Read-like operation). Conceptually, this phase involves only the category of Management Operations.

In **phase II**, the FDOs retrieved in phase I are further processed. Operations that are associated via the type system and requested by Management Operations are applied to the information record content to process kernel metadata. In this phase, both Management and Information Record Operations are involved.

Finally, in **phase III**, operations are also applied to the bit sequence of an FDO, comprising the retrieval of the original data resource, a modified version of it, or any result that is yielded from its processing. In this phase, all categories of Management, Information Record, and Data Resource Operations are involved.

Although Management Operations must be implemented directly in a service that aims to manage FDOs, Information Record and Data Resource Operations could, in principle, be implemented anywhere and made available to the client through the management service and FDO type system. The relations between these three phases and the three categories of operation are summarized in Table 5.1.

Table 5.1.: Overview of the three FDO operation phases and their relation to the three categories of operations.

Operation categories use in phases	Management Operations	Information Record Operations	Data Resource Operations
Phase I	Retrieve FDOs, possibly based on the textual information record content.	Not used.	
Phase II	Read, interpret, and act on information record metadata.		Not used.
Phase III	Read, interpret, and act on information record and the digital resource.		

5.3.2. Embedding Typing Mechanisms in DTR Model

We introduce a DTR model that realizes our proposed typing mechanisms (s. Section 4.3.1) to determine the FDO Type, i.e., Profile-, Record-, and Attribute Typing (PT, RT, AT), in a functional way. The foundation is the typing concept of Basic PITs, Composite PITs, and KIP types (here referred to as KIPs) that are employed in conventional DTR systems, i.e., ePIC and EOSC. The model logic is depicted in a UML class diagram (s. Figure 5.1), which visualizes an implementation-oriented representation of how the principles of subtyping by inheritance, polymorphism, and a way to define modular operation signatures are realized. When addressing the UML class instances, we write them in capital italics.

In this model, *PITs* (including Composite and Basic PITs) may inherit from at most one parent type, and each value must satisfy all constraints in the inheritance chain, supporting controlled specialization and reuse. Besides, they still capture the syntax of simple values using JSON primitives (boolean, integer, number, string), enriched with constraints such as regular expressions, value ranges, and enumerations of permitted or forbidden values.

KIPs support multiple inheritance, where OOP-known potential conflicts such as the *diamond problem* [224] are easier to mitigate because all *PITs*, *Typed Properties*, and *Operation Signatures* are defined independently of *KIPs* and are globally identified by PIDs. *Typed Properties* that are aggregated by a parent *KIP* may be overwritten if their `cardinalityRange`, which expresses the property's obligation and repeatability, is 0..1 or 0..*. *KIPs* can associate operations via a `"declares"` relationship with *Operation Signatures* according to PT.

Typed Properties are modeled independently from *KIPs* by which they are aggregated but keep their functionality of linking attributes in FDO information records to the type system. Each *Typed Property* conforms to exactly one *PIT*, may define default values, and can be specified independently from a *KIP* with individual cardinality ranges, covering optional and mandatory single values. *Typed Properties* are also association points for *Operation Signatures*, either via a `"declares"` relationship according to RT or via an `"executableOn"` relationship according to AT.

Operation Signatures define operations and are detailed by *Operation Steps*, which are associated with reusable *Technologies*, and may call other operations that participate in this step.

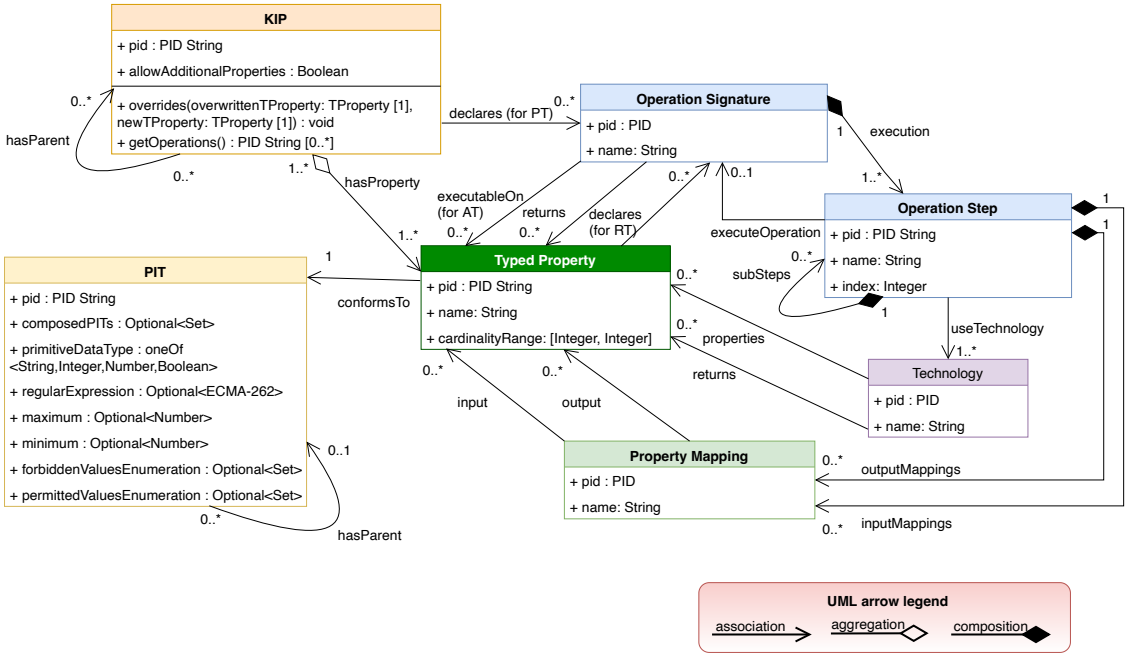


Figure 5.1.: Advanced DTR model with FDO typing mechanisms as UML class diagram.

Technologies define abstract input/output signatures for concrete technologies (such as a Web API or a Python runtime) in the implementation of an operation.

Property Mappings connect the operation-step-dependent *Technology* input/output signature via *Typed Properties* with the corresponding attributes in an FDO information record. This design allows *Operation Signatures* to be specified once and reused across different FDOs that share compatible types.

Each typing mechanism is realized within this model and ensures that *Operation Signatures* can be validated (for PT and RT before FDO instantiation time, and for AT during operation-FDO encounter at runtime).

5.3.3. Representing Operations Using FDOs

Our introduced DTR model implements the general (sub)typing rules for registered PITs and KIPs, whilst the concrete implementation of operations that correspond to instances of the *Operation Signature* class are independently constituted. To represent operations uniformly as abstract entities, we represent them as FDOs that we call *Operation FDOs*, which are consequently applicable to *Target FDOs*. Operation FDOs represent an operation as a resource in its own right, and thus may themselves act as Target FDOs for other operations. This aligns with the idea of FDOs to abstract all kinds of (potentially executable) digital resources, fits with the concept of extended operations in the DOA (s. Section 2.4.1), and supports the holistic view of "*everything is an FDO*" [225, 130]. That given, the terms "Target FDO" and "Operation FDO" are more context-dependent meta descriptions than

fixed categories. Since Management Operations (s. Section 5.3.1) are crucial for interactions with the FDO ecosystem, they are already required to create any Operation FDO. Hence, we apply the notion of Operation FDOs only to the categories of Information Record Operations and Data Resource Operations.

An Operation FDO has the same structure as a generic FDO, i.e., it has an identifiable and typed information record that instantiates a KIP. In this case, the information record must also include the technical details that correspond to an *Operation Signature*. Depending on the typing mechanism, an Operation FDO is either declared and referenced by a *KIP* (in PT) or by a *Typed Property* (in RT). In the case of AT, the *Operation Signature* requires specific attributes to be present in the FDO information record of a Target FDO (s. `executableOn` association in Section 5.3.2). Here, the *Operation Signature* can only ensure that *Typed Properties* expected by the corresponding Operation FDO are present in a Target FDO, but currently they cannot check for specific property values, nor can they express logical connectives beyond conjunctions for input requirements.

In general, compliance with the *Operation Signature* must be checked for an individual Target FDO by requesting the DTR. This is less of a problem in PT and RT, because the signature is supposed to be statically checked before FDO instantiation time, but for AT, it must be checked dynamically at runtime when an Operation FDO encounters a Target FDO. Providing the conditions of the *Operation Signature* directly within the corresponding Operation FDOs would circumvent this drawback and enhance the duck typing-like aspect of AT. Consequently, the advantages of AT come to the fore, which is why we proceed in the following with AT and a strategy to improve it.

5.3.3.1. Extended AT Associations

To reduce the need to repeat DTR requests when performing typechecking with AT, we embed the association conditions directly into the information record of an Operation FDO. Operation inference can then take place purely on the object, i.e., the PID service record level (we assume Handle System records). In addition, this allows us to extend the association conditions by logical statements that consider different key-value pair constellations. Compared to PT and RT, this greatly increases the granularity and semantic expressiveness of the conditions, allowing rules such as: "An FDO must have attribute x with value y *and* attribute z, *or* it must have attribute...". In the following, we provide a general definition of these extended association conditions.

Definition 15 (Attribute-typed association predicate) Denote O as the set of all FDO operations and F as the set of all FDOs. For each $f \in F$, let R_f denote its information record where each element $(k, V_\ell) \in K \times (\mathcal{P}(V) \setminus \{\varnothing\})$ is the key–value pair of a typed attribute, with K being the set of all attribute keys and V_ℓ a list, i.e., an array, that is an element of the power set $\mathcal{P}(V)$ of first-level record attribute values (sets as elements of power sets are convertible to array (list) format). The conformance between the attributes of the record and the properties typed according to the model in Definition 5 is thereby

ensured. Let $R_{all} \coloneqq \{R_f \mid f \in F\}$ define the set of all information records. We define the mapping function

$$Associated : O \times R_{all} \rightarrow \{\text{true}, \text{false}\},$$

as the predicate that decides whether an operation $o \in O$ is associated with an FDO $f \in F$ based on the contents of typed attributes in R_f.

A Management Operation $o \in O_M \subseteq O$ is a special category and is, by default, associated with any $f \in F$ unless at least one negation condition $\mathcal{N}_i(o)$ with $i \in \mathbb{N}^+$ applies (s. Section 5.3.5):

$$\forall o \in O_M,\ \forall f \in F : \neg\big[\mathcal{N}_1(o) \vee \cdots \vee \mathcal{N}_n(o)\big] \ \Rightarrow\ Associated(o, R_f) = \text{true}. \tag{5.1}$$

■

The condition predicates $\mathcal{N}_i(o)$ encode cases in which a Management Operation must *not* be applicable to a given FDO.

For Information Record and Data Resource Operations $O_{RR} \subseteq O$, the association is governed by conditions for typed attributes interpreted as logical expressions.

Definition 16 (Logical Attribute Typing conditions) Let

$$C_o \coloneqq \{\{s_{1,1}, \ldots, s_{1,x}\}, \ldots, \{s_{y,1}, \ldots, s_{y,z}\}\},$$

be a set of *conditions* for an operation $o \in O_{RR}$, where each condition is a finite set of key–value specifications:

$$s_{i,j} \in \{(k, v) \mid k \in K, v \in V \cup \{\varnothing\}\}, \tag{5.2}$$

with projections

$$\pi_1(s_{i,j}) = k, \qquad \pi_2(s_{i,j}) = v.$$

Here, $\pi_1(s_{i,j})$ denotes a typed attribute key and $\pi_2(s_{i,j})$ its required value (which is $\varnothing$ if only the presence of the attribute is required). For $o \in O_{RR}$, the set C_o is interpreted as disjunctions (for each $c \in C_o$) of conjunctions ($s_{i,j} \in c$). An FDO $f \in F$ with an information record R_f satisfies a condition $c \in C_o$, written $R_f \models c$, iff:

$$\forall s_{i,j} \in c : \ \Big(\exists\ (\pi_1(s_{i,j}), V_\ell) \in R_f \ \wedge\ \big(\pi_2(s_{i,j}) = \varnothing \ \vee\ \pi_2(s_{i,j}) \in V_\ell\big)\Big),$$

where the PIT of the key in the tuple $(\pi_1(s_{i,j}), V_\ell) \in R_f$ may be any subtype of the required PIT specified in the operation's condition, which could alternatively also occur in a separate disjunctive statement. The general requirement for o is then

$$R_f \models C_o \quad \Leftrightarrow \quad \exists c \in C_o : R_f \models c.$$

The association for Information record and Data resource Operations is given by

$$\forall o \in O_{RR},\ \forall f \in F \ :\ \big(Associated(o, R_f) = \text{true}\big) \ \Rightarrow\ \big(R_f \models C_o\big). \tag{5.3}$$

■

Note that (5.3) also applies to Management Operations $o \in O_M$ whenever at least one $\mathcal{N}_i(o)$ holds, i.e., when their applicability is constrained by specific attribute patterns rather than being unconditional. Associations can be asserted by Algorithm 2, for which a generic minimal working example is given in Appendix Listing A.1. An Operation FDO has an association conditions set C_o within its information record denoted as set R_o such that $(\textit{Typed attribute for conditions}, C_o) \in R_o$.

Algorithm 2 The *Associated* function with logical statements for asserting the association between an operation (possibly represented by an Operation FDO) and its Target FDO.

Input: o and R_f
Output: Boolean

if $(o \in O_M)$ ***and*** $(\neg[\mathcal{N}_1(o) \vee \ldots \vee \mathcal{N}_n(o)])$ **then**
    return *TRUE*
`// Check if any condition` c `of the requirements` C_o `is fulfilled (disjunctive term)`
for $c \in C_o$ **do**
    $result \leftarrow$ TRUE
    `// Check if each condition in` c `is fulfilled (conjunctive term)`
    `//` $\pi_1(s_{i,j})$ `and` $\pi_2(s_{i,j})$ `denote the key and value of` $s_{i,j}$ `as defined in Equation (5.2)`
    for $s_{i,j} \in c$ **do**
        if $\nexists V_\ell$ *s.t.* $\big((\pi_1(s_{i,j}), V_\ell) \in R_f \wedge (\pi_2(s_{i,j}) = \varnothing \vee \pi_2(s_{i,j}) \in V_\ell)\big)$ **then**
            $result \leftarrow$ FALSE
            Break
    `// Return as soon as one of the conditions in` C_o `is evaluated to true`
    if $result =$ *TRUE* **then**
        return *result*
return *FALSE*

5.3.3.2. Operation-Target FDO Graph

Although we can now assess the associations by the conditions array purely on the Handle record level, retrieval is still time-consuming since the entire record of both, Target and Operation FDO, must be checked.

To store their associations in a graph structure for efficient querying, we can represent them as triples in a graph model. As described in Section 3.3.3.3, PID-triples can be directly derived from those typed attributes in an FDO's information record that reference another FDO. In the following, we derive a similar constellation for Target-Operation FDO relationships.

Definition 17 (AT-based PID graph formation.) Denote $\mathcal{I}$ as the set of all existing PIDs, and let $\tilde{F} \subset \mathcal{I}$ be the set of PIDs that identify FDOs, $\tilde{O} \subset \tilde{F}$ the set that identify Operation

FDOs, and $\tilde{K} \subset \mathcal{I}$ the set of all PIDs of typed attributes. Let $Q \subseteq \tilde{F} \times \tilde{K} \times \tilde{K} \times \tilde{O}$ be the set of all existing PID-quadruples. Denote G_H as an undirected PID hypergraph, given by:

$$G_H = (V_H, E_H)\,, \tag{5.4}$$

with

$$V_H = \tilde{F} \cup \tilde{K} \cup \tilde{O}, \qquad E_H = \{\{\tilde{f}, \tilde{k}_f, \tilde{k}_o, \tilde{o}\} \mid (\tilde{f}, \tilde{k}_f, \tilde{k}_o, \tilde{o}) \in Q\},$$

where $\tilde{f}$ is the PID of a Target FDO $f \in F$, $\tilde{k}_f$ is the PID of a typed attribute in the information record of f, $\tilde{o}$ the PID of an Operation FDO that represents an operation $o \in O$, and $\tilde{k}_o$ is a typed attribute in the information record of the Operation FDO that defines an association condition.

Based on the association predicate, we define the set of Target–Operation triples

$$\mathcal{T} = \left\{(\tilde{f}, \mathsf{hasOperation}, \tilde{o}), (\tilde{o}, \mathsf{isOperationFor}, \tilde{f}) \;\middle|\; \tilde{f} \in \tilde{F},\ \tilde{o} \in \tilde{O},\ \mathit{Associated}(o, R_f) = \text{true}\right\},$$

where

$$\left(\tilde{f} \in \tilde{F},\ \tilde{o} \in \tilde{O},\ \mathit{Associated}(o, R_f) = \text{true}\right) \implies \exists\, \tilde{k}_f, \tilde{k}_o : (\tilde{f}, \tilde{k}_f, \tilde{k}_o, \tilde{o}) \in Q.$$

The collapsing duplicates from different $\tilde{k}_f, \tilde{k}_o$ over quadruples that have the same $\tilde{f}, \tilde{o}$ induces a labeled FDO digraph $G_D = (V_D, E_D)$ with

$$V_D = \tilde{F} \cup \tilde{O}, \qquad E_D = \mathcal{T}.$$

■

Figure 5.2 illustrates a minimal example for both graphs and the construction of G_D.

This enables for every associated pair (f, o) the explicit relationships emerging from AT

$$\text{Target FDO} \underset{\text{isOperationFor}}{\overset{\text{hasOperation}}{\rightleftarrows}} \text{Operation FDO.} \tag{5.5}$$

5.3.3.3. Executing Operations

Operation FDOs connect the abstract DTR-level operation signatures to a particular service or runtime of a corresponding technology. For these specifications, we introduce the concept of an *execution protocol*, which provides a harmonized representation of the technical execution details of different implementation technologies. Complementary to the association conditions set C_o, the protocol specifies which typed attributes in the Target FDO information record provide input for the operation and which operation steps are performed. Therefore, this protocol connects the workflow defined by instances of the Operation Step, Property Mapping, and Technology classes (s. Figure 5.1), which are specified for a given *Operation Signature* within the DTR, with the operation's implementation.

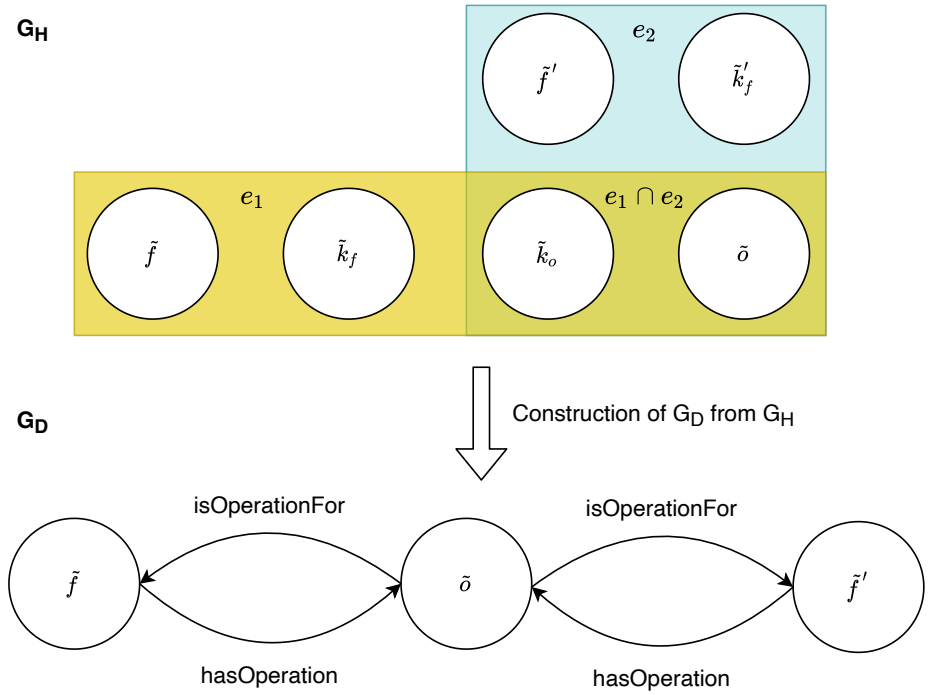


Figure 5.2.: Example of a PID hypergraph G_H with $V_H = \{\tilde{f}, \tilde{k}_f, \tilde{k}_o, \tilde{o}, \tilde{f}', \tilde{k}'_f\}$ and $E_H = \{e_1, e_2\} = \{\{\tilde{f}, \tilde{k}_f, \tilde{k}_o, \tilde{o}\}, \{\tilde{f}', \tilde{k}'_f, \tilde{k}_o, \tilde{o}\}\}$, from which a PID digraph G_D with $V_D = \{\tilde{f}, \tilde{f}', \tilde{o}\}$ and $E_D = \{(\tilde{f}, \text{hasOperation}, \tilde{o}), (\tilde{o}, \text{isOperationFor}, \tilde{f}), (\tilde{f}', \text{hasOperation}, \tilde{o}), (\tilde{o}, \text{isOperationFor}, \tilde{f}')\}$ is constructed.

Definition 18 (Execution protocol) Let $F_t \subseteq F$ be the set of all Target FDOs and $F_o \subseteq F$ be the set of all Operation FDOs that have this role in the context of a relationship to a Target FDO as defined in Equation (5.5). Let $e_o = (k_e, V_e) \in R_o$ be the execution protocol of an Operation FDO $f_o \in F_o$, for which we require:

- $k_e \in K$ is an *execution_protocol_type*, represented by the PID of a Composite PIT,
- $V_e \in \mathcal{P}(V)\backslash\{\varnothing\}$ is a single element list that contains the execution protocol parameter object, denoted as the set P_e.

Each protocol parameter $p_e \in P_e$ is a tuple

$$p_e = (\textit{parameter_type}, (\textit{parameter_key}, \textit{parameter_value}, \textit{merge_mul_entries})),$$

where:

- *parameter_type* is the PID of an object PIT describing this protocol parameter,
- *parameter_key* is a String used in the actual execution request,
- *parameter_value* specifies either
  1. the PIT of a typed attribute in the Target FDO information record that provides the value, or
  2. a static (literal) value, or

3. information to be supplied by a client at runtime, or
4. a nested execution protocol (sub-protocol),

where combinations are allowed.

- *merge_mul_entries* describes whether and how multiple values (for the same typed attribute) are merged. It is given by

$$merge_mul_entries = \begin{cases} (x, y) & \text{if } parameter_value \text{ references a PIT,} \\ \varnothing & \text{otherwise,} \end{cases}$$

where

$$x \in \{\text{true}, \text{false}\}, \quad y = \begin{cases} (prefix, delimiter, suffix) & \text{if } x = \text{true,} \\ \varnothing & \text{if } x = \text{false.} \end{cases}$$

If a = true, all values in the list of a typed attribute in the Target FDO are merged into a single String using the specified (*prefix*, *delimiter*, *suffix*) pattern, and used within a single request. Otherwise, each value is placed in a separate request.

In case the *parameter_value* within a protocol parameter p_e specifies an attribute's key of a key-value pair $(k, V_\ell) \in A_{f_t}$ (s. also Definition 15) of a particular Target FDO $f_t \in F_t$, the value list V_ℓ is input for the operation's execution . We capture this by the mapping functions

$$\mathsf{referencedKey} : P_e \to K_{f_t}, \qquad \mathsf{inputValue} : P_e \to V_{f_t},$$

where $K_{f_t} \subseteq K$ is the set of keys, and $V_{f_t} \subseteq \mathcal{P}(V) \setminus \{\varnothing\}$ is the set of value lists occurring as key-value pairs in $A_{f_t} \subseteq R_{f_t}$, yielding that:

$$\exists p_e \in P_e,\ \exists (k, V_\ell) \in A_{f_t}\ s.t.\ \mathsf{referencedKey}(p_e) = k\ \wedge \mathsf{inputValue}(p_e) = V_\ell. \tag{5.6}$$

■

These mapping functions are used to fill all parameters of the execution protocol with information from a Target FDO and to construct an *execution map M* according to Algorithm 3 that can be consumed by a software component. A minimal generic example for this mapping procedure is given in Appendix Listing A.2.

5.3.4. Cross-Walk Between DTR Model and Operation FDOs

To construct the information record content of an Operation FDO with respect to both, the execution protocol and the extended association conditions in compliance to the corresponding *Operation Signature*, we define the following cross-walk.

Algorithm 3 Mapping of execution protocol parameters and Target FDO information to an execution map.

Input: e_o, R_{f_t}
Output: M

```
Initialize M with one empty request.
for each p_e ∈ P_e do
    if client input is provided for p_e then
        mark parameter_value to be merged with client input
    else if p_e is a sub-execution protocol then
        recursively map sub-protocol and attach result to requests in M
    else if p_e fulfills condition (5.6) then
        if k_{f_t} (returned by referencedKey(p_e)) has multiple values returned by
          inputValue(p_e) then
            if x of merge_mul_entries is true then
                merge all values according to y of merge_mul_entries and add to requests
                  in M
            else
                add the first mapped value with corresponding parameter_key to all exist-
                  ing requests in M and create one new request for each remaining value
        else
            add the single mapped value with corresponding parameter_key to all requests
              in M
    else
        add the static parameter_value with corresponding parameter_key to all requests
          in M
return M
```

Definition 19 (Class instances to information record mappings.) Denote the DTR model class instance sets $\mathbb{O}, \mathbb{S}, \mathbb{A}, \mathbb{M}, \mathbb{T}$ for each operation $o \in O$, where

$\mathbb{O} \hat{=}$ *Operation Signature*, $\mathbb{S} \hat{=}$ *Operation Step*, $\mathbb{A} \hat{=}$ *Typed Property*,
$\mathbb{M} \hat{=}$ *Property Mapping* , $\mathbb{T} \hat{=}$ *Technology*.

UML arrow relationships are written as singletons $\{\cdot\}$.

Recall the information record $R_o \subseteq K \times \mathcal{P}(V) \setminus \{\varnothing\}$, consisting of key-value list pairs for each typed attribute it contains, the set $C_o \in R_o$ of association conditions $\{s_{i,j}\} \in C_o$, and the set P_e of protocol parameters in the execution protocol $(k_e, V_e) = e_o \in R_o$, where

$V_e \coloneqq \{P_e\}$. Let $\mathfrak{L}$ be the set of logical statements for the extended association conditions that occur in the elements of C_o. Define the cartesian products of cartesian product sets:

$$\mathbb{D}_1 = (\mathbb{T} \times \{\text{properties}\} \times \mathbb{A}) \times (\mathbb{T} \times \{\text{returns}\} \times \mathbb{A}),$$
$$\mathbb{D}_2 = \underbrace{(\{\varnothing\} \oplus (\mathbb{M} \times \{\text{input}\} \times \mathbb{A}))}_{\text{either or}} \times (\mathbb{M} \times \{\text{output}\} \times \mathbb{A}),$$
$$\mathbb{D}_3 = (\mathbb{O} \times \{\text{executeOperation}\} \times \mathbb{S}) \times (\{\varnothing\} \oplus (\mathbb{S} \times \{\text{subSteps}\} \times \mathbb{S})),$$
$$\mathbb{D}_4 = \mathbb{D}_3 \times (\mathbb{S} \times \{\text{useTechnology}\} \times \mathbb{D}_1) \times (\mathbb{S} \times \{\text{inputMapping}\} \times \mathbb{D}_2)$$
$$\times (\mathbb{S} \times \{\text{outputMapping}\} \times \mathbb{D}_2).$$

We then have the following partial mappings g_1, g_2, g_3 that constitute the cross-walk:

AT requirement PITs and logical statements to extended association conditions

$$g_1 : \mathbb{O} \times \{\text{executableOn}\} \times \mathbb{A} \times (\{\varnothing\} \oplus \mathfrak{L}) \rightharpoonup C_o, \quad (\text{o}, \text{executableOn}, \text{a}, \mathfrak{l}) \mapsto \{s_{i,j}\}.$$

Technology parameters-dependent operation steps and typed property mappings to execution protocol parameters

$$g_2 : \mathbb{D}_4 \rightharpoonup P_e,$$
$$\big(\text{d}_3, (\text{s}, \text{useTechnology}, \text{d}_1), (\text{s}, \text{inputMapping}, \text{d}_2), (\text{s}, \text{outputMapping}, \text{d}_2)\big)$$
$$\mapsto (\mathit{parameter_type}, (\mathit{parameter_key}, \mathit{parameter_value}, \mathit{merge_mul_entries})).$$

Technology dependent operation signature and returned types to execution protocol

$$g_{3\circ 2} : \big((\mathbb{O} \times \{\mathit{execution}\} \times \mathbb{D}_4 \times \{\mathit{returns}\} \times \mathbb{A})\big) \rightharpoonup R'_o \subseteq R_o,$$

$$\big((\text{o}, \text{execution}, \text{d}_4), \text{returns}, \text{a}\big) \mapsto \{(k_e, \underbrace{\{P_e(\text{d}_4)\}}_{V_e}, (\text{outputType}, V_\ell(\text{a}) \coloneqq \begin{matrix}\text{list of returned}\\ \text{properties}\end{matrix})\}.$$

■

In this way, an Operation FDO that is to be instantiated can be properly constructed in compliance with— and validated according to its *Operation Signature*.

5.3.5. Process Model

With the mechanism for associating different categories of operations to FDOs by AT and representing them as Operation FDOs, we yield the view on an FDO as a functional entity with a set of rules and elementary actions. This is summarized in the FDO-Ops Process Model that is shown in Figure 5.3 as a UML activity diagram.

Each activity in the diagram corresponds to a foundational operation on FDOs and is therefore realized by a Management Operation by definition, although not all possible Management Operations are depicted. Two activities, however, are not universally applicable:

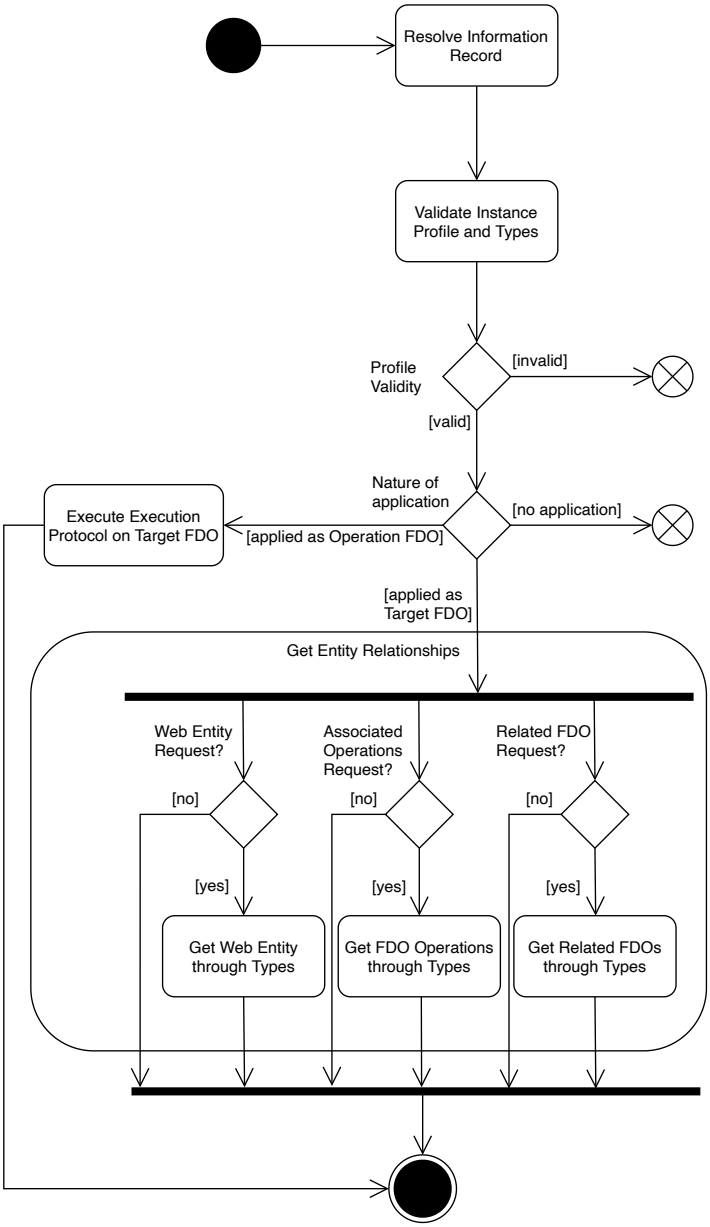


Figure 5.3.: The FDO-Ops Process Model for a FAIR Digital Object.

- `Get Entity Relationships`, which presupposes that the FDO is related to other FDOs via typed attributes, and
- `Execute Execution Protocol on Target FDO`, which presupposes the presence of an execution protocol (i.e., that an FDO can work as an Operation FDO).

To align this with the general association rule for Management Operations in Equation (5.1), we introduce two negation conditions:

- $\mathcal{N}_1(o)$: *the operation requires that the FDO is related to another FDO via a typed attribute*,

- $\mathcal{N}_2(o)$: *the operation requires the presence of an execution protocol.*

These conditions ensure that such activities can be modeled as Management operations, while preventing their application to FDOs that do not fulfill the required preconditions.

The Management Operations represented in the process model, together with an FDO type system, form the backbone of an FDO-centric processing regime with the following characteristics:

- the typed attributes in the information record are syntactically validated against the content, ensuring the integrity of the FDO,
- any FDO may serve as an Operation FDO or as a Target FDO depending on the context, which abstracts the concrete behavior of operations and enables a technology- and use-case-agnostic treatment of FDOs.
- core activities include:
    1. retrieving non-FDO web entities (including bit sequences) referenced via typed attributes,
    2. retrieving associated operations via the AT mechanism (`Get FDO Operations through Types`),
    3. retrieving related FDO entities via typed relationships (`Get Entity Relationships`), and
    4. executing the execution protocol of an Operation FDO on a Target FDO (`Execute Execution Protocol on Target FDO`).

While `Get Entity Relationships` could, in principle, also be modeled as an Operation FDO and executed via the execution protocol mechanism, it is treated as a fundamental service-level activity because FDO relationships are very common and must be supported natively by FDO services.

5.4. Model Implementation

The implementation part can be divided into three aspects:

1) The advanced DTR model that unifies the modeling of typing mechanisms and operation signatures is considered individually based on a prototype, since there is currently no deployed DTR service with its functionalities that could be used in the existing FDO base service infrastructure.

2) Since Operation FDOs also follow the FDO core data model defined in Section 3.3.3, they do not require the advanced DTR model to be *instantiated*. We therefore use conventional DTR systems for their instantiation. Nevertheless, we demonstrate how the logic of an operation signature is modeled in the advanced DTR prototype and mapped to an Operation FDO according to the defined cross-walk in Section 5.3.4.

3) To demonstrate machine-actionability enabled by our methodology, the FDO-Ops process model is implemented as a framework using a testbed FDO service architecture of deployed services and additional components in conjunction with DOIP. This results in an FDO-Ops framework, which is applied on real-world test data for several use cases.

5.4.1. Employment of Next-generation DTR Prototype

IDORIS ("Integrated Data Type and Operations Registry with Inheritance System") [226] was developed as a prototypical (micro)service implementation of the advanced DTR model (s. Section 5.3.2, with individual modifications but preservation of the overall typing logic). It uses a Neo4j[1] graph database such that instances of all model classes are represented as nodes in a labeled-property graph and encodes relationships such as inheritance and property usage as edges. The graph-based representation allows the registry to directly realize type- and operation-centric queries, e.g.,:

- discovering all operation signatures applicable to a property or, transitively, to a type (also considering inheritance),
- resolving and checking inheritance hierarchies,
- detecting cycles and structural inconsistencies in the type/operation signature graph.

These tasks are implemented by leveraging native graph traversal and graph algorithms (e.g., path finding, cycle detection, relationship queries) within the database rather than in application code, keeping the implementation close to the conceptual model.

We employ IDORIS to demonstrate that the proposed advanced DTR model can be realized as a practical registry service, where a graph-based backend reflects the conceptual structure of the model and a rule-based validation layer enforces its semantic constraints.

5.4.2. Digital Object Interface Protocol

To request all categories of FDO operations across different FDO services, a uniform communication protocol is required. DOIP fulfills these requirements (s. Section 2.4.1). Consequently, a DOIP-based FDO service is also suitable to infer the association between FDOs and their operations and to map the execution protocol according to our model specifications. To execute one of the Management or Information Record- and Data Resource Operations, the operation serviceID or Operation FDO PID is provided using the "operationID" parameter. The serviceID or Target FDO PID to which operations are applied is provided using the "targetID" parameter. The payload, which may contain either the FDO's information record or a set of query parameters for the operation, is provided in JSON format using the "attributes.full=true" parameter.

[1] https://neo4j.com/

DOIP can be directly implemented on the basis of TCP/IP, or as DOIP/HTTP[2], making it compatible with widely used HTTP Remote Procedure Call (RPC)-style API clients. In this way, all FDO operations can be requested uniformly using the basic HTTP `POST` and `GET` operations. Different example requests are provided in Appendix Section A.3.3.

5.4.3. FDO-Ops Framework Architecture

The service architecture for an FDO-Ops framework testbed environment is illustrated in Figure 5.4 and follows the concept of the envisioned high-level architecture in Figure 2.6. It consists of a base service infrastructure (PID service, PIT service, and DTR), an adapter called TPM Adapter, and an Executor, which are detailed in the following subsections. The entire implementation is available on GitHub (s. Appendix Section A.3.1).

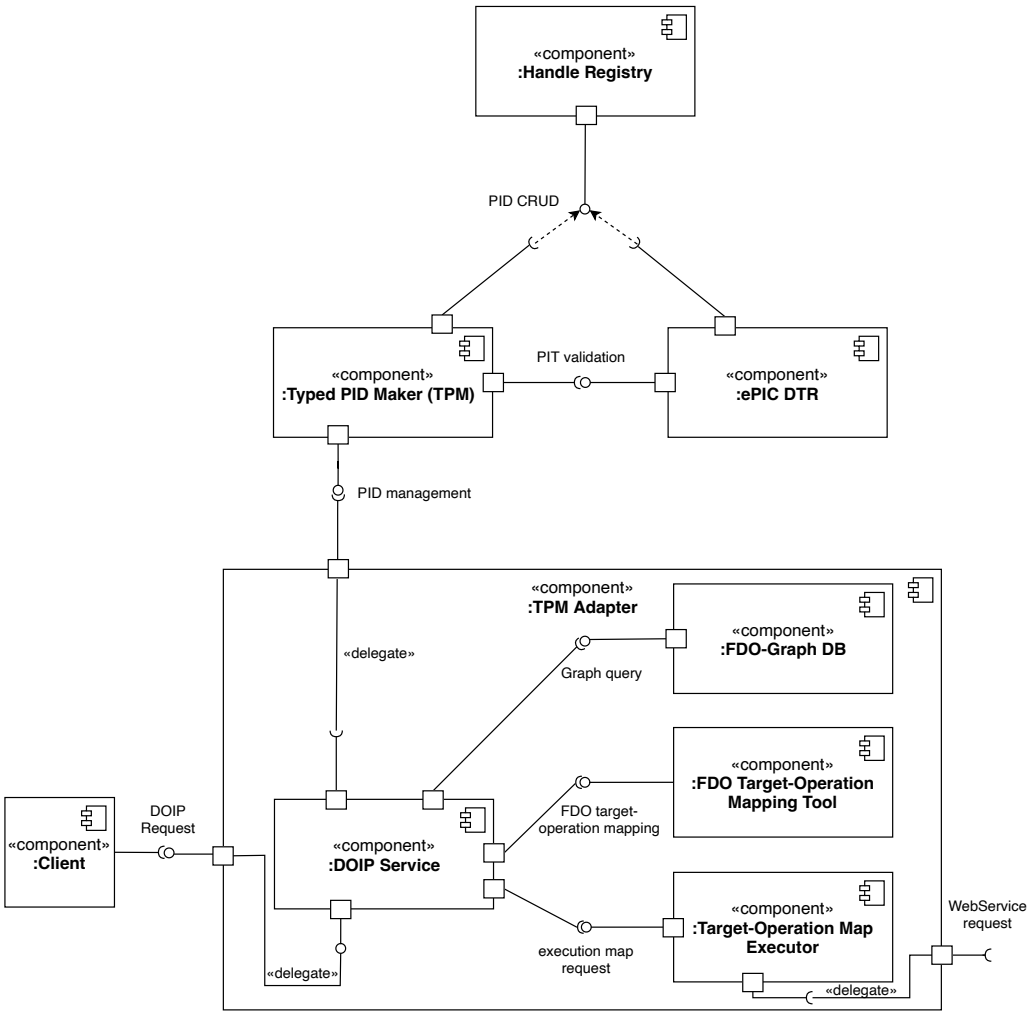


Figure 5.4.: The FDO-Ops service architecture depicted as UML component diagram.

[2] https://www.cordra.org/documentation/api/doip-api-for-http-clients.html

5.4.3.1. Base-service Infrastructure Components

The base service infrastructure follows the one introduced in Section 3.3.4, consisting of the Handle Registry, a Data Type Registry, and an instance of the TPM service [155]. Currently, the TPM supports only the ePIC and EOSC DTR[3], but could also be configured in perspective with a next-generation DTR such as IDORIS.

5.4.3.2. TPM Adapter

The TPM Adapter extends the interface of the TPM service by the following sub-components:

DOIP Service

A set of Management Operations for FDO retrieval and the activities depicted in Figure 5.3 can be requested using DOIP/HTTP and are handled by the *DOIP Service* sub-component. The component supports full-text search for all ingested FDO information records by Elasticsearch[4], which is useful as an entry point in Phase I as detailed later.

FDO-Graph DB

We integrate a Neo4j graph database that ingests the labeled FDO digraph (s. Definition 17) representing the triples as given in Equation (5.5) that is used by the `"Get FDO Operations through Types"` activity. This graph database also stores explicit FDO-FDO relationships constituted by PID-triples, which can be requested by the `Get Related FDOs through PID Information Types (PITs)` activity. Since Neo4j enables querying of opposite directions, storing only the relationship of two opposite edges is sufficient, e.g., the hasOperation and hasMetadata labels, respectively. To deal with duplicates and conflicting PIT-based edges in the Neo4j graph database, we integrate a set of conflict rules that are detailed in Appendix Section A.3.4.

FDO Target-Operation Mapping Tool

This sub-component implements the logic of Algorithm 3 to map the given execution protocol of an Operation FDO and the information record of an associated Target FDO to a JSON-based execution map that can be further processed by the Executor. In order for this setup to work, it is crucial that the TPM Adapter is configured with the PITs expected for the requirements specification in each Operation FDO as well as the PITs for the execution protocol types.

Executor

The Executor is a software sub-component that must be able to handle the output of the FDO Target-Operation Mapping Tool, i.e., the ready-to-use execution map. To deal with the specifications in a given execution map, the Executor must have an implementation for each execution protocol type, e.g., for Web API operations. These protocol types

[3] https://dtr-pit.pidconsortium.net/
[4] https://www.elastic.co/

are naturally limited due to their reuse for multiple operation implementations. On this basis, the Executor deals with the different technologies specified by an Operation FDO. Depending on the operation, the Executor technology module may also have to install additional software, which is detailed in the execution map. The command for the installation requirements is provided with a generic syntax "OS command" that the Executor replaces depending on the given Operating System. In addition, the Executor must be configured with the specific PITs of parameters in the execution protocol for the technologies it supports.

5.4.4. FDO-Ops Framework Sequence

The sequence diagram in Figure 5.5 outlines the sequence of steps that are performed with the FDO-Ops framework during the three phases listed in Table 5.1. For the sake

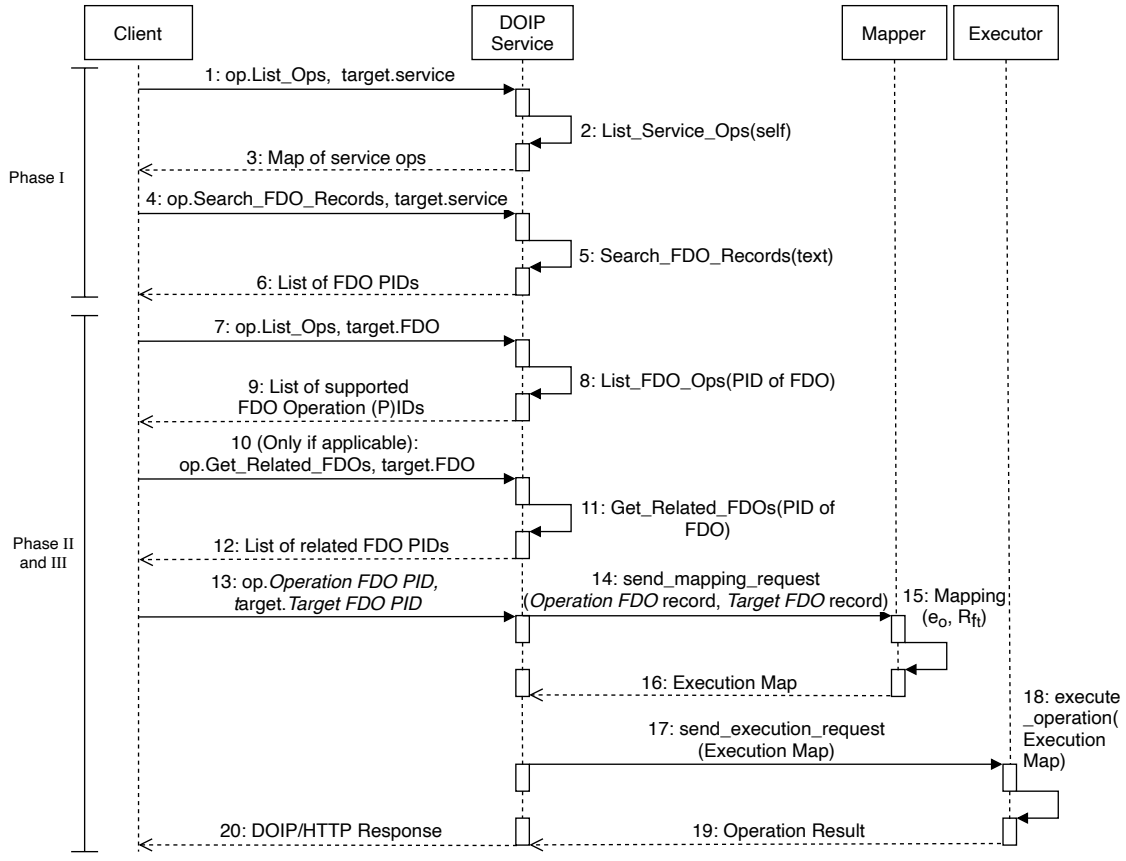


Figure 5.5.: UML sequence diagram of the FDO-Ops framework incorporating the DOIP request pattern.

of simplicity, the diagram does not show intermediate client actions between the steps such as inspection and selection of the individual information records of Operation and Target FDOs after step 9 and before use in step 10, if applicable, or step 13. The client may be an interactive computational environment such as a Jupyter Notebook, as in our demonstrative examples shown in the following section. The client is responsible for

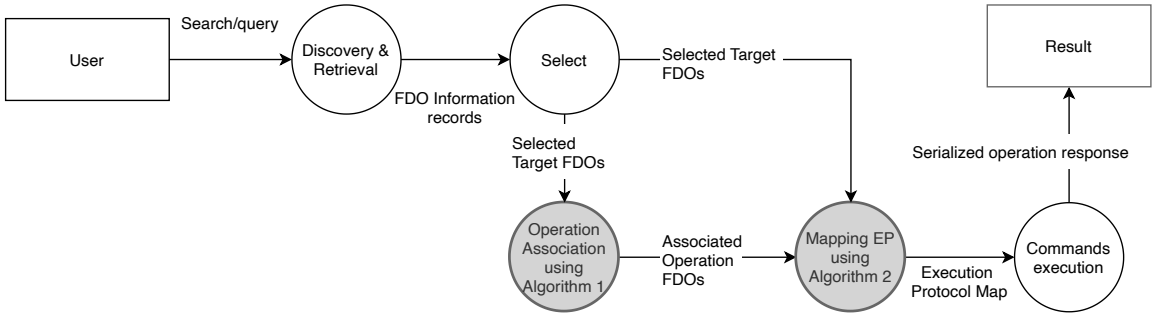


Figure 5.6.: Data Flow Diagram using the FDO-Ops framework according to the process model.

providing use case-specific input in all three phases and for evaluating the responses (such as the list of available Information Record and Data Resource Operations for a given FDO). The framework then handles all machine-actionable processes which are fully automated by means of the FDO-Ops process model. This automation abstracts the technical details of individual requests from the client, which are issued using the uniform DOIP/HTTP request pattern (s. Section 5.4.2), typically via the GET method. In case additional parameters are required, the client issues a POST request along with a payload. With this framework, the data and information flow according to the FDO-Ops process model is orchestrated as illustrated in Figure 5.6. This depicts how the two algorithms are subsequently executed after selection of a Target FDO in the discovery and retrieval phase. The commands of any applied Operation FDO that is associated and mapped with a Target FDO to the execution map can then be executed, and the serialized response is returned as a result.

5.4.5. Example Data

This section specifies the implementation of Operation FDOs, as well as the considered operations and data for testing the FDO-Ops framework.

5.4.5.1. Profile for Operation FDOs

To represent the operations involved in the use cases as Operation FDOs, we specify the *Operation Kernel Information Profile (KIP)*. This profile is derived from the *Helmholtz KIP* [133] and contains additional PITs for the association conditions array and the execution protocols we define, i.e., protocols for Web APIs and scripts that are executable on Unix-based systems with proper Executor software. All PITs that are properties of the Operation KIP, namely *Required Type* (for conditions array), *Required Client Input*, *Web API Execution Protocol*, *Script Execution Protocol*, *Output Type*, *Operation Name*, can be

further inspected via the profile that is currently registered in a running instance of the ePIC DTR [5].

5.4.5.2. Data and Operations Description

The same example datasets from Section 3.3.5 are employed, i.e., Thermal Bridges on Building Rooftops (TBBR) [11] and the OntoMatch dataset [12]. We replicate their information records using sandbox PIDs generated by the TPM to use them in our testbed setup.

For the TBBR dataset, which contains multiple resources, the corresponding FDO-FDO relationships derived from PITs hasMetadata and isMetadataFor, are ingested into the Neo4j graph database as illustrated in Figure 5.7. Note that the original Handle PIDs are illustrated in this figure along with their file names for readability, reflecting the original FDO Graph, whilst sandbox PIDs are used in the testbed.

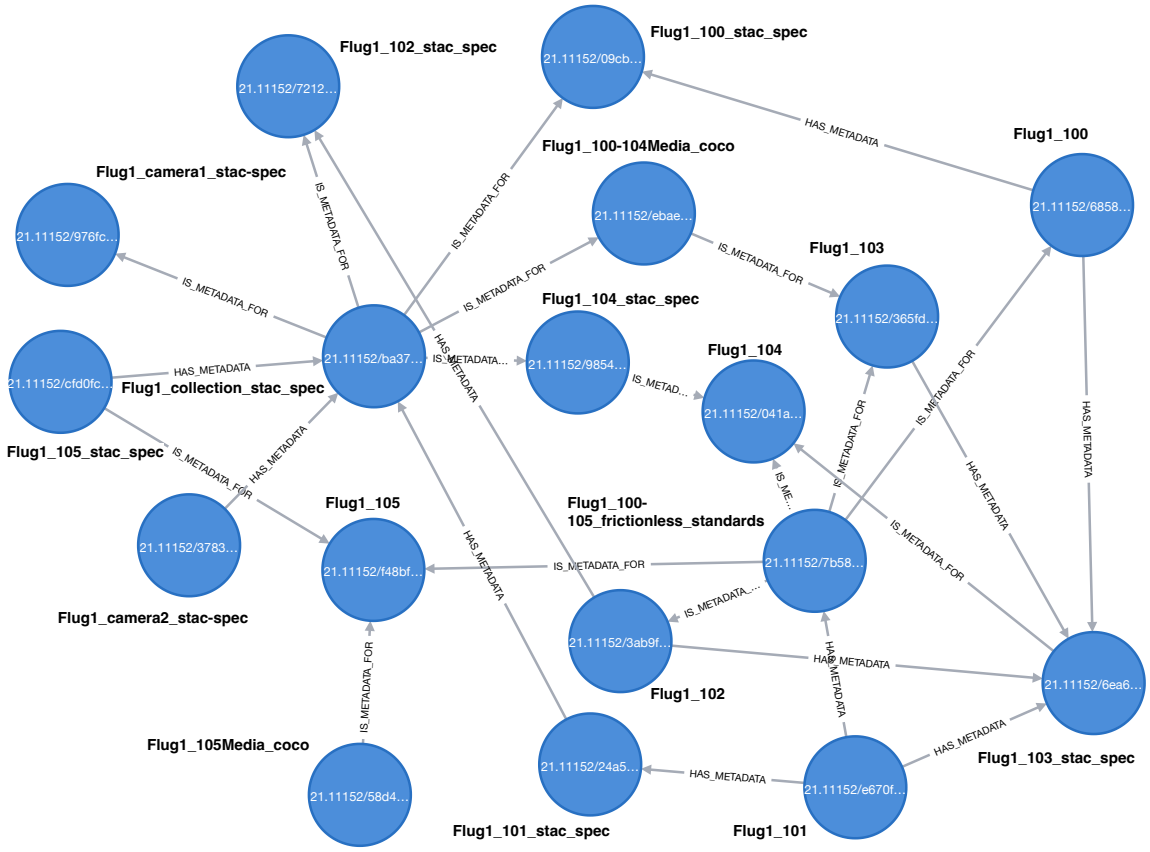


Figure 5.7.: The Neo4j graph of the TBBR dataset representing FDO-FDO relationships of FDO Graph.

In addition to the Management Operations that are already implemented as part of the software architecture, the following operations are represented as Operation FDOs using

[5] https://hdl.handle.net/21.T11148/ea4e93d06a10e15d9cdf?locatt=view:ui

the Operation KIP specified in Section 5.4.5.1 with the appropriate execution protocol: `Get related terms` for retrieving SKOS-based vocabulary terms information, `Convert Numpy to PNG` for converting a binary Numpy file into an image data format, and `Validate SKOS` for validating a file against the SKOS format. In the following use case description, we will refer to these operations by their assigned name reported in the information record. An overview of the inferred hasOperation relationships between these Operation FDOs and their Target FDOs, i.e., the earlier described datasets, is illustrated in Figure 5.8.

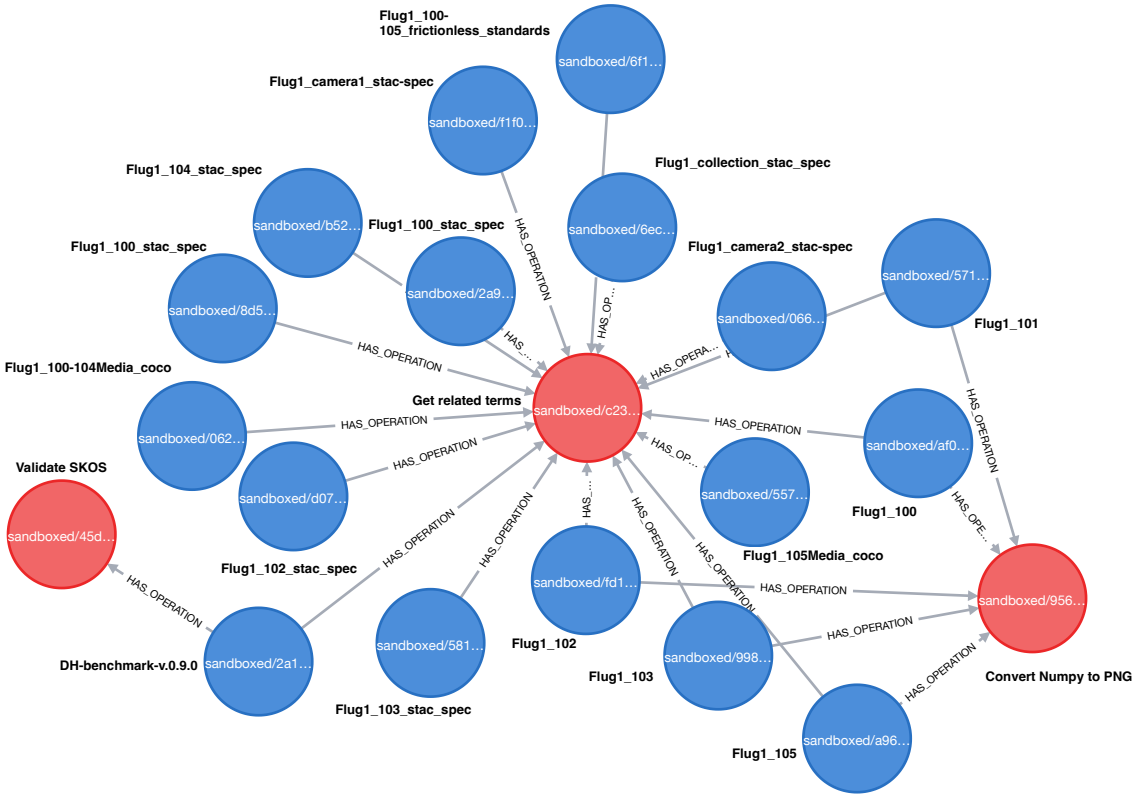


Figure 5.8.: The Neo4j graph of the Operation FDOs (red) and Target FDOs (blue) relationships.

The JSON-information records to create the sandbox PIDs and those operations that are implemented as Python scripts can be found on GitHub (s. Appendix Section A.3.1).

5.5. Evaluation and Results

5.5.1. IDORIS Graph Model

A remodeled extract of the IDORIS Neo4j graph is depicted in Figure 5.9. The original Neo4j graph is depicted in Appendix Figures A.5 and A.6. On this basis, we elaborate on the capabilities of IDORIS for implementing the advanced DTR model.

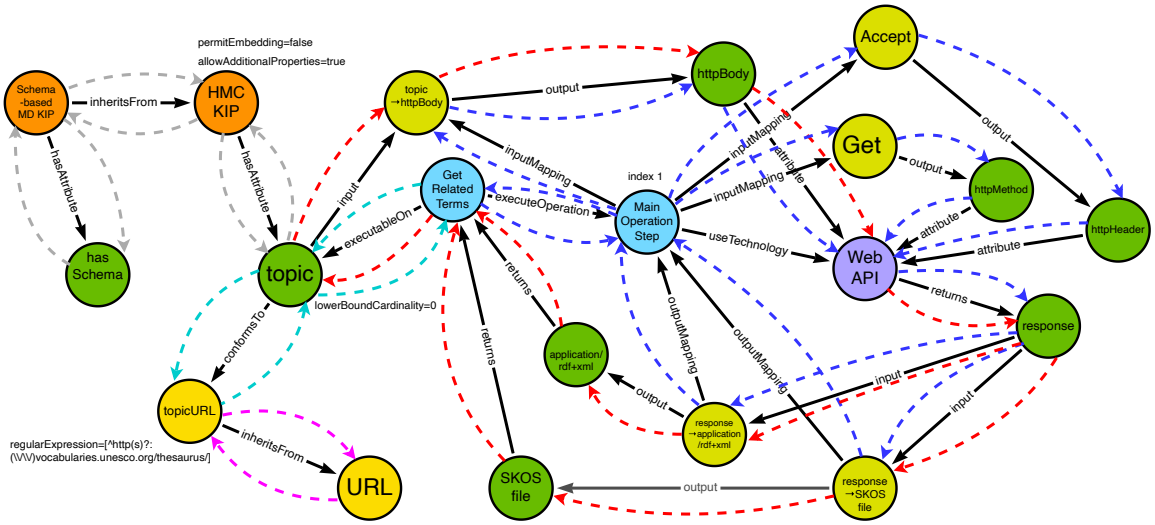


Figure 5.9.: Remodeled extract from IDORIS' Neo4j graph depicting inheritance hierarchies for KIPs and PITs, and the *Operation Signature* for Get Related Terms within a prototypical advanced DTR system. The nodes color code and edge annotations corresponds to the DTR model in Section 5.3.2. Annotated lines represent graph edges, and dashed lines visualize query induced graph traversals.

Resolving and checking inheritance hierarchies Consider the subtyping by inheritance scenarios for the "HMC KIP" and the "topic" property:

- The "Schema-based MD KIP" inherits from the "HMC KIP", for which the condition "`allowAdditionalProperties`" is `true`. The new KIP automatically receives all *Typed Properties* of its supertype and introduces a new *Typed Property* "hasSchema". These relationships and others are traversed and resolved in the graph upon request using Cypher queries which can use different edge directions, as indicated by the dashed gray lines.
- The "topicURL" *PIT* inherits from the "URL" *PIT* and applies a regular expression pattern:

  ```
  ^http(s)?:(\\/\\/)vocabularies.unesco.org/thesaurus/
  ```

  which restricts the allowed "URL" *PIT* instance to the UNESCO Thesaurus base-path syntax, ensuring that topics adhere to a controlled vocabulary. The corresponding traversal is indicated by the dashed magenta lines.

Operation signature discovery

Consider the "Get Related Terms" *Operation Signature*, which is by AT "`executableOn`" the *Typed Property* "topic" that conforms to the "topicURL" *PIT*, as indicated by the dashed cyan lines. To obtain the entire data flow of this *Operation Signature*, consider the cycle detection indicated by the dashed red line. The graph is traversed via the executable "topic" *Property*, which is mapped to the "Main Operation Step" parameter "httpBody" and then passed to the "Web API" technology interface. In parallel, the cycle with the dashed blue

lines retraces the input mappings of the "Main Operation Step" that also join the "Web API" node. The "Web API's" "response" is then mapped to the *OutputMappings* of the "Main Operation Step", which are also the return values of the entire *Operation Signature*, closing the blue- and red dashed cycle lines, respectively.

Cross-walk to Operation FDOs The *Operation Signature* depicted in the graph Figure 5.9 can be mapped to the corresponding Operation FDO that is instantiated with the Operation KIP (s. Section 5.4.5.1) using the cross-walk according to Definition 19. As a result, the association between this Operation FDO and its Target FDOs is backed by the DTR internal graph logic, ensuring governed type safety. Effectively, this is the step from the DTR-internal representation of the operation signature to the object level depicted by the FDO graph in Figure 5.8. The entire cross-walk procedure is provided in Appendix Section A.3.6 and the resulting JSON record of the `Get Related Terms` Operation FDO is provided in Appendix Listing A.3.

5.5.2. Applied Use Cases for FDO-Ops Framework

We consider different use cases to demonstrate how FDO-Ops can facilitate interoperable processes for the two example datasets using the software architecture (Section 5.4.3) and the framework sequence (Section 5.4.4). The selected use cases focus on the early stages of a research activity and target users mainly in the academic context who want to discover, evaluate, and aggregate data resources for further applications, e.g., Machine Learning tasks. To keep the use cases generic, consistent with what we addressed in Section 5.1, we deliberately do not further specify these potential research activities.

Each use case is related to one of the operation phases and framework steps as depicted in Figure 5.5, and begins with a brief description of the problem, followed by the identification of relevant components, i.e., the involved FDOs and operations. To demonstrate the domain-agnostic approach of FDO-Ops, we always apply them to both example datasets. In subsequent listings, we provide the technical details that are used to assess the association of the given operation, the input by the client, if any, the execution map in case an Operation FDO is involved, and the operation output type. For the PITs that specify the execution protocol parameters, we provide the human-readable name instead of the PID for the parameter PIT which is used in the actual records, except for those PITs that are used as a parameter value. On this basis, we demonstrate how these operations enable machine-actionable processes to facilitate the tasks described in the use cases. A Jupyter Notebook of the complete and reproducible results of these operations for each use case is available in the GitHub repository (s. Appendix Section A.3.1).

Phase I - Initial Discovery. A common problem in research is discovering and retrieving suitable digital resources, often by searching in different locations. Typically, users already have an idea of the characteristics and technical requirements that these resources must meet, such as a specific format or license. Full-text search of FDO information records using Elasticsearch serves as an entry point to retrieve the PIDs of potentially relevant FDOs. The client acting on behalf of the user explores this capability through the `List_Ops`

operation (Figure 5.5, steps 1-3) to list all available service operations. It then initiates a query for FDOs based on specific search terms, e.g., "*rdf+xml*" and "*npy-image*" for the Media Type or "*CC-BY*" for the license condition, by applying the Search_FDO_Records operation (Figure 5.5, steps 4-6) with the respective keywords (s. Listing 5.1). This yields the PIDs of both example datasets, i.e., those for the FDOs representing the TBBR images and the OntoMatch dataset (s. Section A.2.1), which can then be further processed.

Listing 5.1: Search_FDO_Records Operation Overview.

```
Implementation: Management Operation in TPM Adapter
Input provided by DOIP/HTTP Request Parameters:
   search terms "rdf+xml", "npy-image" and "CC-BY" in Elasticsearch syntax
Output Type: Handle-PID-list
```

As demonstrated for these examples, other FDOs from different locations that represent similar data can also be retrieved. At the end of this phase, the client has then yielded a selection of FDOs that are further analyzed.

Phase II - Metadata Assessment. Clients typically want to investigate the relevance of the data represented by the retrieved FDOs for their specific research task prior to retrieving the actual bit sequence. Therefore, the client requests additional operations based on the association assessment using the List_ops operation on a Target FDO (Figure 5.5, steps 7-9) as outlined in Listing 5.2.

Listing 5.2: List_ops Operation Overview.

```
Implementation technology: Management Operation in TPM Adapter
Input provided by DOIP/HTTP Request: PID of FDO
Output type: Handle-PID- or Service-ID-list
```

This yields a list of PIDs for Operation FDOs or IDs of service operations that are associated with the requested FDO. As a first action, the client may get all FDOs related to those retrieved in Phase I that support the Get_Related_FDOs Management Operation (Figure 5.5, steps 10-12) based on the specific association requirements as outlined in Listing 5.3. In the case of the TBBR dataset, this retrieves the related annotation and experimental metadata FDOs, while the OntoMatch dataset does not have a directly related FDO and is therefore not applicable to this operation.

Listing 5.3: Get_Related_FDOs Operation Overview.

```
Implementation technology: Management Operation in TPM Adapter
PITs for association requirement:
   hasMetadata (21.T11148/d0773859091aeb451528)
   OR isMetadataFor (21.T11148/4fe7cde52629b61e3b82)
Output type: Handle-PID-list
```

The client could then decide to repeat the procedure of requesting additional operations using List_ops for those related FDOs it just retrieved, or to apply other operations to the initially retrieved FDOs. Regardless of that, to get a better understanding

of their semantic meaning, the client can apply the Get_Related_Terms Operation FDO (Figure 5.5, steps 13-20) also retrieved in steps 7-9, and inspect the *topic* used to categorize the data. This operation is implemented as a Web API and associated with the topic PIT (21.T11148/b415e16fbe4ca40f2270), present in all FDOs of both datasets, which takes as value a reference to a concept entry of the UNESCO Thesaurus[6]. In order to do so, the client must provide a SPARQL request as specified in the `Required Client Input`[7], indicating where the value of the execution protocol parameter, i.e., *httpBody*, that maps to the *topic* PIT, has to be inserted, e.g.,:

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
SELECT ?prefLabel
WHERE {
    httpBody skos:prefLabel ?prefLabel .
}
```

The SPARQL request is then mapped along with the value of the topic PIT from the Target FDO's information record and the execution protocol parameters (s. Listing 5.4) to an execution map that is processed by the Executor. Note that each execution protocol parameter corresponds to the structure in Definition 18.

Listing 5.4: Get_Related_Terms Operation Overview.

```
Implementation technology: Web API
Required Client Input: parameter_type: httpBody PIT,
   description: Provide a SPARQL query using the skos:
   http://www.w3.org/2004/02/skos/core# prefix and the
   value of the parameter_type.
Input provided by DOIP/HTTP Request: SPARQL query
Execution protocol (Web API execution protocol parameters):
   http method: ['-G', '../vocabularies-unesco-org/sparql']
   http header: ['-H','"Accept: application/sparql-results+json"']
   http body: ['--data-urlencode','⟨21.T11148/b415e16fbe4ca40f2270⟩',
   '[true,"[None,' ',None]"]'[8]]
PIT for association conditions:
   topic (21.T11148/b415e16fbe4ca40f2270)
Output type: application/rdf+xml,
   http://www.w3.org/2004/02/skos/core#
```

The client can then inspect the related concepts of the topic, i.e., "Remote Sensing" and "Artificial Intelligence" for the TBBR FDOs, and "Ontology" for the OntoMatch dataset FDO. Likewise, other SPARQL requests for this operation and other operations that process

[6] https://vocabularies.unesco.org

[7] https://vocabularies.unesco.org/sparql

[8] *merge_mul_entries* condition and the data structure for constructing mapped entries of 21.T11148/b415e16fbe4ca40f2270 that are merged using a (prefix, delimiter, suffix) pattern ([None,' ',None]) formatting the request as ⟨mapped PIT content 1⟩ ⟨mapped PIT content 2⟩ ...

the contents of FDO information records can be applied. At the end of this phase, the client has evaluated the kernel information of the retrieved FDOs to assess the relevance of the underlying data resources for the given task.

Phase III - Data Retrieval and Processing. Finally, the client retrieves and processes the represented bit sequence of those FDOs identified as being of relevance, as assessed in phase II. In principle, a Management Operation can retrieve the bit sequence directly via its referencing PIT, as defined by the `Get Web Entity through PITs` activity in the FDO-Ops process model. (s. Figure 5.3). However, the operations available through the FDO-Ops framework may process the bit sequence prior to its retrieval to support the specific task intended by the user in a machine-actionable way. This may be, for example, the extraction and conversion of data into the required format or the validation of file formats. For demonstration, we designed two possible operations that can be applied to our two example datasets, respectively. In the context of the framework sequence, the operations were already retrieved in Phase II using the `List_ops` Management Operation in steps 7-9.

An operation associated with the drone image data from the TBBR dataset example is the conversion of the *numpy* file format to the *png* image format, i.e., the `Convert_Numpy_to_PNG` Operation FDO (s. Figure 5.5, steps 13-20). The actual conversion function is implemented as a Python script that is published on GitHub (s. Appendix Section A.3.1). The script accepts multiple file inputs as specified in the association requirements and returns the converted images.

Likewise, an operation associated with the controlled vocabulary of the OntoMatch dataset can be applied to validate that the corresponding files are well-formatted *SKOS RDF/XML*, i.e., by the `Validate_SKOS_RDF` Operation FDO. The validation function is also implemented as a Python script and is published in the same GitHub repository.

For both examples, the steps of getting the input files from an associated Target FDO, extracting them, retrieving the Python script itself, and applying it to the files are crucial `sub_operations` that are modeled at the level of the Operation FDO, thus wrapping the actual `main_operation`. The sequence of these `sub_operations` is then handled by the Executor such that nested operations are executed first.

Listing 5.5 outlines the `Convert_Numpy_to_PNG` operation. It follows the same overall structure as `Validate_SKOS_RDF`, but differs in execution protocol details, association requirements, and outputs. An overview of the `Validate_SKOS_RDF` operation is provided in Appendix Listing A.4.

Similarly, other available operations could be executed on the images or related data (e.g., the metadata and annotation files). At the end of this phase, the client has the original or properly (pre)processed version of the data resources represented by the previously analyzed FDOs. Depending on the research task, the client can now begin the actual analysis or data reuse procedure, applying any additional task-specific processing beyond what the FDO-Ops framework covers.

Listing 5.5: Convert_Numpy_to_PNG Operation overview.

```
Implementation technology: Python Script
Execution protocols:
 main_operation (Script execution protocol parameters):
   installations (1): ['OS command','python3, pip3']
   installations (2): ['pip install', 'numpy,pillow']
   script method: ['python3',sub_operation1]
   script argument: ['--files_dir',sub_operation2]

   sub_operation 1 (Web API execution protocol parameters):
    method: ['GET','../URL_to_script_file']

   sub_operation 2 (Web API execution protocol parameters):
    method: ['GET','21.T11148/b8457812905b83046284', 'false']
PIT for association conditions:
   digitalObjectType (21.T11148/1c699a5d1b4ad3ba4956)
   with value constellations : application/x-npy-image
      OR [application/zstd AND application/x-npy-image]
   AND digitalObjectLocation (21.T11148/b8457812905b83046284)
Output type (for Convert Numpy to PNG): image/png
```

5.5.3. Quantification of Impact by FDO-Ops

We quantify the impact of interoperability across the three operation phases by comparing an FDO-Ops client to a baseline client that works directly with heterogeneous KOSs and system-specific interfaces. For FDO-Ops we assume an FDO space reachable via an FDO-DOIP service over DOIP or DOIP/HTTP.

Definition 20 (Metrics for impact assessment.) Let ι be the number of distinct interfaces across data, metadata, operation, and programming services. Denote $\theta(a)$ as the number of synonymous terms used for the kernel metadata attribute a across the ecosystem. Let $\beta(s)$ be the number of bytes (volume) transferred for a bit sequence s, and $\beta(\hat{s})$ the number of bytes for a manipulated bit sequence $\hat{s}$, with $\beta(s) \geq \beta(\hat{s})$. Let $\mathcal{V}_F$ denote the set of nodes (vertices) in a graph database that correspond to all ingested FDOs, and $\mathcal{V}_O \subseteq \mathcal{V}_F$ denote the set of those vertices that correspond to all ingested Operation FDOs. Denote κ as the upper bound on the time complexity for an algorithm to list all operations associated with a given digital resource. Let ψ denote the upper bound on the time complexity for an algorithm to list all digital resources associated with a given operation. ■

Impacts by phase.

- **Phase I — Metadata attribute querying:** FDO-Ops enforces unambiguous typing for each kernel attribute, so a single DOIP query suffices: $\theta(a) = 1\ \forall a$. Non-FDO

paths introduce divergent terms per standard or system, so $\theta(a)$ grows with their heterogeneity.

Result: normalized syntactic access and discovery for syntactic interoperability.

- **Phases I,II&III — Interfaces:** FDO-Ops uses a uniform protocol and service API that abstract underlying APIs of the corresponding digital resources, so $\iota = 1$. Non-FDO paths add interfaces per KOS or repository, so ι grows with ecosystem diversity.

  Result: improved technical interoperability.

- **Phases II&III — Operation and data listing:** Performing the association assessment between an Operation FDO and a Target FDO based on Algorithm 2 has the time complexity $T(C_o)$, which is:

$$T(C_o) = O\left(\sum_{c \in C_o} |c|\right),$$

  since the main loop iterates over all conditions $c \in C_o$, and for each condition it iterates over all of its entries. It is thereby assumed that the number of negation condition checks n does not grow faster than the total number of entries inspected in the main loop. This quantity for the association assessment adds to κ and ψ.

  After initial assessment and ingestion into the Neo4j FDO Graph, this can be neglected, and each retrieval effort using the indexed labels is as follows:

$$T\big(|\mathcal{V}_F|, \deg(v_f)\big) = O\big(log(|\ \mathcal{V}_F\ |) + \text{degree of edges for targeted node } v_f \in \mathcal{V}_F\big),$$
$$T\big(|\mathcal{V}_O|, \deg(v_o)\big) = O\big(log(|\ \mathcal{V}_O\ |) + \text{degree of edges for targeted node } v_o \in \mathcal{V}_O\big).$$

  In our setup, the assessment will typically be performed at FDO instantiation time, and the retrieval of these associations can be performed entirely using the graph database. κ then refers to the time complexity $T\big(|\mathcal{V}_F|, \deg(v_f)\big)$ for the `List_Ops` operation, and ψ likewise to the time complexity $T\big(|\mathcal{V}_O|, \deg(v_o)\big)$ of a symmetric operation, which traverses the same edges in the graph and is requested via the same DOIP/HTTP interface. In heterogeneous KOSs, κ, ψ refer to algorithms whose time complexity depends on the number of schema inspections, terminology alignments, discovery steps (catalogs, portals), and requests to databases.

  Result: a fixed and potentially reduced time of listing available operations, lowering the client's discovery effort.

- **Phase III — Volume transfer:** FDOs support either full transfer $\beta(s)$ or reduced transfer $\beta(\hat{s})$ via standardized Digital Resource Operations (e.g., image conversion), often yielding order-of-magnitude byte savings, caching options and compute-power outsourcing for the client. In turn, the FDO-Ops service must temporarily store and process the full number of bytes. Non-FDO-Ops connected repository data storage systems typically return only full payloads of digital resources.

  Result: higher efficiency for a client when handling different data types.

5.5.4. Enhancement of Interoperability Levels by FDO-Ops

To evaluate how FDO-Ops enhances different levels of interoperability, we use the four-layer interoperability model of the EOSC IF along with the first four levels (up to level 3) of the LCIM that can be directly mapped to each other and the three types of the interoperability trilogy (s. Section 2.3.3). It is then assessed how these layers/levels are met assuming the FDO-Ops model is implemented and used by clients that want to interact with different participating systems. We argue that digital resources that are represented as FDOs become interoperable by the means of the FDO-Ops mechanisms and can be reused in an automated way. Note that some of the considered interoperability level enhancements are supported by the evaluation of the quantification of impact (s. Section 5.5.3).

Technical Interoperability (Level 1 in LCIM): Digital resources located in different storage systems remain unmodified and are represented by uniformly structured, abstract, and machine-interpretable objects, i.e., FDOs, that are available through an FDO infrastructure. This infrastructure is the basis for the FDO-Ops framework that clients can use to process FDOs. Involved systems can communicate with each other using a uniform structure and communication protocol, i.e., DOIP or DOIP/HTTP. The FDO type system introduces rules against which can be programmed, enabling specific activities that an FDO can perform. In this way, FDOs can be processed by the machine-actionable interactions between Target FDOs, Operation FDOs, FDO services and tools that utilize known execution protocols. Hence, the represented digital resources are technically interoperable.

Syntactic and Semantic Interoperability (Levels 2 and 3 in LCIM): FDOs provide a minimum metadata model with core metadata elements of hierarchically structured PITs, aggregated in a KIP. This modular metadata structure allows extensions without breaking core interoperability. PITs enforce syntactic rules for metadata key-value pairs and offer semantic specification via links to other FDOs or web entities such as definitions, ontologies, schemas, and semantic artifact repositories. By utilizing associated operations, semantic contents of related web entities, e.g., vocabularies, can be processed and interpreted. Therefore, syntactic interoperability is inherently provided by the FDO structure and communication pattern, along with semantic interoperability of kernel metadata. Semantics of digital resources represented by FDOs are interoperable through the machine-actionable capabilities of FDO-Ops. An example for this is the use of RDF-based vocabularies in conjunction with a SPARQL-API we have demonstrated in Section 5.5.2.

Legal Interoperability: A machine-actionable license format is mandatory in each FDO complying with the RDA Kernel Information Recommendations (s. Section 2.4.3.3), possibly linked to centralized knowledge sources for verification. KIP and PIT extensions support granular metadata (e.g., timestamps, license expiration) to manage copyright and version control. Operation FDOs can enable legal frameworks and access control policies.

Organizational Interoperability: Organizations share a high-level FDO service infrastructure while managing their digital resources independently. FDO services enable the provision of a clear and standardized PID policy using Handle PIDs and PITs. A consensus on the core machine-interpretable metadata that is leveraged by operations ensures interoperability within the shared FDO space. By using a controlled FDO service infrastructure as provided for FDO-Ops, organizations can ensure that only authorized entities are permitted to publish, modify, or access digital resources.

Data Interoperability: FDO-Ops fulfills the data interoperability layer insofar as it standardizes the way heterogeneous (meta)data are wrapped, identified, and typed. Every resource is exposed as an FDO with a PID and typed metadata, so different systems can recognize the same structural pattern and evaluate the same declared types. This enables automated discovery and use of data across systems even when the underlying payloads differ. What it does not fully provide is portability of the actual domain data formats, i.e., FDO-Ops standardizes the descriptive envelope and access, not the internal domain schema.

Service Interoperability: FDO-Ops fulfills the service interoperability layer by providing a standardized way to describe and invoke services, independent of their concrete API or runtime. Operations are themselves FDOs that declare which FDOs they act on and how to bind input parameters from the target's typed attributes. Because invocation follows a single pattern, a client that understands FDOs can use services from other parties without adapting to their individual APIs. This meets the layer's requirement to use one another's services despite disparate service models.

System Interoperability FDO-Ops fulfills the system interoperability layer on the infrastructural side by relying on a shared FDO-, Handle-, type infrastructure through which systems can publish, discover, and retrieve both data and executable software. Any independent system that is aware of this technology stack can locate the same objects and the same executable capabilities. These can be invoked through the agreed interface, which enables collaborative use of technologies across system boundaries. However, clients must be familiar with using a provided search engine, such as Elasticsearch. Organizational arrangements (governance, vocabulary coordination) remain external, so fulfillment is the strongest in the technical discovery and distribution aspects.

5.6. Discussion

5.6.1. Model and Implementation Design Decisions

Our categorization approach for FDO operations in conjunction with the operation phases provides a baseline for clients to decide on use case-specific interactions with an FDO space. Having operation categories helps to differentiate between generic functionalities

that are necessary for managing an FDO space, and advanced functionalities applied in the frame of specific tasks. These generic and advanced functionalities are, in turn, performed in three phases depending on the use case. Therefore, core interactions in an FDO space can be handled with a minimal set of Management Operations and a uniform protocol (e.g., DOIP). More specialized operations, including those that work on metadata (Phases I and II) or on the bit sequence (Phase III), can then be added incrementally. This avoids the common pitfall of putting all metadata and all potential functions into a single FDO. Instead, individual FDOs remain small, abstract, persistent descriptions of digital resources, while operations are attached only where applicable. This is in line with the idea that machine-actionable data should not require users or clients to know the internal structure or organizational scheme of a given repository or KOS.

The advanced DTR model is a bundled and implementation-oriented representation of the FDO type system that was developed in the earlier chapters. It introduces a scheme for the realization of different typing mechanisms on top of the foundational types that are used in conventional DTRs, along with detailed modeling for operation signatures. Whilst already specifying technology dependencies, the operation signature can then be implemented in various ways. The idea is to keep the implementation level detached from the typing level within the DTR. Representing implementations of operation signatures also as (Operation) FDOs is a strong abstraction step and aligns with the holistic view of the FDO concept to represent digital resources of all types. This leverages the FDO core model technology-agnosticism and allows wrapping existing web-based functions without changing them. The cross-walk to and from the corresponding *Operation Signature* in the advanced DTR model backs type safety of- and compliance with the involved PITs and KIPs. Whilst Operation FDOs used in PT and RT require static typechecking with their operation signatures, AT works independently due to its duck typing traits and enables introduction of extended association criteria by logical statements. However, to ensure type safety, modifications in the FDO information records, condition arrays, or type registry specifications must be reported and typechecking must be performed again. The choice of using inheritance for the declaration of subtypes is a common approach in the OOP context but analogously raises concerns with respect to the Liskov Substitution Principle. An inheriting PIT or KIP must therefore preserve the syntactic constraints (e.g., allowed value space, cardinality, structure) and semantic constraints (i.e., meaning and intended interpretation) of its parent in a genuine “is-a” relationship. An operation signature can then be safely associated via these inheritance chains. However, to remove remaining caveats, especially around behavioral and semantic compatibility and edge cases, a more thorough analysis is necessary that goes beyond the scope of this thesis.

The proposed validation algorithm for the conditions array naturally aligns with the structure of an FDO’s information record that is registered at the Handle Registry. This provides a mechanism for defining very coarse and generic up to highly granular and specific association conditions. To shift the cost from repeated algorithm performance to a write-time enrichment step, the association between an Operation and Target FDO is modeled as a PID digraph and stored in a Neo4j graph database. Neo4j’s support of property-level constraints and large-scale node/edge counts for reading [227] shows that the approach is technically feasible for large FDO spaces, at least under vertical scaling

assumptions. It can also enforce conflict rules at write time based on native property-level constraints on nodes and relationships, i.e., uniqueness, existence, type, and keys. However, what remains open in any case is the behavior in a fully federated, multi-provider environment where access constraints, security policies, and trust requirements may differ.

The execution protocol aligns with the conditions array and provides a convenient way to specify a recursive wrapper for any operation technology resource. These "active" resources can themselves be the target of other operations. This aligns the model with DOIP-based services and shows that FDO-Ops can be integrated into existing PID-based infrastructures without redesigning them. Providing the technical details of the operation directly on the information record level makes its execution straightforward, persistently available, and favorable with respect to costs, but requires high precision by the execution protocol specification and the Executor interpretation. Limitations exist with respect to large monolithic applications that will not always be reasonably wrappable. The model works best with modular APIs, scripts, or microservices. This is an acceptable constraint because it aligns with current trends towards smaller, composable services and modules. In more advanced use cases where this may not be feasible due to operation complexity, execution protocols may be purely used to activate the technical details of external specifications (e.g., Hydra, CWL) for an implemented operation by a supportive component. However, this comes with additional overhead and capabilities for the mapping procedure and the Executor.

Regarding the software architecture required for FDO-Ops, the testbed confirms that a PIT/TPM-like service in conjunction with federated adapters that infer and store associations between FDOs and their operations is a feasible approach. Co-locating an Executor with this adapter minimizes client complexity that only issues DOIP requests, while the server side handles validation, protocol mapping, and execution. This shifts complexity to the infrastructure, which is appropriate for shared FDO spaces. In production settings, however, governance and security become central: which Operation FDOs are allowed to run, under which protocol types, and on which targets? We argue that agreeing on protocol types is easier than agreeing on all possible KOSs and domain technologies. This identifies a concrete standardization requirement of protocol-type specifications for FDO execution.

5.6.2. Evaluation and Results Findings

The prototype of the advanced DTR model demonstrates new perspectives for development of next-generation DTRs that implement an FDO type system as part of an FDO infrastructure. In future setups, a deployed next-gen DTR service may provide the traceable specifications for subtyping and operation signatures that can be mapped to Operation FDOs, which are either created by the same DTR service, or a conventional DTR. This yields an FDO space where well-typed FDOs and associated operation signatures can be validated by the type system provided, whilst an FDO Graph serves as the bridge between this type system and the FDO-Ops framework.

The performed use cases with example data and operations point out that the model is domain-agnostic, but also that the actual richness of the environment depends on the community, i.e., only data and operations that communities model and publish as Target and Operation FDOs will be available. At the current maturity level, this yields good support for generic tasks such as selection, aggregation, basic validation, and format-related checks. More advanced, domain-specific processing will only emerge when communities provide the corresponding digital resources and proper execution protocols. This clearly separates what the model enables (uniform discovery, validation, invocation) from what the ecosystem must supply (high-value, domain-specific operations). Perspectively, the model could serve as an execution substrate for more intelligent services, but these are not inherent to FDO-Ops and require further work. This could be, for example, operation orchestration, workflow chaining, or AI-based recommendation of suitable operation sequences by Large Language Models as initially shown in [228].

Integration of KOS and related procedures, like cross-walks or schema dereferencing, into FDO-Ops is a central aspect. Their technical details for handling their interoperability mechanisms are abstracted from the user/client through a uniform protocol, making them truly reusable. We therefore argue that a machine-actionable environment according to FDO-Ops decreases the hurdles of leveraging KOSs, because they ease application of required procedures and assembling of individual data units when required. However, semantic issues inherent to KOSs (e.g., information loss in crosswalks or ambiguous mappings) cannot be resolved by FDO-Ops, since FDOs can only operate on what is already present in the digital resources they represent. This is a realistic boundary condition.

The FDO-Ops model mainly enhances technical and syntactic interoperability, whilst legal and organizational interoperability depend on domain communities. The technical-centric benefits are also underlined when considering the data-, service- and system interoperability layers, where standardization alone removes many barriers. Semantic interoperability is only available if semantic descriptions are linked to kernel attribute types or if semantic resources are themselves represented as FDOs, equipped with suitable operations, and related to other FDOs. This leads to the reasonable conclusion that interoperability and machine-actionability are mutually reinforcing rather than strictly hierarchical. Defining levels of machine-actionability, analogous to interoperability levels, is a logical next step and would help quantify progress in FDO-based infrastructures. From our perspective and in line with the LCIM, semantic interoperability is a necessary enabler for more advanced, possibly intelligent, machine-driven decision making that requires technical and syntactic interoperability. FDO-Ops is intended to ensure precisely those preceding interoperability levels and should therefore be understood primarily as a carrier-technology: it opens the way for automation and cross-system usage, but it still depends on domain communities to provide the semantic, legal and organizational layers needed to reach full interoperability.

6. Conclusions and Future Work

6.1. Summary

The dissertation addressed the need for a systematic, machine-actionable approach of information processing that shifts data reuse preparation work from users to digital clients.

The idea of approaching this by coordinating behavior through explicit and enforceable typing was already grounded in the concept of Digital Objects and clearly connects to the paradigm of OOP. Our proposed FDO core data model introduced in Chapter 3 builds on these concepts to position FDOs in the larger context of FAIR data spaces and assigns them the unambiguous role of machine-actionable entities as part of a meta-level. The resulting paradigm is simple, yet effective: leave the individual FAIRification approaches to the communities and make their characteristics uniformly available by an abstract representation (the FDO) of the FAIRified digital resources. As we have shown in a FAIR criteria assessment, proof of concept, and comparison to other FDO developments, the model indeed realizes strong agnosticism towards the technologies and schemes of the corresponding digital resources. As a result, FAIRification efforts can ultimately converge at the level of typed, connected, and interoperable object representations rather than at the level of repository-specific interfaces and the organization of evolving communities.

For the provision of operations through an FDO type system, the mechanisms that are also used in OOP type systems can be applied on top of PID Information Types as we deduced in Section 4.4.1, i.e., for Record-, Profile-, and Attribute Typing. We proved that the FDO Type can be defined by a rule-based typing scheme on the basis of subtyping and polymorphism, enabling a systematic and interoperable scheme for associating operations to FDOs. As in OOP, type systems are mainly meant to ensure that objects are reliably assigned the right operations, but in the FDO context, they also need to account for performance, especially in large-scale FDO spaces. We have shown how to get from abstract type system mechanisms to quantifiable association models using the FDO core data model specification for Entity-Relationship (ER) and graph representations. Quantitative and qualitative indicators as we assessed them are a solid foundation for balancing different costs like graph storage, graph traversal, extent of required client information, and FDO component reuse options, also concerning the capabilities of different infrastructure maturity levels. We find that existing base service infrastructure services, in particular Data Type Registries (DTRs), already provide a foundation for our proposed FDO typing mechanisms and corresponding operation management, but still lack required functionalities for yielding a machine-actionable setup.

This was the motivation for the FDO-Ops model and framework that we introduced in Chapter 5, along with an advanced DTR model that realizes our typing scheme in line with our FDO core data model. The DTR model is a blueprint for how associating operations to FDOs based on different typing mechanisms can be enforced based on typed FDO components. Compliant to this, FDO-Ops introduces a specific implementation approach with a focus on Attribute Typing (AT) and categorization of operations for better coordination of object behavior. Following the idea of Duck Typing, AT is the most practical typing mechanism, as it dynamically checks type compliance and can therefore be also enforced outside of DTR systems. Nevertheless, the corresponding operation signatures are subject to DTR specification and are mapped to a concrete implementation represented by Operation FDOs, which realizes a holistic approach. Another important aspect we realized is the pre-computation and storage of AT-checked relationships in an FDO graph to shift costs from repeated dynamic typechecking to graph storage requests. The resulting possibility of interacting with an FDO space through identifier schemes on behalf of the Digital Object Interface Protocol (i.e., DOIP/HTTP) is the ultimate brick for an interoperable system. The conjunction with Knowledge Organization Systems and technologies used for FAIRification is evident: FDOs are a generic carrier technology that leverages existing technologies and schemes, which is in line with what we defined in the FDO core data model. FDO-Ops is a realization for that paradigm that outlines how machines can make structured incremental decisions (e.g., filtering, validating, and converting) without requiring users to fully comprehend repository- and domain-specific tooling. Representing operations and execution details as identifiable first-class objects reduces the effort to invoke the “right” function on the “right” object in a heterogeneous environment.

The richness of a resulting FDO space finally depends on communities publishing (Operation) FDOs, maintaining them, and specifying reliable execution protocols. More advanced data processing capabilities require sustained community efforts that go beyond the scope of this dissertation. Hence, our FDO-Ops testbed examples demonstrate technical and domain-independent feasibility under controlled assumptions, but open questions remain (e.g., security and performance aspects) for transferability to fully federated, large-scale, and multi-community environments.

6.2. Future Work

The degree of rigor introduced throughout this work makes precise assumptions on the technical and syntactic traits of FDOs and the required infrastructure, which is in line with earlier efforts that have been elaborated by works on Digital Objects. Future works should focus on model semantics for a modular design, i.e., how it can be ensured that individual digital resources are represented as FDOs and linked to related resources (also FDOs), enabling their joint use through appropriate operations. This also relates to the capabilities of FDO type systems, for which our formalization provides a foundation. Future works should address further aspects of type theory applied in the FDO context such as

other forms of polymorphism beyond subtyping, or rules for safe type substitutability when using inheritance. In the future, even more thorough type-checking and validation toolchains could then emerge.

We supported our claims by proof of concept implementations with two data sets of distinct research domains throughout the work, but more sophisticated evaluation methods and broader implementation examples would add robustness in future setups. For example, scalability studies for big data cases, considering benchmarks and error rates that cover heterogeneous repositories, multiple domains, and diverse operation technologies. Other usability studies could introduce refined metrics to quantify machine-actionability and its relationship to interoperability levels.

Since many practical tasks require chaining multiple operations, future work on top of FDO-Ops should explore composition semantics for Operation FDOs with respect to typed inputs/outputs, or preconditions/postconditions. Also relevant are orchestration mechanisms or recommendation approaches for operation sequences (including heuristic or learning-based methods) while preserving type safety and the challenge of leveraging semantics in general by integrating Knowledge Organization Systems. One aspect thereby is to define feasible conventions for representing common technologies that make use of Semantic Web methods (e.g., REST/HTTP, SPARQL endpoints, workflow runners) as execution protocol types. Certainly, latest advancements in Large Language Model breakthroughs enable new possibilities for speeding up semantic interpretation and orchestration tasks, but are still prone to reproducibility and reliability issues. With respect to security, future works should also engage in defining and assessing policy-bounded executions (e.g., authorization, Access Control Lists for Operation FDOs and protocol types).

Overall, whilst the dissertation suggests and demonstrates a way of how FDOs work for different domains, an ecosystem must supply the infrastructure and curated resources that make them usable in practice. This would support machine-actionable data not only for more efficient, but also for reliable research. In an era where Artificial Intelligence both consumes and produces ever-larger data volumes, FDOs could provide the rule-based, persistent, and manageable foundation needed to keep trustworthy curated data reusable in the long term.

Bibliography

[1] Mark D. Wilkinson et al. “The FAIR Guiding Principles for scientific data management and stewardship”. In: *Scientific Data* 3 (2016), p. 160018. DOI: 10.1038/sdata.2016.18.

[2] Annika Jacobsen et al. “FAIR Principles: Interpretations and Implementation Considerations”. In: *Data Intelligence* 2.1-2 (Jan. 2020), pp. 10–29. ISSN: 2641-435X. DOI: 10.1162/dint_r_00024.

[3] Peter Wittenburg et al. *Digital Objects as Drivers towards Convergence in Data Infrastructures*. Version v1. 2019. URL: https://b2share.eudat.eu/records/b605d85809ca45679b110719b6c6cb11 (visited on 01/24/2026).

[4] Andreas Hutterer, Barbara Krumay, and Manuel Mühlburger. “What Constitutes a Dataspace? Conceptual Clarity beyond Technical Aspects”. In: *AMCIS 2023 Proceedings. 5*. 2023. URL: https://aisel.aisnet.org/amcis2023/eco_systems/eco_systems/5 (visited on 01/27/2026).

[5] Erik Schultes and Peter Wittenburg. “FAIR Principles and Digital Objects: Accelerating Convergence on a Data Infrastructure”. In: *Data Analytics and Management in Data Intensive Domains. DAMDID/RCDL 2018*. Ed. by Yannis Manolopoulos and Sergey Stupnikov. Cham: Springer International Publishing, 2019, pp. 3–16. ISBN: 978-3-030-23584-0. DOI: 10.1007/978-3-030-23584-0_1.

[6] *Research Data Alliance (RDA)*. Research Data Alliance. URL: https://www.rd-alliance.org/ (visited on 02/02/2026).

[7] *FAIR Digital Objects*. FAIR Digital Objects Forum (FDO Forum). URL: https://fairdo.org/ (visited on 02/02/2026).

[8] *European Open Science Cloud (EOSC)*. European Commission. URL: https://open-science-cloud.ec.europa.eu/ (visited on 02/02/2026).

[9] Nicolas Blumenröhr et al. “FAIR Digital Objects for the Realization of Globally Aligned Data Spaces”. In: *2024 IEEE International Conference on Big Data (BigData)*. ISSN: 2573-2978. Dec. 2024, pp. 374–383. DOI: 10.1109/BigData62323.2024.10825796.

[10] Nicolas Blumenröhr, Neil MacKinnon, and Rossella Aversa. “FAIR Data Management Workflow for MRI Data”. In: *Proceedings of the 3rd Workshop on Metadata and Research (objects) Management for Linked Open Science (DaMaLOS 2023) - Co-located with Extended Semantic Web Conference (ESWC)*. May 2023, p. 5349. DOI: 10.4126/FRL01-006444992.

[11] Zoe Mayer et al. “Thermal Bridges on Building Rooftops”. In: *Scientific Data* 10.268 (2023). ISSN: 2052-4463. DOI: 10.1038/s41597-023-02140-z.

[12] Felix Kraus et al. “A Gold Standard Benchmark Dataset for Digital Humanities”. In: *Proceedings of the 19th International Workshop on Ontology Matching - Co-located with the 23rd International Semantic Web Conference (ISWC 2024)*. Ed. by Ernesto Jiménez-Ruiz et al. Vol. 3897. CEUR Workshop Proceedings. Baltimore, USA: CEUR, Nov. 2024, pp. 1–17. DOI: 10.5445/IR/1000178023.

[13] Nicolas Blumenröhr et al. “A Comparative Analysis of Modeling Approaches for the Association of FAIR Digital Objects Operations”. In: *Data Science Journal* 24 (2025), p. 22. ISSN: 1683-1470. DOI: 10.5334/dsj-2025-022.

[14] Nicolas Blumenröhr and Rossella Aversa. “From implementation to application: FAIR digital objects for training data composition”. In: *Research Ideas and Outcomes* 9 (2023), e108706. ISSN: 2367-7163. DOI: 10.3897/rio.9.e108706.

[15] Nicolas Blumenröhr et al. “FDO-Ops: A Model for Machine-Actionable FAIR Digital Objects to Enhance Interoperability”. In: *Data Intelligence* (2026). ISSN: 2641-435X. DOI: 10.3724/2096-7004.di.2025.0345.

[16] Maximilian Inckmann, Nicolas Blumenröhr, and Rossella Aversa. “Towards Machine-actionable FAIR Digital Objects with a Typing Model That Enables Operations”. In: *2025 IEEE International Conference on eScience (eScience)*. Chicago, IL, USA: IEEE, Sept. 2025, pp. 111–120. ISBN: 979-8-3315-9145-8. DOI: 10.1109/eScience65000.2025.00022.

[17] Grady Booch et al. *Object-oriented analysis and design with applications, third edition*. Third. Addison-Wesley Professional, Mar. 2007. ISBN: 978-0-201-89551-3.

[18] Luca Cardelli. “Type systems”. In: *ACM Comput. Surv.* 28.1 (Mar. 1996), pp. 263–264. ISSN: 0360-0300. DOI: 10.1145/234313.234418.

[19] Luca Cardelli. “Type Systems”. In: *CRC Handbook of Computer Science and Engineering*. Ed. by Allen B. Tucker. 2nd ed. CRC Press, 2004. Chap. 97. URL: http://lucacardelli.name/papers/typesystems.pdf (visited on 01/22/2026).

[20] Benjamin C. Pierce. *Types and Programming Languages*. 1st ed. Cambridge, Massachusetts: MIT Press, Feb. 2002. ISBN: 0-262-16209-1.

[21] Barbara Liskov and John Guttag. *Abstraction and specification in program development*. Cambridge, MA, USA: MIT Press, 1986. ISBN: 978-0-262-12112-5.

[22] D. L. Parnas. “On the criteria to be used in decomposing systems into modules”. In: *Communications of the ACM* 15.12 (1972), pp. 1053–1058. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/361598.361623.

[23] Barbara Liskov and Stephen Zilles. “Programming with abstract data types”. In: *SIGPLAN Not.* 9.4 (1974), pp. 50–59. ISSN: 0362-1340. DOI: 10.1145/942572.807045.

[24] Luca Cardelli and Peter Wegner. “On understanding types, data abstraction, and polymorphism”. In: *ACM Comput. Surv.* 17.4 (1985), pp. 471–523. ISSN: 0360-0300. DOI: 10.1145/6041.6042.

[25] William R. Cook. "On understanding data abstraction, revisited". In: *SIGPLAN Not.* 44.10 (2009), pp. 557–572. ISSN: 0362-1340. DOI: 10.1145/1639949.1640133.

[26] Oscar Nierstrasz. "A survey of object-oriented concepts". In: *Object-Oriented Concepts, Databases, and Applications*. New York, NY, USA: Association for Computing Machinery, 1989, pp. 3–21. ISBN: 0201144107. DOI: 10.1145/63320.66468.

[27] Nathanael Schärli, Andrew P. Black, and Stéphane Ducasse. "Object-oriented encapsulation for dynamically typed languages". In: *SIGPLAN Not.* 39.10 (2004), pp. 130–149. ISSN: 0362-1340. DOI: 10.1145/1035292.1028988.

[28] Bertrand Meyer. *Object-Oriented Software Construction*. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1997. ISBN: 978-0136291558.

[29] Chandrasekhar Boyapati, Barbara Liskov, and Liuba Shrira. "Ownership types for object encapsulation". In: *SIGPLAN Not.* 38.1 (2003), pp. 213–223. ISSN: 0362-1340. DOI: 10.1145/640128.604156.

[30] Barbara Liskov. "Keynote address - data abstraction and hierarchy". In: *SIGPLAN Not.* 23.5 (1987), pp. 17–34. ISSN: 0362-1340. DOI: 10.1145/62139.62141.

[31] Barbara H. Liskov and Jeannette M. Wing. "A behavioral notion of subtyping". In: *ACM Trans. Program. Lang. Syst.* 16.6 (1994), pp. 1811–1841. ISSN: 0164-0925. DOI: 10.1145/197320.197383.

[32] Luca Cardelli. "A semantics of multiple inheritance". In: *Information and Computation* 76.2 (1988), pp. 138–164. ISSN: 0890-5401. DOI: 10.1016/0890-5401(88)90007-7.

[33] William R. Cook, Walter Hill, and Peter S. Canning. "Inheritance is not subtyping". In: *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '90. San Francisco, California, USA: Association for Computing Machinery, 1989, pp. 125–135. ISBN: 0897913434. DOI: 10.1145/96709.96721.

[34] Nevena Milojkovic, Mohammad Ghafari, and Oscar Nierstrasz. "It's Duck (Typing) Season!" In: *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. May 2017, pp. 312–315. DOI: 10.1109/ICPC.2017.10.

[35] Randall Davis, Howard Shrobe, and Peter Szolovits. "What Is a Knowledge Representation?" In: *AI Magazine* 14.1 (1993), pp. 17–33. ISSN: 2371-9621. DOI: 10.1609/aimag.v14i1.1029.

[36] Peter Pin-Shan Chen. "The entity-relationship model—toward a unified view of data". In: *ACM Trans. Database Syst.* 1.1 (1976), pp. 9–36. ISSN: 0362-5915. DOI: 10.1145/320434.320440.

[37] Bernhard Thalheim. *Entity-Relationship Modeling*. Berlin, Heidelberg: Springer, 2000. ISBN: 978-3-662-04058-4. DOI: 10.1007/978-3-662-04058-4.

[38] Bernhard Thalheim. "The Strength of ER Modeling". In: *Conceptual Modeling: Current Issues and Future Directions*. Ed. by G. Goos et al. Berlin, Heidelberg: Springer, 1999, pp. 227–242. ISBN: 978-3-540-48854-5. DOI: 10.1007/3-540-48854-5_18.

[39] Marcia Lei Zeng. "Knowledge Organization Systems (KOS)". In: *Knowledge Organization* 35.2-3 (2008), pp. 160–182. ISSN: 0943-7444. DOI: 10.5771/0943-7444-2008-2-3-160.

[40] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* 1st edition. HarperOne, 1999. ISBN: 0-06-251586-1.

[41] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web: A New Form of Web Content that is Meaningful to Computers will Unleash a Revolution of New Possibilities". In: *Linking the World's Information.* Ed. by Oshani Seneviratne and James Hendler. 1st ed. New York, NY, USA: ACM, 2023, pp. 91–103. ISBN: 9798400707940. DOI: 10.1145/3591366.3591376.

[42] Richard Cyganiak, David Hyland-Wood, and Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax.* W3C Recommendation. 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ (visited on 01/24/2025).

[43] Guus Schreiber and Yves Raimond. *RDF 1.1 Primer.* W3C Working Group Note. 2014. URL: https://www.w3.org/TR/rdf11-primer/ (visited on 11/06/2023).

[44] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview (Second Edition).* W3C Recommendation. 2012. URL: https://www.w3.org/TR/owl2-overview/ (visited on 11/06/2025).

[45] Katherine Thornton et al. "Using Shape Expressions (ShEx) to Share RDF Data Models and to Guide Curation with Rigorous Validation". In: *The Semantic Web.* Ed. by Pascal Hitzler et al. Cham: Springer International Publishing, 2019, pp. 606–620. ISBN: 978-3-030-21348-0. DOI: 10.1007/978-3-030-21348-0_39.

[46] Holger Knublauch and Dimitris Kontokostas. *Shapes Constraint Language (SHACL).* W3C Recommendation. 2017. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/ (visited on 01/14/2026).

[47] Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF.* W3C Recommendation. 2008. URL: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/ (visited on 11/14/2025).

[48] Tim Berners-Lee. *Linked Data.* W3C Design Issues. 2006. URL: https://www.w3.org/DesignIssues/LinkedData.html (visited on 11/14/2025).

[49] Sandi Pohorec, Milan Zorman, and Peter Kokol. "Analysis of approaches to structured data on the web". In: *Computer Standards & Interfaces* 36.1 (2013), pp. 256–262. ISSN: 0920-5489. DOI: 10.1016/j.csi.2013.06.003.

[50] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked Data: The Story so Far". In: *International Journal on Semantic Web and Information Systems* 5 (2009), pp. 1–22. DOI: 10.4018/jswis.2009081901.

[51] Jenn Riley. *Understanding Metadata: What is Metadata, and What is it For?* NISO Primer. Baltimore, MD: National Information Standards Organization (NISO), 2017. ISBN: 978-1-937522-72-8.

[52] J. Paul Getty Trust. *Introduction to Metadata*. Ed. by Murtha Baca. 2nd ed. Version 3.0. Los Angeles: Getty Publications, 2008. ISBN: 978-0-89236-896-9. URL: https://d2aohiyo3d3idm.cloudfront.net/publications/virtuallibrary/0892368969.pdf (visited on 01/24/2026).

[53] Daniel S. Katz, Morane Gruenpeter, and Tom Honeyman. "Taking a fresh look at FAIR for research software". In: *Patterns* 2.3 (2021), p. 100222. ISSN: 2666-3899. DOI: https://doi.org/10.1016/j.patter.2021.100222.

[54] Carole Goble et al. "FAIR Computational Workflows". In: *Data Intelligence* 2.1-2 (2024), pp. 108–121. ISSN: 2096-7004. DOI: 10.1162/dint_a_00033.

[55] Daniel S. Katz, Fotis Psomopoulos, and Leyla Jael Castro. "Working Towards Understanding the Role of FAIR for Machine Learning". In: *Proc. 2nd Workshop on Data and Research Objects Management for Linked Open Science*. Virtual Conference, 2021, pp. 1–6. DOI: 10.4126/FRL01-006429415.

[56] Lars Vogt et al. "Suggestions for extending the FAIR Principles based on a linguistic perspective on semantic interoperability". In: *Scientific Data* 12.688 (2025). ISSN: 2052-4463. DOI: 10.1038/s41597-025-05011-x.

[57] Michelle Barker et al. "Introducing the FAIR Principles for research software". In: *Scientific Data* 9.622 (2022). Publisher: Nature Publishing Group. ISSN: 2052-4463. DOI: 10.1038/s41597-022-01710-x.

[58] Philip A. Bernstein and Umeshwar Dayal. "An Overview of Repository Technology". In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 705–713. ISBN: 978-1-55860-153-6.

[59] Massimiliano Assante et al. "Are Scientific Data Repositories Coping with Research Data Publishing?" In: *Data Science Journal* 15.0 (2016), p. 6. ISSN: 1683-1470. DOI: 10.5334/dsj-2016-006.

[60] Zilong He and Wei Fang. "Research data management in institutional repositories: an architectural approach using data lakehouses". In: *Digital Library Perspectives* 41.1 (2024), pp. 145–178. ISSN: 2059-5816. DOI: 10.1108/DLP-02-2024-0022.

[61] João Pedro de Carvalho Castro et al. "The BigFAIR Architecture: Enabling Big Data Analytics in FAIR-compliant Repositories". In: *J. Data and Information Quality* 17.4 (2025), 24:1–24:28. ISSN: 1936-1955. DOI: 10.1145/3774755.

[62] James Dixon. *Pentaho, Hadoop, and Data Lakes*. Blog post. Oct. 14, 2010. URL: https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/ (visited on 12/19/2025).

[63] Shaker H. Ali El-Sappagh, Abdeltawab M. Ahmed Hendawi, and Ali Hamed El Bastawissy. "A proposed model for data warehouse ETL processes". In: *Journal of King Saud University - Computer and Information Sciences* 23.2 (2011), pp. 91–104. ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2011.05.005.

[64] Sam Sun, Larry Lannom, and Brian Boesch. *Handle System Overview*. Request for Comments RFC 3650. CNRI, 2003. DOI: 10.17487/RFC3650.

[65] Bill Rosenblatt. "The Digital Object Identifier: Solving the Dilemma of Copyright Protection Online". In: *Journal of Electronic Publishing* 3.2 (1997). Publisher: Michigan Publishing, University of Michigan Library. ISSN: 1080-2711. DOI: 10.3998/3336451.0003.204.

[66] Jens Klump and Robert Huber. "20 Years of Persistent Identifiers – Which Systems are Here to Stay?" In: *Data Science Journal* 16.0 (2017), p. 9. ISSN: 1683-1470. DOI: 10.5334/dsj-2017-009.

[67] Angelo Salatino et al. "A survey of knowledge organization systems of research fields: Resources and challenges". In: *Quantitative Science Studies* 6 (2025), pp. 567–610. ISSN: 2641-3337. DOI: 10.1162/qss_a_00363.

[68] Bernadette Hyland, Ghislain Atemezing, and Boris Villazón-Terrazas. *Best Practices for Publishing Linked Data*. W3C Working Group Note. Published by the W3C Government Linked Data Working Group. 2014. URL: https://www.w3.org/TR/2014/NOTE-ld-bp-20140109/ (visited on 01/24/2026).

[69] Herbert Van de Sompel et al. "Persistent Identifiers for Scholarly Assets and the Web: The Need for an Unambiguous Mapping". In: *International Journal of Digital Curation* 9.1 (2014), pp. 331–342. ISSN: 1746-8256. DOI: 10.2218/ijdc.v9i1.320.

[70] Sören Auer et al. "Diachronic linked data: towards long-term preservation of structured interrelated information". In: *Proceedings of the First International Workshop on Open Data*. WOD '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 31–39. ISBN: 978-1-4503-1404-6. DOI: 10.1145/2422604.2422610.

[71] Phil Archer, Stijn Goedertier, and Nikolaos Loutas. *D7.1.3 - Study on persistent URIs, with identification of best practices and recommendations on the topic for the MSs and the EC*. Deliverable / Report D7.1.3. European Commission, ISA Programme, 2012.

[72] Sean Bechhofer et al. "Why linked data is not enough for scientists". In: *Future Generation Computer Systems*. Special section: Recent advances in e-Science 29.2 (2013), pp. 599–611. ISSN: 0167-739X. DOI: 10.1016/j.future.2011.08.004.

[73] Carl Lagoze and David Fielding. "Defining Collections in Distributed Digital Libraries". In: *D-Lib Magazine* 4.11 (1998). ISSN: 1082-9873. URL: https://www.dlib.org/dlib/november98/lagoze/11lagoze.html (visited on 12/18/2025).

[74] Sean Bechhofer et al. "Research Objects: Towards Exchange and Reuse of Digital Knowledge". In: *Nature Precedings* (2010). ISSN: 1756-0357. DOI: 10.1038/npre.2010.4626.1.

[75] Weiland Claus et al. *FDO Machine Actionability*. Version 2.2. 2022. DOI: 10.5281/zenodo.7825650.

[76] Rita Suzana Pitangueira Maciel et al. "Systems Interoperability Types: A Tertiary Study". In: *ACM Comput. Surv.* 56.10 (2024), 254:1–254:37. ISSN: 0360-0300. DOI: 10.1145/3659098.

[77] *New European Interoperability Framework: Promoting seamless services and data flows for European public administrations*. Luxembourg: European Commission, 2017. DOI: 10.2799/78681.

[78] Directorate-General for Research and Innovation (European Commission) et al. *EOSC interoperability framework: report from the EOSC Executive Board Working Groups FAIR and Architecture*. eng. LU: Publications Office of the European Union, 2021. ISBN: 978-92-76-28949-4. URL: https://data.europa.eu/doi/10.2777/620649 (visited on 11/14/2023).

[79] Mersedeh Sadeghi et al. "Interoperability of heterogeneous Systems of Systems: from requirements to a reference architecture". In: *The Journal of Supercomputing* 80.7 (2024), pp. 8954–8987. ISSN: 1573-0484. DOI: 10.1007/s11227-023-05774-3.

[80] Andreas Tolk and James A. Muguira. "The Levels of Conceptual Interoperability Model". In: *Proceedings of the 2003 Fall Simulation Interoperability Workshop*. Vol. 7. Simulation Interoperability Standards Organization (SISO). Orlando, FL, USA, 2003, pp. 1–11. URL: https://www.mscoe.org/content/uploads/2017/12/Tolk-Muguira-The-Levels-of-Conceptual-Interoperability-Models.pdf (visited on 05/27/2025).

[81] Charles D Turnitsa. "Extending the levels of conceptual interoperability model". In: *Proceedings IEEE summer computer simulation conference, IEEE CS Press*. 2005.

[82] Andreas Tolk, Lisa Jean Bair, and Saikou Y. Diallo. "Supporting Network Enabled Capability by extending the Levels of Conceptual Interoperability Model to an interoperability maturity model". In: *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 10.2 (2013), pp. 145–160. DOI: 10.1177/1548512911428457.

[83] Jakob Axelsson. "Achieving System-of-Systems Interoperability Levels Using Linked Data and Ontologies". In: *INCOSE International Symposium* 30.1 (2020), pp. 651–665. ISSN: 2334-5837. DOI: 10.1002/j.2334-5837.2020.00746.x.

[84] Gabriel da Silva Serapião Leal, Wided Guédria, and Hervé Panetto. "Interoperability assessment: A systematic literature review". In: *Computers in Industry* 106 (2019), pp. 111–132. ISSN: 0166-3615. DOI: 10.1016/j.compind.2019.01.002.

[85] Edward Curry, Simon Scerri, and Tuomo Tuikka. "Data Spaces: Design, Deployment, and Future Directions". In: *Data Spaces : Design, Deployment and Future Directions*. Ed. by Edward Curry, Simon Scerri, and Tuomo Tuikka. Cham: Springer International Publishing, 2022, pp. 1–17. ISBN: 978-3-030-98636-0.

[86] Michael Franklin, Alon Halevy, and David Maier. "From databases to dataspaces: a new abstraction for information management". In: *ACM SIGMOD Record* 34.4 (2005), pp. 27–33. ISSN: 0163-5808. DOI: 10.1145/1107499.1107502.

[87] Boris Otto, Michael ten Hompel, and Stefan Wrobel. "International Data Spaces". In: *Digital Transformation*. Ed. by Reimund Neugebauer. Berlin, Heidelberg: Springer, 2019, pp. 109–128. ISBN: 978-3-662-58134-6. DOI: 10.1007/978-3-662-58134-6_8.

[88] Edward Curry. "The Big Data Value Chain: Definitions, Concepts, and Theoretical Approaches". In: *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*. Ed. by José María Cavanillas, Edward Curry, and Wolfgang Wahlster. Cham: Springer International Publishing, 2016, pp. 29–37. ISBN: 978-3-319-21569-3. DOI: 10.1007/978-3-319-21569-3_3.

[89] M. Hauff et al. "FAIRness in dataspaces : the role of semantics for data management". In: *SDS 2024 : Semantics in Dataspaces 2024 : Proceedings of the Second International Workshop on Semantics in Dataspaces (SDS 2024) co-located with the 21st Extended Semantic Web Conference (ESWC 2024)*. Vol. 3705. ISSN: 1613-0073. 2024. URL: http://hdl.handle.net/1854/LU-01J6SGPAD72CZQ0MWFY2BPKNXM (visited on 11/13/2025).

[90] A. Anil Sinaci et al. "From Raw Data to FAIR Data: The FAIRification Workflow for Health Research". In: *Methods of Information in Medicine* 59.S 01 (2020). Publisher: Georg Thieme Verlag KG, e21–e32. ISSN: 0026-1270, 2511-705X. DOI: 10.1055/s-0040-1713684.

[91] GO FAIR. *FAIRification Process*. URL: https://www.go-fair.org/fair-principles/fairification-process/ (visited on 01/27/2026).

[92] Robert Kahn and Robert Wilensky. *A Framework for Distributed Digital Object Services*. Technical report cnri.dlib/tn95-01. Corporation for National Research Initiatives, 1995. URL: https://www.cnri.reston.va.us/home/cstr/arch/k-w.html (visited on 01/06/2026).

[93] Robert Kahn and Robert Wilensky. "A framework for distributed digital object services". In: *International Journal on Digital Libraries* 6.2 (2006), pp. 115–123. ISSN: 1432-5012, 1432-1300. DOI: 10.1007/s00799-005-0128-x.

[94] Kenneth P. Birman. "CORBA: The Common Object Request Broker Architecture". In: *Reliable Distributed Systems: Technologies, Web Services, and Applications*. New York, NY: Springer New York, 2005, pp. 119–140. ISBN: 978-0-387-27601-4. DOI: 10.1007/0-387-27601-7_6.

[95] Mahamat Ali Hisseine et al. "A review of digital object architecture and handle system: Development, current applications and prospective". In: *Internet of Things* 26 (2024), p. 101230. ISSN: 2542-6605. DOI: 10.1016/j.iot.2024.101230.

[96] DONA Foundation. *Digital Object Interface Protocol Specification*. 2018. URL: https://www.dona.net/sites/default/files/2018-11/DOIPv2Spec_1.pdf (visited on 07/28/2021).

[97] Norman Paskin. "Digital Object Identifier (DOI®) System". In: *Encyclopedia of Library and Information Sciences*. Ed. by Marcia J. Bates and Mary Niles Maack. 3rd ed. CRC Press / Taylor & Francis, 2010, pp. 1586–1592. DOI: 10.1081/E-ELIS3-120044418.

[98] Gary Berg-Cross, Raphael Ritz, and Peter Wittenburg. *Data Foundation and Terminology Work Group Products*. 2015. DOI: 10.15497/06825049-8CA4-40BD-BCAF-DE9F0EA2FADF.

[99] Nikolay A. Skvortsov. "A FAIR Problem-Solving Lifecycle Architecture". In: *Supplementary Proceedings of the XXIII International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2021)* (Oct. 26–29, 2021). Ed. by Alexei Pozanenko et al. Vol. 3036. CEUR Workshop Proceedings. Moscow, Russia, 2021, pp. 79–93. URL: https://ceur-ws.org/Vol-3036/paper06.pdf (visited on 12/29/2025).

[100] E. Seidewitz. "What models mean". In: *IEEE Software* 20.5 (2003), pp. 26–32. ISSN: 1937-4194. DOI: 10.1109/MS.2003.1231147.

[101] Tobias Weigel et al. "A Framework for Extended Persistent Identification of Scientific Assets". In: *Data Science Journal* 12.0 (Mar. 2013), pp. 10–22. ISSN: 1683-1470. DOI: 10.2481/dsj.12-036.

[102] Tobias Weigel, Stephan Kindermann, and Michael Lautenschlager. "Actionable Persistent Identifier Collections". In: *Data Science Journal* 12.0 (2014), pp. 191–206. ISSN: 1683-1470. DOI: 10.2481/dsj.12-058.

[103] Martin Fenner and Amir Aryani. *Introducing the PID Graph*. 2019. URL: https://datacite.org/blog/introducing-the-pid-graph/ (visited on 11/14/2023).

[104] Helena Cousijn et al. "Connected Research: The Potential of the PID Graph". In: *Patterns (New York, N.Y.)* 2.1 (2021), p. 100180. ISSN: 2666-3899. DOI: 10.1016/j.patter.2020.100180.

[105] Kostas Saidis, George Pyrounakis, and Mara Nikolaidou. "On the Effective Manipulation of Digital Objects: A Prototype-Based Instantiation Approach". In: *Research and Advanced Technology for Digital Libraries*. Ed. by Andreas Rauber, Stavros Christodoulakis, and A. Min Tjoa. Berlin, Heidelberg: Springer, 2005, pp. 13–24. ISBN: 978-3-540-31931-3. DOI: 10.1007/11551362_2.

[106] Kostas Saidis et al. "Digital Object Prototypes: An Effective Realization of Digital Object Types". In: *Research and Advanced Technology for Digital Libraries*. Ed. by Julio Gonzalo et al. Berlin, Heidelberg: Springer, 2006, pp. 123–134. ISBN: 978-3-540-44638-5. DOI: 10.1007/11863878_11.

[107] Larry Lannom, Daan Broeder, and Giridhar Manepalli. *RDA Data Type Registries Working Group Output*. 2015. DOI: 10.15497/A5BCD108-ECC4-41BE-91A7-20112FF77458.

[108] Tobias Weigel, Timothy DiLauro, and Thomas Zastrow. *PID Information Types WG final deliverable*. 2015. DOI: 10.15497/FDAA09D5-5ED0-403D-B97A-2675E1EBE786.

[109] Ulrich Schwardmann. "Automated schema extraction for PID information types". In: *Proc. IEEE International Conference on Big Data (Big Data)*. 2016, pp. 3036–3044. DOI: 10.1109/BigData.2016.7840957.

[110] ePIC Consortium. *Persistent Identifiers for eResearch*. 2025. URL: https://www.pidconsortium.net/ (visited on 12/24/2025).

[111] Tobias Weigel et al. *RDA Recommendation on PID Kernel Information*. Version 2. 2019. DOI: 10.15497/rda00031.

[112] Tobias Weigel et al. "Making Data and Workflows Findable for Machines". In: *Data Intelligence* 2.1-2 (2020), pp. 40–46. ISSN: 2641-435X. DOI: 10.1162/dint_a_00026.

[113] Keith Jeffery et al. “Not Ready for Convergence in Data Infrastructures”. In: *Data Intelligence* 3.1 (2021), pp. 116–135. ISSN: 2641-435X. DOI: 10.1162/dint_a_00084.

[114] Peter Wittenburg. “From Persistent Identifiers to Digital Objects to Make Data Science More Efficient”. In: *Data Intelligence* 1.1 (2024), pp. 6–20. ISSN: 2096-7004. DOI: 10.1162/dint_a_00004.

[115] Annika Jacobsen et al. “A Generic Workflow for the Data FAIRification Process”. In: *Data Intelligence* 2.1-2 (2020), pp. 56–65. ISSN: 2641-435X. DOI: 10.1162/dint_a_00028.

[116] N. A. Kalinin and N. A. Skvortsov. “Difficulties of FAIR Principles Implementation in Cross-Domain Research Infrastructures”. In: *Lobachevskii Journal of Mathematics* 44.1 (2023), pp. 147–156. ISSN: 1818-9962. DOI: 10.1134/S199508022301016X.

[117] Mark D. Wilkinson et al. “A design framework and exemplar metrics for FAIRness”. In: *Scientific Data* 5.180118 (2018). ISSN: 2052-4463. DOI: 10.1038/sdata.2018.118.

[118] Anusuriya Devaraju and Robert Huber. “An automated solution for measuring the progress toward FAIR research data”. In: *Patterns* 2.11 (2021), p. 100370. ISSN: 2666-3899. DOI: https://doi.org/10.1016/j.patter.2021.100370.

[119] MD Wilkinson et al. “Community-driven governance of FAIRness assessment: an open issue, an open discussion”. In: *Open Research Europe* 2.146 (2023). DOI: 10.12688/openreseurope.15364.2.

[120] Daniel J. B. Clarke et al. “FAIRshake: Toolkit to Evaluate the FAIRness of Research Digital Resources”. In: *Cell Systems* 9.5 (2019), pp. 417–421. ISSN: 2405-4712. DOI: 10.1016/j.cels.2019.09.011.

[121] Yifan Chen et al. “A FAIR and AI-ready Higgs boson decay dataset”. In: 9.31 (2022). ISSN: 2052-4463. DOI: 10.1038/s41597-021-01109-0.

[122] Philippe Lamarre et al. “A generic framework to better understand and compare FAIRness measures”. In: *Knowledge Engineering and Knowledge Management: 24th International Conference, EKAW 2024, Amsterdam, The Netherlands, November 26–28, 2024, Proceedings*. Amsterdam, The Netherlands: Springer-Verlag, 2024, pp. 291–308. ISBN: 978-3-031-77791-2. DOI: 10.1007/978-3-031-77792-9_18.

[123] E. A. Huerta et al. “FAIR for AI: An interdisciplinary and international community building perspective”. In: *Scientific Data* 10.487 (2023). ISSN: 2052-4463. DOI: 10.1038/s41597-023-02298-6.

[124] Yuji Roh, Geon Heo, and Steven Euijong Whang. “A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective”. In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2021), pp. 1328–1347. ISSN: 1558-2191. DOI: 10.1109/TKDE.2019.2946162.

[125] José Delgado. “Interoperability Effect in Big Data”. In: *Handbook of Smart Cities*. Ed. by Juan Carlos Augusto. Cham: Springer International Publishing, 2021, pp. 875–901. ISBN: 978-3-030-69698-6. DOI: 10.1007/978-3-030-69698-6_48.

[126] Philipp Wieder et al. “Data Economy and FDOs”. In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1413.

[127] George Pyrounakis, Mara Nikolaidou, and Michael Hatzopoulos. "Digital Object Abstraction Layer: A Middleware for Building Federated Digital Libraries". In: *22nd International Conference on Parallel and Distributed Computing and Communication Systems (PDCCS 2009), September 24–26, 2009, Marriott Louisville Downtown, Louisville, Kentucky, USA*. Ed. by James H. Graham and Anthony Skjellum. ISCA, 2009, pp. 172–178. URL: https://mara.dit.people.hua.gr/publications/pdccs09.pdf (visited on 12/29/2025).

[128] Hendrik Nolte and Philipp Wieder. "Realising Data-Centric Scientific Workflows with Provenance-Capturing on Data Lakes". In: *Data Intelligence* 4.2 (2022), pp. 426–438. ISSN: 2641-435X. DOI: 10.1162/dint_a_00141.

[129] Larry Lannom et al. *FDO Forum FDO Configuration Types*. 2022. DOI: 10.5281/zenodo.7825703.

[130] Christophe Blanchi et al. *Implementation of Attributes, Types, Profiles and Registries*. 2025. DOI: 10.5281/zenodo.14614908.

[131] Ivonne Anders et al. *FDO Forum FDO Requirement Specifications*. 2023. DOI: 10.5281/zenodo.7782262.

[132] Tobias Weigel et al. *RDA Research Data Collections WG Recommendations*. 2017. DOI: 10.15497/RDA00022.

[133] Thomas Jejkal et al. "Realizing FAIR Digital Objects for the German Helmholtz Association of Research Centres". In: *Research Ideas and Outcomes* 8 (2022), e94758. ISSN: 2367-7163. DOI: 10.3897/rio.8.e94758.

[134] Jan Schweikert et al. "A Photovoltaic System Model Integrating FAIR Digital Objects and Ontologies". In: *Energies* 16.3 (2023), p. 1444. ISSN: 1996-1073. DOI: 10.3390/en16031444.

[135] Markus Stocker et al. "Persistent Identification of Instruments". In: *Data Science Journal* 19.1 (2020), p. 18. ISSN: 1683-1470. DOI: 10.5334/dsj-2020-018.

[136] Ulrich Schwardmann and Tibor Kálmán. "How FDO attributes can support machine- and human-readability? - a description along three examples". In: *Research Ideas and Outcomes* 9 (2023), e108737. ISSN: 2367-7163. DOI: 10.3897/rio.9.e108737.

[137] Sharif Islam et al. "Assessing the FAIR Digital Object Framework for Global Biodiversity Research". In: *Research Ideas and Outcomes* 9 (2023). DOI: 10.3897/rio.9.e108808.

[138] Sharif Islam. "FAIR digital objects, persistent identifiers and machine actionability". In: *FAIR Connect: Empowering Data Stewardship* 1.1 (2023), pp. 29–34. DOI: 10.3233/FC-230001.

[139] Christa Deeleman-Reinhold, Wouter Addink, and Jeremy Miller. "The genera Chrysilla and Phintelloides revisited with the description of a new species (Araneae, Salticidae) using digital specimen DOIs and nanopublications". In: *Biodiversity Data Journal* 12 (2024), e129438. ISSN: 1314-2828. DOI: 10.3897/BDJ.12.e129438.

[140] Wouter Addink. "A Primer for FDO Profile Creation Based on Early Experience". In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1178.

[141] Koenraad Smedt, Dimitrios Koureas, and Peter Wittenburg. "FAIR Digital Objects for Science: From Data Pieces to Actionable Knowledge Units". In: *Publications* 8 (2020), p. 21. DOI: 10.3390/publications8020021.

[142] Stian Soiland-Reyes et al. "Packaging research artefacts with RO-Crate". In: *Data Science* 5.2 (2022), pp. 97–138. ISSN: 2451-8484. DOI: 10.3233/DS-210053.

[143] Stian Soiland-Reyes et al. "Creating lightweight FAIR Digital Objects with RO-Crate". In: *Research Ideas and Outcomes* 8 (2022), e93937. ISSN: 2367-7163. DOI: 10.3897/rio.8.e93937.

[144] Leyla Jael Castro, Stian Soiland-Reyes, and Dietrich Rebholz-Schuhmann. "RO-Crates Meets FAIR Digital Objects". In: *Proceedings of the Conference on Research Data Infrastructure* 1 (2023). ISSN: 2941-296X. DOI: 10.52825/cordi.v1i.396.

[145] Julian Lopez Gordillo and Sharif Islam. "FDO as an Interoperability Framework for the Biodiversity Digital Twin Project". In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1190.

[146] Mark Wilkinson et al. *Report on FAIR Signposting and its Uptake by the Community*. 2024. DOI: 10.5281/zenodo.10490289.

[147] Stian Soiland-Reyes et al. "Practical webby FDOs with RO-Crate and FAIR Signposting: Experiences and lessons learned". In: *Open Conference Proceedings: International FAIR Digital Objects Implementation Summit 2024*. TIB Open Publishing, 2024. URL: https://research.manchester.ac.uk/en/publications/practical-webby-fdos-with-ro-crate-and-fair-signposting-experienc (visited on 04/01/2025).

[148] Stian Soiland-Reyes et al. *BioHackEU23 report: Enabling FAIR Digital Objects with RO-Crate, Signposting and Bioschemas*. en-us. Jan. 2024. DOI: 10.37044/osf.io/gmk2h. URL: https://osf.io/gmk2h_v1 (visited on 08/15/2025).

[149] Luiz Olavo Bonino da Silva Santos et al. "Towards a Conceptual Model for the FAIR Digital Object Framework". In: *Formal Ontology in Information Systems*. IOS Press, 2023, pp. 227–241. DOI: 10.3233/FAIA231131.

[150] Barbara Magagna et al. "Nanopublications as FAIR Digital Object Implementations". In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1417.

[151] Stian Soiland-Reyes, Carole Goble, and Paul Groth. "Evaluating FAIR Digital Object and Linked Data as distributed object systems". In: *PeerJ Computer Science* 10 (2024), e1781. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.1781.

[152] *Handle.Net Registry*. Corporation for National Research Initiatives (CNRI). URL: https://www.handle.net/ (visited on 01/27/2026).

[153] Persistent Identifiers for eResearch (ePIC). *ePIC Data Type Registry*. URL: https://dtr-pit.pidconsortium.net/ (visited on 12/20/2025).

[154] Robert Tupelo-Schneck. "An Introduction to Cordra". In: *Research Ideas and Outcomes* 8 (2022). Publisher: Pensoft Publishers, e95966. ISSN: 2367-7163. DOI: 10.3897/rio.8.e95966.

[155] Andreas Pfeil and Thomas Jejkal. *Typed PID Maker*. Version 2.2.1. June 2025. URL: https://github.com/kit-data-manager/pit-service (visited on 01/09/2026).

[156] Zoe Mayer et al. *Thermal Bridges on Building Rooftops - Hyperspectral (RGB + Thermal + Height) drone images of Karlsruhe, Germany, with thermal bridge annotations*. 2022. DOI: 10.5281/zenodo.7022736.

[157] Zoe Mayer et al. "Deep learning approaches to building rooftop thermal bridge detection from aerial images". In: *Autom. Constr.* 146 (2023), p. 104690. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2022.104690.

[158] Mark D. Wilkinson et al. "Evaluating FAIR maturity through a scalable, automated, community-governed framework". In: *Scientific Data* 6.174 (2019). ISSN: 2052-4463. DOI: 10.1038/s41597-019-0184-5.

[159] Alex R Hardisty et al. "Digital Extended Specimens: Enabling an Extensible Network of Biodiversity Data Records as Integrated Digital Objects on the Internet". In: *BioScience* 72.10 (2022), pp. 978–987. ISSN: 0006-3568. DOI: 10.1093/biosci/biac060.

[160] Stian Soiland-Reyes. *FAIR Research Objects and Computational Workflows – A Linked Data Approach*. 2024. DOI: 10.5281/zenodo.10468447.

[161] George Strawn. "Open Science, Business Analytics, and FAIR Digital Objects". In: *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. ISSN: 0730-3157. 2019, pp. 658–663. DOI: 10.1109/COMPSAC.2019.10283.

[162] Larry Lannom et al. *Typing FAIR Digital Objects*. 2022. DOI: 10.5281/zenodo.7825599.

[163] Ivonne Anders et al. *FAIR Digital Object Technical Overview*. 2023. DOI: 10.5281/zenodo.7824714.

[164] Ulrich Schwardmann. "How to Design FDO Profiles and Kernel Attributes? An Investigation Along Processing, Grammars and Automata". In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1392.

[165] Christophe Blanchi, Binyam Gebre, and Peter Wittenburg. "Canonical Workflow for Machine Learning Tasks". In: *Data Intelligence* 4.2 (2022), pp. 173–185. ISSN: 2641-435X. DOI: 10.1162/dint_a_00124.

[166] Larry Lannom, Dimitris Koureas, and Alex R. Hardisty. "FAIR Data and Services in Biodiversity Science and Geoscience". In: *Data Intelligence* 2.1-2 (2020), pp. 122–130. ISSN: 2641-435X. DOI: 10.1162/dint_a_00034.

[167] Zachary Trautt. "FDOs and Materials Science and Engineering". In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1096.

[168] Christophe Blanchi. "FDO Access Control Using FDO Operations". In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1193.

[169] Henrik tom Wörden et al. “Meet the FDO Manager: Reference Implementation of FDO Operations (Create, Retrieve, Update, Delete) Based on the Digital Object Interface Protocol (DOIP).” In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1197.

[170] Sven Bingert et al. “FDO Testbed”. In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1195.

[171] Oussama Zoubia et al. “FDO Manager: Minimum Viable FAIR Digital Object Implementation”. In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1421.

[172] Claus Weiland et al. “Employing “Webby” FAIR Digital Objects to Support Large-Scale Ecosystem Monitoring and Mitigation of Biodiversity Loss in the Anthropocene”. In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1181.

[173] Benedikt Heinrichs, Sirieam Hunke, and Marius Politze. “Combining FAIR Digital Object Implementation Concepts in a Real-Life Application”. In: *Open Conference Proceedings* 5 (2024). ISSN: 2749-5841. DOI: 10.52825/ocp.v5i.1173.

[174] Daniel L. Moody. “Metrics for Evaluating the Quality of Entity Relationship Models”. In: *Conceptual Modeling – ER ’98*. Ed. by Tok-Wang Ling, Sudha Ram, and Mong Li Lee. Berlin, Heidelberg: Springer, 1998, pp. 211–225. ISBN: 978-3-540-49524-6. DOI: 10.1007/978-3-540-49524-6_18.

[175] Mark D. Wilkinson et al. “Interoperability and FAIRness through a novel combination of Web technologies”. In: *PeerJ Computer Science* 3 (2017), e110. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.110.

[176] Randi Vita et al. “FAIR principles and the IEDB: short-term improvements and a long-term vision of OBO-foundry mediated machine-actionable interoperability”. In: *Database* 2018 (2018). ISSN: 1758-0463. DOI: 10.1093/database/bax105.

[177] Nikolay Skvortsov. “The Principles of Data Reuse in Research Infrastructures”. In: *Proceedings of the International Conference "Common Digital Space of Scientific Knowledge: Problems & Solutions" (CDSSK 2020)* (Nov. 10–12, 2020). Ed. by Alexander Elizarov, Nikolay Kalenov, and Alexander Sotnikov. Vol. 2990. CEUR Workshop Proceedings. Moscow, Russia, 2021, pp. 62–74. URL: https://ceur-ws.org/Vol-2990/rpaper6.pdf (visited on 12/29/2025).

[178] C. Fan et al. “A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data”. In: *Frontiers in Energy Research* 9 (2021). DOI: 10.3389/fenrg.2021.652801.

[179] David Haynes. “Metadata standards”. In: *Metadata for Information Management and Retrieval: Understanding metadata and its use*. Facet, 2018, pp. 49–74.

[180] Jonathan Lathem, Karthik Gomadam, and Amit Sheth. “SA-REST and (S)mashups: Adding Semantics to RESTful Services”. In: *International Conference on Semantic Computing (ICSC 2007)* (2007), pp. 469–476. DOI: 10.1109/ICSC.2007.94.

[181] Franck Michel et al. “Integration of Web APIs and Linked Data Using SPARQL Micro-Services—Application to Biodiversity Use Cases”. In: *Information* 9.12 (2018). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 310. ISSN: 2078-2489. DOI: 10.3390/info9120310.

[182] Mark D. Wilkinson, Benjamin Vandervalk, and Luke McCarthy. “The Semantic Automated Discovery and Integration (SADI) Web service Design-Pattern, API and Reference Implementation”. In: *Journal of Biomedical Semantics* 2.1 (2011), p. 8. ISSN: 2041-1480. DOI: 10.1186/2041-1480-2-8.

[183] Ulrich Schwardmann. “Digital Objects – FAIR Digital Objects: Which Services Are Required?” In: *Data Science Journal* 19 (2020), p. 15. DOI: 10.5334/dsj-2020-015.

[184] Yuri Demchenko, Shashank Shrestha, and Panayiotis Andreou. *D7.2. SLICES Interoperability Framework and Integration with EOSC*. Deliverable D7.2. Horizon Europe Grant Agreement No. 101079774. SLICES-PP project consortium, 2024. DOI: 10.3030/101079774.

[185] Wenguang Wang, Andreas Tolk, and Weiping Wang. *The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S*. arXiv:0908.0191 [cs]. 2009. DOI: 10.48550/arXiv.0908.0191.

[186] C. Shanthi, M. S. Josephine, and V. JeyabalaRaja. “Analysis of interoperability between mobile apps cross-platforms development using LCIM Model”. In: *Journal of Chemical and Pharmaceutical Sciences* 9.4 (2016), pp. 2149–2152. ISSN: 0974-2115.

[187] Andreas Tolk, Saikou Diallo, and Charles Turnitsa. “Applying the Levels of Conceptual Interoperability Model in Support of Integratability, Interoperability, and Composability for System-of-Systems Engineering”. In: *Journal of Systems, Cybernetics, and Informatics* 5.5 (2007). URL: https://digitalcommons.odu.edu/msve_fac_pubs/27 (visited on 11/07/2025).

[188] Erik Wassermann and Alexander Fay. “Interoperability rules for heterogenous multi-agent systems: Levels of conceptual interoperability model applied for multi-agent systems”. In: *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. ISSN: 2378-363X. 2017, pp. 89–95. DOI: 10.1109/INDIN.2017.8104752.

[189] Abhishek Hazra et al. “A Comprehensive Survey on Interoperability for IIoT: Taxonomy, Standards, and Future Directions”. In: *ACM Computing Surveys* 55.1 (2023), pp. 1–35. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3485130.

[190] Christoph Quix, Rihan Hai, and Ivan Vatov. “GEMMS: A Generic and Extensible Metadata Management System for Data Lakes”. In: *Proceedings of the CAiSE’16 Forum, at the 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016)* (Ljubljana, Slovenia, June 13–17, 2016). Ed. by Sergio España, Mirjana Ivanović, and Miloš Savić. Vol. 1612. CEUR Workshop Proceedings. CEUR, 2016, pp. 129–136. URL: https://ceur-ws.org/Vol-1612/paper17.pdf (visited on 12/19/2025).

[191] Étienne Scholly et al. “Coining goldMEDAL: A New Contribution to Data Lake Generic Metadata Modeling”. In: *Proceedings of the 23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP 2021), co-located with EDBT/ICDT 2021* (Mar. 23, 2021). Ed. by Kostas Stefanidis and Patrick Marcel. Vol. 2840. CEUR Workshop Proceedings. Nicosia, Cyprus: CEUR, 2021, pp. 31–40. URL: https://ceur-ws.org/Vol-2840/paper5.pdf (visited on 12/19/2025).

[192] Pranav Subramaniam et al. *Comprehensive and Comprehensible Data Catalogs: The What, Who, Where, When, Why, and How of Metadata Management*. arXiv:2103.07532 [cs]. 2023. DOI: 10.48550/arXiv.2103.07532.

[193] Ashwini Gajare. “Implementing Data Lineage and Metadata Tracking with Open Lineage and Amundsen”. In: *International Journal of Multidisciplinary Research in Science, Engineering and Technology* 6.12 (2023). URL: https://philarchive.org/rec/ASHIDL (visited on 11/10/2025).

[194] Philipp Wieder and Hendrik Nolte. “Toward data lakes as central building blocks for data management and analysis”. In: *Frontiers in Big Data* 5 (2022). ISSN: 2624-909X. DOI: 10.3389/fdata.2022.945720.

[195] Matthias Liffers et al. *DataCite Metadata Schema Documentation for the Publication and Citation of Research Data and Other Research Outputs*. Version 4.6. 2024. DOI: 10.14454/mzv1-5b55.

[196] Dublin Core Metadata Initiative. *Dublin Core™ Metadata Element Set, Version 1.1: Reference Description*. URL: https://www.dublincore.org/documents/dces/ (visited on 12/20/2025).

[197] Riccardo Albertoni et al. *Data Catalog Vocabulary (DCAT) – Version 3*. W3C Recommendation. World Wide Web Consortium (W3C), 2024. URL: https://www.w3.org/TR/vocab-dcat-3/ (visited on 12/20/2025).

[198] Schema.org Community. *Schema.org*. URL: https://schema.org/ (visited on 12/20/2025).

[199] Riccardo Albertoni et al. *JSON-LD 1.1*. W3C Recommendation. World Wide Web Consortium (W3C), 2020. URL: https://www.w3.org/TR/json-ld11/ (visited on 12/20/2025).

[200] Zenodo. *FAIR Principles (Zenodo): Metadata formats and policies*. URL: https://about.zenodo.org/principles/ (visited on 12/20/2025).

[201] Felicitas Löffler et al. “Dataset search in biodiversity research: Do metadata in data repositories reflect scholarly information needs?” In: *PloS One* 16.3 (2021), e0246099. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0246099.

[202] Jana Martínková et al. “Moving towards FAIR mappings and crosswalks.” In: FAIR principles for Ontologies and Metadata in Knowledge Management, FOAM 2024. 2024. URL: https://research.manchester.ac.uk/en/publications/moving-towards-fair-mappings-and-crosswalks (visited on 01/27/2026).

[203] Sergey Stupnikov and Leonid Kalinichenko. "Extensible Unifying Data Model Design for Data Integration in FAIR Data Infrastructures". In: *Data Analytics and Management in Data Intensive Domains*. Ed. by Yannis Manolopoulos and Sergey Stupnikov. Cham: Springer International Publishing, 2019, pp. 17–36. ISBN: 978-3-030-23584-0. DOI: 10.1007/978-3-030-23584-0_2.

[204] Lois Mai Chan and Marcia Lei Zeng. "Metadata Interoperability and Standardization – A Study of Methodology Part I: Achieving Interoperability at the Schema Level". In: *D-Lib Magazine* 12.6 (2006). ISSN: 1082-9873. DOI: 10.1045/june2006-chan.

[205] Karsten Menzel et al. "Linked Data and Ontologies for Semantic Interoperability". In: *Innovative Tools and Methods Using BIM for an Efficient Renovation in Buildings*. Ed. by Bruno Daniotti et al. Cham: Springer International Publishing, 2022, pp. 17–28. ISBN: 978-3-031-04670-4. DOI: 10.1007/978-3-031-04670-4_2.

[206] David Martin et al. "Bringing Semantics to Web Services with OWL-S". In: *World Wide Web* 10.3 (2007), pp. 243–277. ISSN: 1573-1413. DOI: 10.1007/s11280-007-0033-x.

[207] Axel Polleres and Dieter Fensel. "Towards Intelligent web Services: Web Service Modeling Ontology (WSMO)". In: Springer, 2005. DOI: 10.13025/20960.

[208] Elena Paslaru Bontas Simperl, Christoph Tempich, and Malgorzata Mochol. "Cost Estimation for Ontology Development: Applying the ONTOCOM Model". In: *Technologies for Business Information Systems*. Ed. by Witold Abramowicz and Heinrich C. Mayr. Dordrecht: Springer Netherlands, 2007, pp. 327–339. ISBN: 978-1-4020-5634-5. DOI: 10.1007/1-4020-5634-6_28.

[209] Olivier Corby, Catherine Faron-Zucker, and Fabien Gandon. "LDScript: A Linked Data Script Language". In: *The Semantic Web – ISWC 2017*. Ed. by Claudia d'Amato et al. Cham: Springer International Publishing, 2017, pp. 208–224. ISBN: 978-3-319-68288-4. DOI: 10.1007/978-3-319-68288-4_13.

[210] Eyal Oren, Benjamin Heitmann, and Stefan Decker. "ActiveRDF: Embedding Semantic Web data into object-oriented languages". In: *Journal of Web Semantics*. World Wide Web Conference 2007Semantic Web Track 6.3 (2008), pp. 191–202. ISSN: 1570-8268. DOI: 10.1016/j.websem.2008.04.003.

[211] Wardani Muhamad, Suhardi, and Yoanes Bandung. "Transforming OpenAPI Specification 3.0 documents into RDF-based semantic web services". In: *Journal of Big Data* 9.1 (2022), p. 55. ISSN: 2196-1115. DOI: 10.1186/s40537-022-00600-8.

[212] Markus Lanthaler and Christian Gütl. "Hydra: A Vocabulary for Hypermedia-Driven Web APIs". In: *Proceedings of the WWW2013 Workshop on Linked Data on the Web (LDOW2013)*. Linked Data on the Web (LDOW2013) (Rio de Janeiro, Brazil). Ed. by Christian Bizer et al. Vol. 996. CEUR Workshop Proceedings. CEUR-WS.org, May 14, 2013. URL: https://ceur-ws.org/Vol-996/papers/ldow2013-paper-03.pdf (visited on 01/27/2026).

[213] LuizOlavoBonino daSilvaSantos et al. "FAIR Data Point: A FAIR-Oriented Approach for Metadata Publication". In: *Data Intelligence* 5.1 (2025), pp. 163–183. ISSN: 2096-7004. DOI: 10.1162/dint_a_00160.

[214] Nishad Thalhath, Mitsuharu Nagamori, and Tetsuo Sakaguchi. "Metadata application profile as a mechanism for semantic interoperability in FAIR and open data publishing". In: *Data and Information Management* 9.1 (2025), p. 100068. ISSN: 2543-9251. DOI: 10.1016/j.dim.2024.100068.

[215] OpenAPI Initiative. *OpenAPI*. URL: https://www.openapis.org/ (visited on 12/20/2025).

[216] Balaji Varanasi and Maxim Bartkov. "HATEOAS". In: *Spring REST: Building Java Microservices and Cloud Applications*. Ed. by Balaji Varanasi and Maxim Bartkov. Berkeley, CA: Apress, 2022, pp. 223–236. ISBN: 978-1-4842-7477-4. DOI: 10.1007/978-1-4842-7477-4_10.

[217] Peter Amstutz et al. *Common Workflow Language, v2.0. Specification*. Common Workflow Language working group, 2016. DOI: 10.6084/m9.figshare.3115156.v2.

[218] Peter Wittenburg et al. "Canonical Workflows to Make Data FAIR". In: *Data Intelligence* 4.2 (2024), pp. 286–305. ISSN: 2096-7004. DOI: 10.1162/dint_a_00132.

[219] Dirk Betz et al. "Canonical Workflow for Experimental Research". In: *Data Intelligence* 4.2 (2022), pp. 155–172. ISSN: 2641-435X. DOI: 10.1162/dint_a_00123.

[220] Amirpasha Mozaffari et al. "HPC-oriented Canonical Workflows for Machine Learning Applications in Climate and Weather Prediction". In: *Data Intelligence* 4.2 (2024), pp. 271–285. ISSN: 2096-7004. DOI: 10.1162/dint_a_00131.

[221] Nikolay A. Skvortsov and Sergey A. Stupnikov. "A Semantic Approach to Workflow Management and Reuse for Research Problem Solving". In: *Data Intelligence* 4.2 (Apr. 2022), pp. 439–454. ISSN: 2641-435X. DOI: 10.1162/dint_a_00142. URL: https://doi.org/10.1162/dint_a_00142 (visited on 12/28/2025).

[222] Persistent Identifiers for eResearch (ePIC). *ePIC Data Type Registry (testing)*. URL: https://dtr-test.pidconsortium.net/ (visited on 12/20/2025).

[223] Persistent Identifiers for eResearch (ePIC). *Typeregistry Lab*. URL: https://typeregistry.lab.pidconsortium.net/ (visited on 12/20/2025).

[224] Eddy Truyen et al. "A Generalization and Solution to the Common Ancestor Dilemma Problem in Delegation-Based Object Systems". In: *Proceedings of the 2004 Dynamic Aspects Workshop (DAW04)* (Lancaster, UK, Mar. 23, 2004). Ed. by Robert E. Filman et al. RIACS Technical Report 04.01. 2004, pp. 103–119. URL: https://lirias.kuleuven.be/1653772 (visited on 02/02/2026).

[225] Bartosz Blachut. "Extending data repositories with a novel FAIR digital objects framework". Diploma Thesis. Technische Universität Wien, 2025, p. 105. DOI: 10.34726/hss.2025.129384.

[226] Maximilian Inckmann. "Entwicklung einer Registry für typgebundene Operationen auf FAIR Digital Objects". German. Abschlussarbeit - Bachelor. Karlsruher Institut für Technologie (KIT), 2024. DOI: 10.5445/IR/1000174814.

[227] David Montag. *Understanding Neo4j Scalability*. White paper. Neo Technology, Jan. 2013, p. 8. URL: https://go.neo4j.com/rs/710-RRC-335/images/Understanding%20Neo4j%20Scalability%282%29.pdf (visited on 09/27/2025).

[228] Nicolas Blumenröhr and Felix Kraus. “Leveraging Large Language Models for Processing and Evaluating FAIR Digital Objects”. In: *Proceedings of the 2nd International Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2025) - Co-located with the 22nd Extended Semantic Web Conference (ESWC 2025)*. Ed. by Georg Rehm et al. Vol. 3977. CEUR Workshop Proceedings. Portorož, Slovenia: CEUR, 2025. DOI: 10.5445/IR/1000183869.

A. Appendix

A.1. Background Chapter Appendix

Table A.1.: Overview of FAIR principles criteria as outlined in [1].

The FAIR Guiding Principles	
To be Findable:	F1. (meta)data are assigned a globally unique and persistent identifier F2. data are described with rich metadata (defined by R1 below) F3. metadata clearly and explicitly include the identifier of the data it describes F4. (meta)data are registered or indexed in a searchable resource
To be Accessible:	A1. (meta)data are retrievable by their 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 data are no longer available
To be Interoperable:	I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation I2. (meta)data use vocabularies that follow FAIR principles I3. (meta)data include qualified references to other (meta)data
To be Reusable:	R1. meta(data) are richly described with a plurality of accurate and relevant attributes R1.1. (meta)data are released with a clear and accessible data usage license R1.2. (meta)data are associated with detailed provenance R1.3. (meta)data meet domain-relevant community standards

Table A.2.: Overview of (modified) FAIR4RS Principles criteria as outlines in [57].

The FAIR4RS Principles	
To be Findable:	F1.1 Components of the software representing levels of granularity are assigned distinct identifiers. F1.2 Different versions of the software are assigned distinct identifiers.
To be Interoperable:	I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards. I2. Software includes qualified references to other objects.

Table A.3.: Overview of updated FAIR principles criteria as outlines in [56]

The Updated FAIR Guiding Principles after Vogt et al.	
To be Accessible:	A1.3 the protocol is compliant with existing data protection regulations (e.g., General Data Protection Regulation, GDPR)
To be Interoperable:	I2.1 vocabularies used in (meta)data ideally include multilingual labels and specify all relevant synonyms for their terms I2.2 terms with the same intensional meaning and/or the same extension are ideally mapped across all relevant vocabularies through intensional or extensional entity mappings I2.3 terms provide human-readable specifications of their intensional meaning and, where applicable, human-readable recognition criteria for correctly applying them I4 (meta)data schemata are used that support propositional interoperability I4.1 ideally, the same (meta)data schema is used for the same type of statement or collection of statement types, and the schema is referenced with its identifier in the statement's metadata I4.2 ideally, (meta)data schemata for the same type of statement are aligned and mapped across all relevant schemata (i.e., schema crosswalks) I4.3 (meta)data use a formalism to clearly distinguish between lexical, assertonal, contingent, prototypical, and universal statements I4.4 (meta)data specify the logical framework that has been used for their modeling (e.g., Description Logics using OWL or First Order Logic using Common Logic Interchange Framework)
To be Reusable:	R1.4 metadata indicate the certainty level of the truthfulness of the semantic content of their corresponding data

A.2. Chapter 3 Appendix

A.2.1. FDO Graph for Example Data

Association of nodes in FDO Graph with corresponding Handle PIDs, resolvable via https://hdl.handle.net/:

$$
\begin{aligned}
f_1 &= \texttt{21.11152/6ea60288-d895-414e-80c0-26c9fdd662b2}\\
f_2 &= \texttt{21.11152/58d43ddc-5e29-4980-8675-ae579b50a1e2}\\
f_3 &= \texttt{21.11152/6858a0b5-cc60-40e9-afef-8c2dd8b35e8e}\\
f_4 &= \texttt{21.11152/e670f510-7e00-4d3a-9b90-3bac7a7c069e}\\
f_5 &= \texttt{21.11152/3ab9f444-05f6-445e-a691-62fae4021bea}\\
f_6 &= \texttt{21.11152/365fd8cf-8e86-41b8-9d0e-b816fdd01d29}\\
f_7 &= \texttt{21.11152/041a6111-644a-4617-afb3-3c421a88e8e3}\\
f_8 &= \texttt{21.11152/f48bf4e7-3879-4216-8f64-45a060b8f658}\\
f_9 &= \texttt{21.11152/7b58b3b5-75eb-4417-ac4d-abe025e159f6}\\
f_{10} &= \texttt{21.11152/ba370aa3-6422-428c-9ff7-c2ef429df603}\\
f_{11} &= \texttt{21.11152/09cb76fc-b8cb-4116-a22a-68c5bdfa77b0}\\
f_{12} &= \texttt{21.11152/24a55398-b96b-43dd-b0fb-cd8ce302c7ce}\\
f_{13} &= \texttt{21.11152/721234ac-4b5a-4d02-9944-82a08ef2db35}\\
f_{14} &= \texttt{21.11152/ebaeb5bc-0514-47c9-bcd2-98f0253843d8}\\
f_{15} &= \texttt{21.11152/9854677c-77c5-4a0b-916b-57dd9ec20198}\\
f_{16} &= \texttt{21.11152/cfd0fc0e-f5ea-464e-a57f-28e882924860}\\
f_{17} &= \texttt{21.11152/976fcf28-f924-4a21-b53d-5d054ad8198d}\\
f_{18} &= \texttt{21.11152/37833c54-1d36-42e4-858d-831447122863}\\
f_{19} &= \texttt{21.11152/a3f19b32-4550-40bb-9f69-b8ffd4f6d0ea}
\end{aligned}
$$

FDO Graph with all PID triples in $\mathcal{T}$:

$\tilde{F} = \{\tilde{f_1}, \tilde{f_2}, \tilde{f_3}, \tilde{f_4}, \tilde{f_5}, \tilde{f_6}, \tilde{f_7}, \tilde{f_8}, \tilde{f_9}, \tilde{f_{10}}, \tilde{f_{11}}, \tilde{f_{12}}, \tilde{f_{13}}, \tilde{f_{14}}, \tilde{f_{15}}, \tilde{f_{16}}, \tilde{f_{17}}, \tilde{f_{18}}\}$,

$\tilde{K} = \{a, b \mid a \mathrel{\hat{=}} hasMetadataPIT,\ b \mathrel{\hat{=}} isMetadataForPIT\}$,

$\mathcal{T} = \{(\tilde{f_1}, b, \tilde{f_3}),\ (\tilde{f_1}, b, \tilde{f_4}),\ (\tilde{f_1}, b, \tilde{f_5}),\ (\tilde{f_1}, b, \tilde{f_6}),\ (\tilde{f_1}, b, \tilde{f_7}),$
$(\tilde{f_2}, b, \tilde{f_8}),\ (\tilde{f_9}, b, \tilde{f_3}),\ (\tilde{f_9}, b, \tilde{f_4}),\ (\tilde{f_9}, b, \tilde{f_5}),\ (\tilde{f_9}, b, \tilde{f_6}),\ (\tilde{f_9}, b, \tilde{f_7}),\ (\tilde{f_9}, b, \tilde{f_8}),\ (\tilde{f_{10}}, b, \tilde{f_{11}}),$
$(\tilde{f_{10}}, b, \tilde{f_{12}}),\ (\tilde{f_{10}}, b, \tilde{f_{13}}),\ (\tilde{f_{10}}, b, \tilde{f_{14}}),\ (\tilde{f_{10}}, b, \tilde{f_{15}}),\ (\tilde{f_{10}}, b, \tilde{f_{16}}),\ (\tilde{f_{10}}, b, \tilde{f_{17}}),; (\tilde{f_{10}}, b, \tilde{f_{18}}),$
$(\tilde{f_{11}}, b, \tilde{f_3}),\ (\tilde{f_{13}}, b, \tilde{f_5}),\ (\tilde{f_{14}}, b, \tilde{f_6}),\ (\tilde{f_{15}}, b, \tilde{f_7}),\ (\tilde{f_{16}}, b, \tilde{f_8}),\ (\tilde{f_{12}}, b, \tilde{f_4})$
$(\tilde{f_3}, a, \tilde{f_1}),\ (\tilde{f_3}, a, \tilde{f_{11}}),\ (\tilde{f_3}, a, \tilde{f_9}),\ (\tilde{f_4}, a, \tilde{f_1}),\ (\tilde{f_4}, a, \tilde{f_{12}}),\ (\tilde{f_4}, a, \tilde{f_9}),\ (\tilde{f_5}, a, \tilde{f_1}),\ (\tilde{f_5}, a, \tilde{f_{13}}),$
$(\tilde{f_5}, a, \tilde{f_9}),\ (\tilde{f_6}, a, \tilde{f_1}),\ (\tilde{f_6}, a, \tilde{f_{14}}),\ (\tilde{f_6}, a, \tilde{f_9}),\ (\tilde{f_7}, a, \tilde{f_1}),\ (\tilde{f_7}, a, \tilde{f_{15}}),\ (\tilde{f_7}, a, \tilde{f_9}),\ (\tilde{f_8}, a, \tilde{f_2}),$
$(\tilde{f_8}, a, \tilde{f_{16}}),\ (\tilde{f_8}, a, \tilde{f_9}),\ (\tilde{f_{11}}, a, \tilde{f_{10}}),\ (\tilde{f_{12}}, a, \tilde{f_{10}}),\ (\tilde{f_{13}}, a, \tilde{f_{10}}),\ (\tilde{f_{14}}, a, \tilde{f_{10}}),\ (\tilde{f_{15}}, a, \tilde{f_{10}}),$
$(\tilde{f_{16}}, a, \tilde{f_{10}}),\ (\tilde{f_{17}}, a, \tilde{f_{10}}),\ (\tilde{f_{18}}, a, \tilde{f_{10}})\}$

A.2.2. Original Handle Record of Example Data FDO

Handle.Net®

Handle Values for: 21.11152/6858a0b5-cc60-40e9-afef-8c2dd8b35e8e

Index	Type	Timestamp	Data
1	21.T11148/076759916209e5d62bd5	2022-08-25 09:43:43Z	21.T11148/b9b76f887845e32d29f7
2	21.T11148/397d831aa3a9d18eb52c	2022-08-25 09:43:43Z	2022-08-19T00:00:00+00:00
3	21.T11148/82e2503c49209e987740	2022-08-25 09:43:43Z	{ "md5sum": "716acce83a51ad2fc958ab3ce0026f71" }
4	21.T11148/aafd5fb4c7222e2d950a	2022-08-25 09:43:43Z	2022-05-30T00:00:00+00:00
5	21.T11148/b8457812905b83046284	2022-08-25 09:43:43Z	https://zenodo.org/record/6517768/files/Flug1_100.tar.zst?download=1
6	21.T11148/c692273deb2772da307f	2022-08-25 09:43:43Z	1.0.0
7	21.T11148/1c699a5d1b4ad3ba4956	2022-08-25 09:43:43Z	21.11152/1faf85ec-c661-4c0d-a47c-8b07c9e3835c
8	21.T11148/2f314c8fe5fb6a0063a8	2022-08-25 09:43:43Z	https://creativecommons.org/licenses/by/4.0/
9	21.T11148/d0773859091aeb451528	2022-08-25 09:43:43Z	21.11152/6ea60288-d895-414e-80c0-26c9fdd662b2
10	21.T11148/d0773859091aeb451528	2022-08-25 09:43:43Z	21.11152/09cb76fc-b8cb-4116-a22a-68c5bdfa77b0
11	21.T11148/d0773859091aeb451528	2022-08-25 09:43:43Z	21.11152/7b58b3b5-75eb-4417-ac4d-abe025e159f6
12	21.T11148/b415e16fbe4ca40f2270	2022-08-25 09:43:43Z	http://vocabularies.unesco.org/thesaurus/concept1557
13	21.T11148/1a73af9e7ae00182733b	2022-08-25 09:43:43Z	https://orcid.org/0000-0002-9082-9095
14	21.T11148/1a73af9e7ae00182733b	2022-08-25 09:43:43Z	https://orcid.org/0000-0002-8517-2359
15	21.T11148/1a73af9e7ae00182733b	2022-08-25 09:43:43Z	https://orcid.org/0000-0002-9822-244X
16	21.T11148/1a73af9e7ae00182733b	2022-08-25 09:43:43Z	https://orcid.org/0000-0002-2233-1041
17	21.T11148/1a73af9e7ae00182733b	2022-08-25 09:43:43Z	https://orcid.org/0000-0001-9930-5354
18	21.T11148/1a73af9e7ae00182733b	2022-08-25 09:43:43Z	https://orcid.org/0000-0002-8059-4149
100	HS_ADMIN	2022-08-25 09:43:43Z	handle=21.11152/USER01; index=300; [create hdl,delete hdl,create derived prefix,delete derived prefix,read val,modify val,del val,add val,modify admin,del admin,add admin,list]

Figure A.1.: The original Handle Record of FDO f_3 in Section 3.4.2.2.

A.2.3. Original Handle Records Used in the Comparative Analysis

Handle.Net®

Handle Values for: 21.T11998/0000-001A-3905-1

Index	Type	Timestamp	Data
1	URL	2021-04-14 14:02:37Z	https://linkedsystems.uk/system/instance/TOOL0022_2490/current/
2	21.T11148/8eb858ee0b12e8e463a5	2021-04-15 16:41:02Z	{"identifierValue":"http://hdl.handle.net/21.T11998/0000-001A-3905-1","identifierType":"Handle"}
3	21.T11148/9a15a4735d4bda329d80	2021-04-14 14:02:37Z	https://linkedsystems.uk/system/instance/TOOL0022_2490/current/
4	21.T11148/709a23220f2c3d64d1e1	2021-04-14 14:02:37Z	Sea-Bird SBE 37-IM MicroCAT C-T Sensor
5	21.T11148/4eaec4bc0f1df68ab2a7	2021-04-14 14:11:28Z	[{"owner": {"ownerName":"National Oceanography Centre","ownerContact":"louise.darroch@bodc.ac.uk","ownerIdentifier": {"ownerIdentifierValue":"http://vocab.nerc.ac.uk/collection/B75/current/ORG00009/","ownerIdentifierType":"URL"}}}]
6	21.T11148/1f3e82ddf0697a497432	2021-04-14 14:05:56Z	[{"manufacturer":{"manufacturerName":"Sea-Bird Scientific","manufacturerIdentifier": {"manufacturerIdentifierValue":"http://vocab.nerc.ac.uk/collection/L35/current/MAN0013/","manufacturerIdentifierType":"URL"}}}]
7	21.T11148/c1a0ec5ad347427f25d6	2021-04-14 14:02:37Z	[{"modelName":"Sea-Bird SBE 37 MicroCat IM-CT with optional pressure (submersible) CTD sensor series","modelIdentifier": {"modelIdentifierValue":"http://vocab.nerc.ac.uk/collection/L22/current/TOOL0022/","modelIdentifierType":"URL"}}]
8	21.T11148/55f8ebc805e65b5b71dd	2021-04-14 14:02:37Z	A high accuracy conductivity and temperature recorder with an optional pressure sensor designed for deployment on moorings. The IM model has an inductive modem for real-time data transmission plus internal flash memory data storage.
9	21.T11148/f76ad9d0324302fc47dd	2021-04-14 14:02:37Z	["http://vocab.nerc.ac.uk/collection/L05/current/134/","http://vocab.nerc.ac.uk/collection/L05/current/350/"]
10	21.T11148/72928b84e060d491ee41	2021-04-14 14:05:56Z	[{"measuredVariable":{"variableMeasured":"http://vocab.nerc.ac.uk/collection/P01/current/CNDCPR01/"}}, {"measuredVariable":{"variableMeasured":"http://vocab.nerc.ac.uk/collection/P01/current/PSALPR01/"}}, {"measuredVariable":{"variableMeasured":"http://vocab.nerc.ac.uk/collection/P01/current/TEMPPR01/"}}, {"measuredVariable":{"variableMeasured":"http://vocab.nerc.ac.uk/collection/P01/current/PREXMCAT/"}}]
11	21.T11148/22c62082a4d2d9ae2602	2021-04-14 14:05:56Z	[{"date":{"dateValue":"1999-11-01","dateType":"Commissioned"}}]
12	21.T11148/eb3c713572f681e6c4c3	2021-04-14 14:05:56Z	[{"alternateIdentifier":{"alternateIdentifierValue":"2490","alternateIdentifierType":"serialNumber"}}]
13	21.T11148/178fb558abc755ca7046	2021-04-14 14:05:56Z	[{"relatedIdentifier": {"relatedIdentifierValue":"https://www.bodc.ac.uk/data/documents/nodb/pdf/37imbrochurejul08.pdf","relatedIdentifierType":"URL","relationType":"IsDescribedBy "}}]
14	0.TYPE/0.PROFILE	2023-03-09 11:34:47Z	hdl:21.T11148/17ce618137e697852ea6
100	HS_ADMIN	2021-04-14 14:02:37Z	handle=21.T11998/USER22; index=300; [create hdl,delete hdl,read val,modify val,del val,add val,modify admin,del admin,add admin]

Figure A.2.: The original PIDINST Handle Record.

Handle.Net®

Handle Values for: 21.11113/0000-000B-CA4C-D

Index	Type	Timestamp	Data
1	CREATOR	2017-12-07 20:59:10Z	PID Service pid-webapp-4.22.0.201711102014
2	ADM_MD	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D/adm
3	FILESIZE	2017-12-07 20:59:19Z	4802
4	RESPONSIBLE	2017-12-07 20:59:19Z	BeataMache@dariah.eu
5	CHECKSUM	2017-12-07 20:59:19Z	md5:d53305cfd84972afec2393bc9328c8b5
6	BAG	2017-12-07 20:59:19Z	https://cdstar.de.dariah.eu/public/EAEA0-E069-7925-F28A-0
7	PUBDATE	2017-12-07 20:59:19Z	2017-12-07 21:59:18 +0100
8	PROV_MD	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D/prov
9	URL	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D
10	DATA	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D/data
11	LANDING	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D/landing
12	SOURCE	2017-12-07 20:59:19Z	https://cdstar.de.dariah.eu/dariah/EAEA0-FA65-59AB-6BD6-0
13	INDEX	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D/index
14	METADATA	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D/metadata
15	TECH_MD	2017-12-07 20:59:19Z	https://repository.de.dariah.eu/1.0/dhcrud/21.11113/0000-000B-CA4C-D/tech
16	DOI	2017-12-07 20:59:19Z	http://dx.doi.org/10.20375/0000-000B-CA4C-D
17	INST	2017-12-07 20:59:19Z	2000
18	PUBLISHED	2017-12-07 20:59:40Z	true
100	HS_ADMIN	2017-12-07 20:59:10Z	handle=21.11113/USER02; index=1; [create hdl,delete hdl,read val,modify val,del val,add val,modify admin,del admin,add admin]

Figure A.3.: The original DARIAH Handle Record.

Handle.Net®

Handle Values for: 10.3535/G0G-G7D-N5J

Index	Type	Timestamp	Data
1	fdoProfile	2024-12-03 16:54:23Z	https://doi.org/21.T11148/d8de0819e144e4096645
2	fdoRecordLicenseId	2024-12-03 16:54:23Z	https://spdx.org/licenses/CC0-1.0.json
3	fdoRecordLicenseName	2024-12-03 16:54:23Z	CC0 1.0 Universal
4	digitalObjectType	2024-12-03 16:54:23Z	https://doi.org/21.T11148/d8de0819e144e4096645
5	digitalObjectName	2024-12-03 16:54:23Z	DigitalSpecimen
6	pid	2024-12-03 16:54:23Z	https://doi.org/10.3535/G0G-G7D-N5J
7	pidIssuer	2024-12-03 16:54:23Z	https://ror.org/04wxnsj81
8	pidIssuerName	2024-12-03 16:54:23Z	DataCite
9	pidRecordIssueDate	2024-12-03 16:54:23Z	2024-12-03T16:54:23.000Z
10	pidRecordIssueNumber	2025-07-02 11:38:14Z	2
11	pidStatus	2025-07-02 11:38:14Z	ACTIVE
40	issuedForAgent	2024-12-03 16:54:23Z	https://ror.org/04wxnsj81
41	issuedForAgentName	2024-12-03 16:54:23Z	Distributed System of Scientific Collections
42	referentName	2025-07-02 11:38:14Z	Chrysilla lauta Thorell, 1887
200	specimenHost	2025-07-02 11:38:14Z	https://ror.org/02aswte26
201	specimenHostName	2025-07-02 11:38:14Z	Manchester University
202	normalisedPrimarySpecimenObjectId	2025-07-02 11:38:14Z	CM 15726:MANUAL_SOURCE_SYSTEM
203	otherSpecimenIds	2025-07-02 11:38:14Z	[{"identifierValue":"CM 15726","identifierType":"physical specimen identifier","resolvable":false},{"identifierValue":"CM 15726","identifierType":"Locally unique identifier","resolvable":false},{"identifierValue":"MMUE G7572.5441","identifierType":"Locally unique identifier","resolvable":false}]
204	topicOrigin	2025-07-02 11:38:14Z	Natural
205	topicDomain	2025-07-02 11:38:14Z	Life
206	topicDiscipline	2025-07-02 11:38:14Z	Zoology
207	topicCategory	2025-07-02 11:38:14Z	
208	livingOrPreserved	2025-07-02 11:38:14Z	Preserved
209	materialSampleType	2025-07-02 11:38:14Z	
210	markedAsType	2025-07-02 11:38:14Z	false
211	catalogNumber	2025-07-02 11:38:14Z	
100	HS_ADMIN	2024-12-03 16:54:23Z	handle=0.NA/10.3535; index=200; [create hdl,delete hdl,read val,modify val,del val,add val,modify admin,del admin,add admin]
101	10320/loc	2025-07-02 11:38:14Z	<locations><location href="https://disscover.dissco.eu/ds/10.3535/G0G-G7D-N5J" id="0" view="HTML" weight="1"/> <location href="https://api.dissco.eu/digital-specimen/v1/10.3535/G0G-G7D-N5J" id="1" view="JSON" weight="0"/> </locations>

Figure A.4.: The original DES Handle Record.

A.3. Chapter 5 Appendix

A.3.1. Testbed Code and Records Availability

The entire repository with the FDO-Ops testbed implementation, the example FDO information records, and a ready-to-use Jupyter Notebook to test the framework according to

our described examples is available on: https://github.com/kit-data-manager/DOIP-Client-and-TPM-Adapter-Service-for-FDOs.

A.3.2. Algorithm Examples

Listing A.1 provides a minimal generic example for Algorithm 2.

Listing A.1: Mapping of condition object C_o to record set R_f.

```
C_o = {
  "condition1": [["recordPIT1", "value1"], "recordPIT2"],
  "condition2": [["recordPIT1", "value2"], "recordPIT2"]
}

R_f = {
  "recordPIT1": ["value1"],
  "recordPIT2": ["value3"]
}

ReturnValue = true
```

Listing A.2 provides a minimal generic example for Algorithm 3.

Listing A.2: Mapping of execution-protocol e_o and FDO record R_{f_t} to an execution map M.

```
e_o = {
"execution_protocol_type": [
  ["protocol_parameter_PIT1", "[key1, {
    [sub_protocol_parameter_PIT1, [key2, target_record_PIT1,
      [true, {prefix:'[', delimiter:',', suffix:']'}]
    ]]
  }]"],
  ["protocol_parameter_PIT2", "[key3, target_record_PIT2, false]"],
  ["protocol_parameter_PIT3", "[key4, value3]"]
  ]
}

R_ft = {
  "record_PIT1": ["value1", "value2"],
  "record_PIT2": ["value4", "value5"],
}

client_input = {
  "protocol_parameter_PIT3": "value3 additionalInput"
}

M = {
  "request_index1": {
```

```
   "execution_protocol_type": [
    ["protocol_parameter_PIT1", "[key1: {
     request_index1: {
        sub_execution_protocol_type: {
         [sub_protocol_parameter_PIT1, [key2: [value1, value2]]],
        }
     }
    }]"],
    ["protocol_parameter_PIT2", "[key3: value4]"],
    ["protocol_parameter_PIT3", "[key4: value3 additionalInput]"]
   ]
  },
  "request_index2": {
   "execution_protocol_type": [
    ["protocol_parameter_PIT1", "[key1: {
     request_index1: {
        sub_execution_protocol_type: {
         [sub_protocol_parameter_PIT1, [key2: [value1, value2]]],
        }
     }
    }]"],
    ["protocol_parameter_PIT2", "[key3: value5]"],
    ["protocol_parameter_PIT3", "[key4: value3 additionalInput]"]
   ]
  }
}
```

A.3.3. DOIP/HTTP Requests

A DOIP/HTTP-based Management Operation using the convention prefix 0.DOIP for operations and status codes (e.g., 0.DOIP/Op.GET_FDO) on an FDO (e.g., exemplaryPrefix/123) is:

```
GET /doip?operationId=0.DOIP/Op.GET_FDO
&targetId=exemplaryPrefix/123
Content-Type: application/json;charset=utf-8
```

and likewise, for any Information Record- or Data Resource Operation represented as Operation FDO (e.g., exemplaryPrefix/456):

```
GET /doip?operationId=exemplaryPrefix/456
&targetId=exemplaryPrefix/123
Content-Type: application/json;charset=utf-8
```

In case additional parameters must be forwarded, the HTTP POST request is used along with the payload:

```
POST /doip?operationId=exemplaryPrefix/456
&targetId=exemplaryPrefix/123&attributes.full=true
Content-Type: application/json;charset=utf-8

{"exampleParameter": "exampleValue"}
```

A.3.4. Conflict Rules in the FDO-Graph

We define conflict rules through a combination of native Neo4j constraints and the systematic use of the `MERGE` operation. We also enforce application-specific contradictions at write time using APOC (Awesome Procedures on Cypher) library triggers. This centralizes rule logic on the server, ensures client-agnostic enforcement, and removes duplication in the application layer. The rules are in particular:

- **Node Key Constraint**: Each label, (e.g., `FDO`), requires a unique and non-null property `pid`. This prevents duplicate identifiers.
- **Type Constraint**: The property `pid` must always be of type `STRING`.
- **MERGE Semantics**: Relationships are created using `MERGE`, which guarantees idempotent creation. If the relationship already exists, no duplicate is added.
- **Contradiction rule**: Forbid same-direction pairs: if `HAS_METADATA(a,b)` exists, `IS_METADATA_FOR(a,b)` is invalid and vice versa. Likewise, if `HAS_OPERATION(x,y)` exists, `IS_OPERATION_FOR(x,y)` is invalid and vice versa.

The following lists the corresponding Cypher graph query language and rules syntax:

- **Node Key Constraint Example**:

  ```
  CREATE CONSTRAINT fdo_pid_key
  FOR (n:FDO) REQUIRE n.pid IS NODE KEY;
  ```

- **Type Constraint Example**:

  ```
  CREATE CONSTRAINT fdo_pid_type
  FOR (n:FDO) REQUIRE n.pid IS :: STRING;
  ```

- **MERGE Semantics Example**:

  ```
  CREATE (a:FDO {pid:'A'})
  CREATE (b:Operation_FDO {pid:'B'});

  // relationship (idempotent) with no duplicate created
  MERGE (a)-[:HAS_OPERATION]->(b);
  MERGE (a)-[:HAS_OPERATION]->(b);
  ```

- **APOC triggers for contradiction rule**:

```
/* Forbid both HAS_METADATA and IS_METADATA_FOR between
same pair */
CALL apoc.trigger.add('no_conflicting_metadata_edges', "
  MATCH (a:orig_FDO)-[:HAS_METADATA]->(b:orig_FDO)
  MATCH (a)-[:IS_METADATA_FOR]->(b)
  WITH count(*) AS c
  CALL apoc.util.validate(c>0,'HAS_METADATA and
  IS_METADATA_FOR cannot coexist',[])
  YIELD value RETURN value
", {phase:'before'});
```

A.3.5. IDORIS' Neo4j Graph

Figures A.5 and A.6 depict IDORI's Neo4j graph db with nodes and edges for the entire HMC KIP with attributes, an exemplary inheriting KIP "Schema-based MD KIP", and the example operation signature for "Get Related Terms".

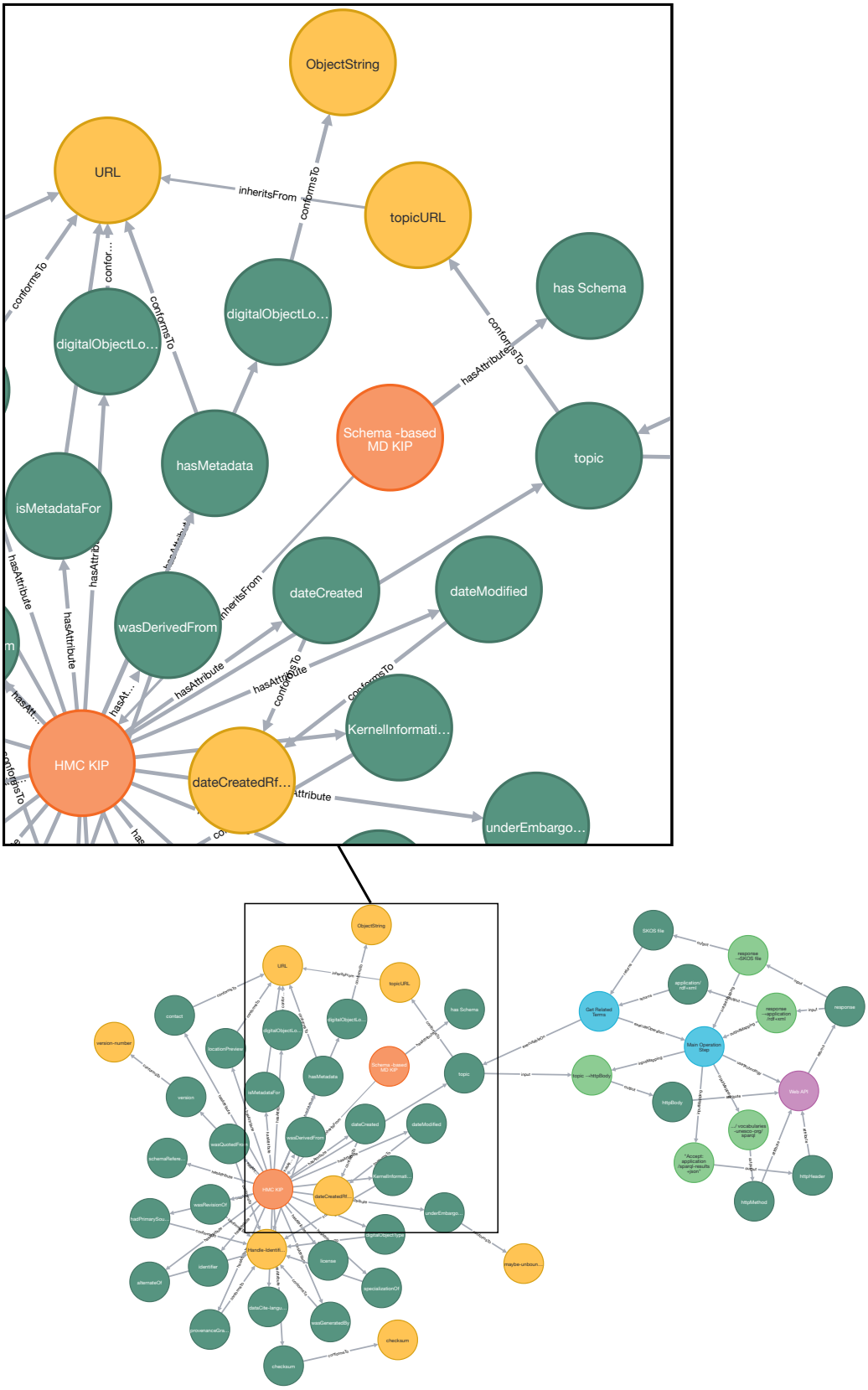


Figure A.5.: Original Neo4j graph in IDORIS, with edge annotations and node color code conforming to the advanced DTR UML model (s. Section 5.3.2). KIP and Attribute related parts are zoomed.

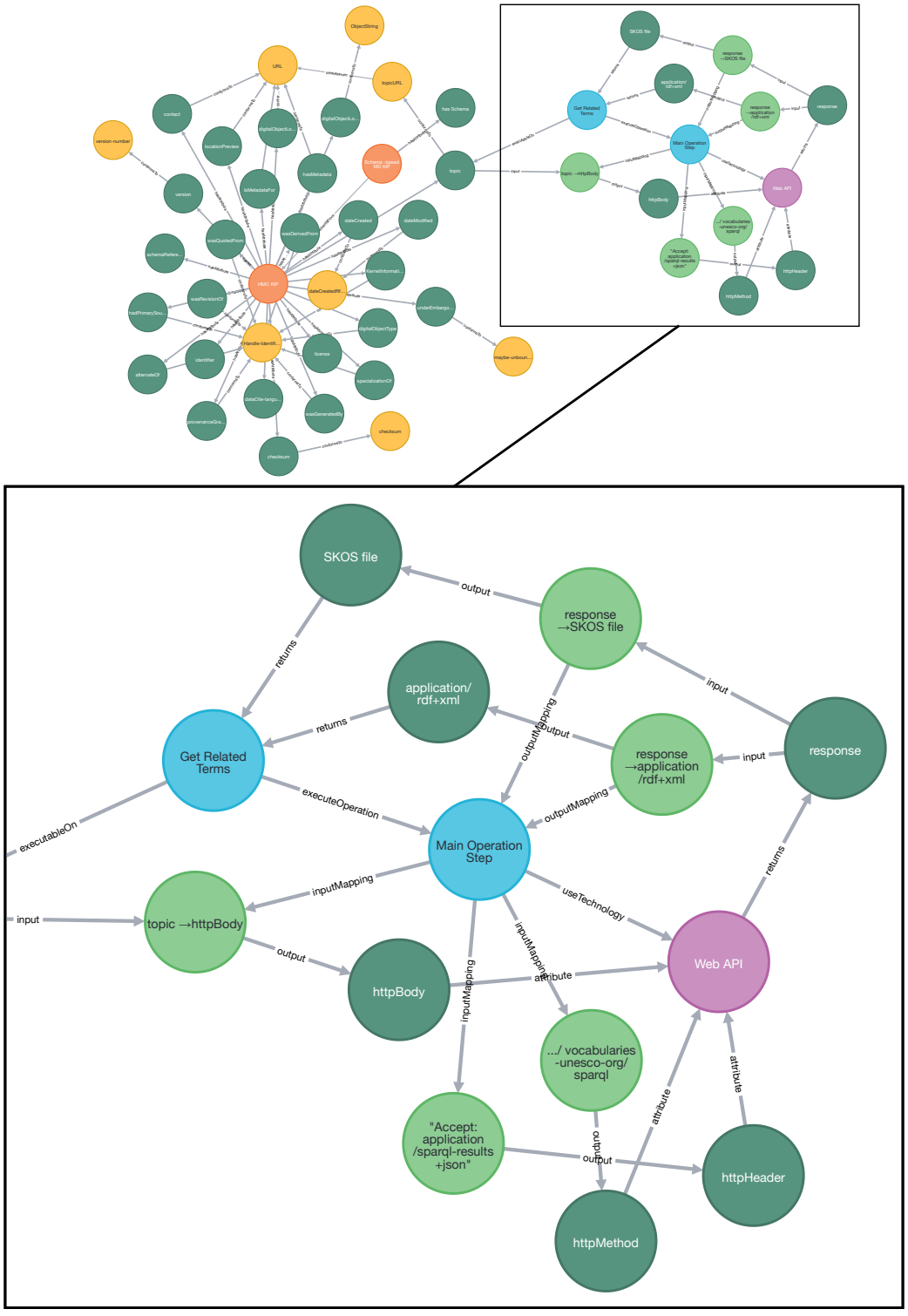


Figure A.6.: Original Neo4j graph in IDORIS, with edge annotations and node color code conforming to the advanced DTR UML model (s. Section 5.3.2). Operation signature related parts are zoomed.

A.3.6. Cross-walk Example for Get Related Terms Operation

$\mathbb{O} := \{\mathrm{o}\}$, $\mathrm{o} \mathrel{\hat{=}}$ GetRelatedTerms Operation,
$\mathbb{S} := \{s\}$, $s \mathrel{\hat{=}}$ MainOperationStep,
$\mathbb{T} := \{t\}$, $t \mathrel{\hat{=}}$ WebAPI.

$\mathbb{A} := \{a_{\text{topic}}, a_{\text{httpBody}}, a_{\text{httpMethod}}, a_{\text{httpHeader}}, a_{\text{response}}, a_{\text{rdf+xml}}, a_{\text{skos}}\}$,
$a_{\text{topic}} \mathrel{\hat{=}}$ topic, $a_{\text{httpBody}} \mathrel{\hat{=}}$ HTTP Body, $a_{\text{httpMethod}} \mathrel{\hat{=}}$ HTTP method,
$a_{\text{httpHeader}} \mathrel{\hat{=}}$ httpHeader, $a_{\text{response}} \mathrel{\hat{=}}$ response, $a_{\text{rdf+xml}} \mathrel{\hat{=}}$ application/rdf+xml,
$a_{\text{skos}} \mathrel{\hat{=}}$ SKOS file.

$\mathbb{M} := \{m_{\text{topic}\to\text{body}}, m_{\text{constMethod}}, m_{\text{constAccept}}, m_{\text{response}\to\text{rdf+xml}}, m_{\text{response}\to\text{skos}}\}$,
$m_{\text{topic}\to\text{body}} \mathrel{\hat{=}}$ from topic $\longrightarrow$ httpBody mapping,
$m_{\text{constMethod}} \mathrel{\hat{=}}$ constant HTTP GET method mapping,
$m_{\text{constAccept}} \mathrel{\hat{=}}$ constant HTTP Accept field mapping,
$m_{\text{response}\to\text{rdf+xml}} \mathrel{\hat{=}}$ response $\longrightarrow$ application/rdf+xml mapping,
$m_{\text{response}\to\text{skos}} \mathrel{\hat{=}}$ response $\longrightarrow$ SKOS file mapping.

$R_{GRT} \mathrel{\hat{=}}$ Operation FDO record for Get Related Terms,
$k_{webAPI} \mathrel{\hat{=}}$ Execution Protocol Type for Web APIs,
$P_{GRT} \mathrel{\hat{=}}$ Protocol parameter object for Get Related Terms operation.

$\mathrm{d}_{1,1} := \big((t, \text{properties}, a_{\text{httpBody}}), (t, \text{returns}, a_{\text{response}})\big) \in \mathbb{D}_1$,
$\mathrm{d}_{1,2} := \big((t, \text{properties}, a_{\text{httpMethod}}), (t, \text{returns}, a_{\text{response}})\big) \in \mathbb{D}_1$,
$\mathrm{d}_{1,3} := \big((t, \text{properties}, a_{\text{httpHeader}}), (t, \text{returns}, a_{\text{response}})\big) \in \mathbb{D}_1$.

$\mathrm{d}_{2,1} := \big((m_{\text{topic}\to\text{body}}, \text{input}, a_{\text{topic}}), (m_{\text{topic}\to\text{body}}, \text{output}, a_{\text{httpBody}})\big) \in \mathbb{D}_2$,
$\mathrm{d}_{2,2} := \big(\varnothing, (m_{\text{constMethod}}, \text{output}, a_{\text{httpMethod}})\big) \in \mathbb{D}_2$,
$\mathrm{d}_{2,3} := \big(\varnothing, (m_{\text{constAccept}}, \text{output}, a_{\text{httpHeader}})\big) \in \mathbb{D}_2$,
$\mathrm{d}_{2,4} := \big((m_{\text{response}\to\text{rdf+xml}}, \text{input}, a_{\text{response}}), (m_{\text{response}\to\text{rdf+xml}}, \text{output}, a_{\text{rdf+xml}})\big) \in \mathbb{D}_2$,
$\mathrm{d}_{2,5} := \big((m_{\text{response}\to\text{skos}}, \text{input}, a_{\text{response}}), (m_{\text{response}\to\text{skos}}, \text{output}, a_{\text{skos}})\big) \in \mathbb{D}_2$.

$\mathrm{d}_3 := \big((\mathrm{o}, \text{executeOperation}, s), \varnothing\big) \in \mathbb{D}_3$.

$\mathrm{d}_{4,1} := \big(\mathrm{d}_3, (s, \text{useTechnology}, \mathrm{d}_{1,1}), (s, \text{inputMapping}, \mathrm{d}_{2,1}), (s, \text{outputMapping}, \mathrm{d}_{2,4})\big) \in \mathbb{D}_4$,
$\mathrm{d}_{4,2} := \big(\mathrm{d}_3, (s, \text{useTechnology}, \mathrm{d}_{1,1}), (s, \text{inputMapping}, \mathrm{d}_{2,1}), (s, \text{outputMapping}, \mathrm{d}_{2,5})\big) \in \mathbb{D}_4$,
$\mathrm{d}_{4,3} := \big(\mathrm{d}_3, (s, \text{useTechnology}, \mathrm{d}_{1,2}), (s, \text{inputMapping}, \mathrm{d}_{2,2}), (s, \text{outputMapping}, \mathrm{d}_{2,4})\big) \in \mathbb{D}_4$,
$\mathrm{d}_{4,4} := \big(\mathrm{d}_3, (s, \text{useTechnology}, \mathrm{d}_{1,2}), (s, \text{inputMapping}, \mathrm{d}_{2,2}), (s, \text{outputMapping}, \mathrm{d}_{2,5})\big) \in \mathbb{D}_4$,
$\mathrm{d}_{4,5} := \big(\mathrm{d}_3, (s, \text{useTechnology}, \mathrm{d}_{1,3}), (s, \text{inputMapping}, \mathrm{d}_{2,3}), (s, \text{outputMapping}, \mathrm{d}_{2,4})\big) \in \mathbb{D}_4$,
$\mathrm{d}_{4,6} := \big(\mathrm{d}_3, (s, \text{useTechnology}, \mathrm{d}_{1,3}), (s, \text{inputMapping}, \mathrm{d}_{2,3}), (s, \text{outputMapping}, \mathrm{d}_{2,5})\big) \in \mathbb{D}_4$.

$g_1(\mathrm{o}, \mathsf{executableOn}, a_{\text{topic}}, \varnothing) \mapsto \{(a_{\text{topic}}, \varnothing)\} \in C_o$.

$g_2(\mathrm{d}_{4,1})$
$\mapsto \big(\mathsf{httpBody}, (\texttt{-data-urlencode}, \langle 21.\text{T}11148/\text{b}415\text{e}16\text{fbe}4\text{ca}40\text{f}2270\rangle,$
$\texttt{[true,[None,' ',None]]})\big) \in P_{GRT}$,
$g_2(\mathrm{d}_{4,2})$
$\mapsto \big(\mathsf{httpBody}, (\texttt{-data-urlencode}, \langle 21.\text{T}11148/\text{b}415\text{e}16\text{fbe}4\text{ca}40\text{f}2270\rangle,$
$\texttt{[true,[None,' ',None]]})\big) \in P_{GRT}$,
$g_2(\mathrm{d}_{4,3}) \mapsto \big(\mathsf{httpMethod}, (\texttt{-G}, \texttt{../vocabularies-unesco-org/sparql}, \varnothing\big) \in P_{GRT}$,
$g_2(\mathrm{d}_{4,4}) \mapsto \big(\mathsf{httpMethod}, (\texttt{-G}, \texttt{../vocabularies-unesco-org/sparql}, \varnothing\big) \in P_{GRT}$,
$g_2(\mathrm{d}_{4,5}) \mapsto \big(\mathsf{httpHeader}, (\texttt{-H}, \texttt{"Accept: application/sparql-results+json"}, \varnothing\big) \in P_{GRT}$,
$g_2(\mathrm{d}_{4,6}) \mapsto \big(\mathsf{httpHeader}, (\texttt{-H}, \texttt{"Accept: application/sparql-results+json"}, \varnothing\big) \in P_{GRT}$.

$g_{3\circ 2}\big((\mathrm{o}, \mathsf{execution}, d_{4,1}), \mathsf{returns}, a_{\text{rdf+xml}})\big)$
$\mapsto \{(k_{\text{webAPI}}, \{P_{GRT}(d_{4,1})\}), (\mathsf{outputType}, [a_{\text{rdf+xml}}, a_{\text{skos}}])\} \subseteq R_{GRT}$,
$\ldots$,
$g_{3\circ 2}\big((\mathrm{o}, \mathsf{execution}, d_{4,6}), \mathsf{returns}, a_{\text{rdf+xml}})\big)$
$\mapsto \{(k_{\text{webAPI}}, \{P_{GRT}(d_{4,6})\}), (\mathsf{outputType}, [a_{\text{rdf+xml}}, a_{\text{skos}}])\} \subseteq R_{GRT}$,
$g_{3\circ 2}\big((\mathrm{o}, \mathsf{execution}, d_{4,1}), \mathsf{returns}, a_{\text{skos}})\big)$
$\mapsto \{(k_{\text{webAPI}}, \{P_{GRT}(d_{4,1})\}), (\mathsf{outputType}, [a_{\text{rdf+xml}}, a_{\text{skos}}])\} \subseteq R_{GRT}$,
$\ldots$,
$g_{3\circ 2}\big((\mathrm{o}, \mathsf{execution}, d_{4,6}), \mathsf{returns}, a_{\text{skos}})\big)$
$\mapsto \{(k_{\text{webAPI}}, \{P_{GRT}(d_{4,6})\}), (\mathsf{outputType}, [a_{\text{rdf+xml}}, a_{\text{skos}}])\} \subseteq R_{GRT}$.

A.3.7. Sandbox Information Record of Operation FDO "Get Related Terms"

Listing A.3: Operation FDO Information Record for "Get Related Terms."


```
{
   "entries": {
    "21.T11148/aafd5fb4c7222e2d950a": [
     {
       "key": "21.T11148/aafd5fb4c7222e2d950a",
       "name": "dateCreated",
       "value": "2025-04-09T00:00:00+00:00"
     }
    ],
    "21.T11148/1c699a5d1b4ad3ba4956": [
     {
       "key": "21.T11148/1c699a5d1b4ad3ba4956",
       "name": "digitalObjectType",
       "value": "21.T11148/86c11d9215ebc2d2c8ff"
     }
    ],
    "21.T11148/2f314c8fe5fb6a0063a8": [
     {
       "key": "21.T11148/2f314c8fe5fb6a0063a8",
       "name": "licenseURL",
       "value": "https://opensource.org/license/mit"
     }
    ],
    "21.T11148/076759916209e5d62bd5": [
     {
       "key": "21.T11148/076759916209e5d62bd5",
       "name": "kernelInformationProfile",
       "value": "21.T11148/ea4e93d06a10e15d9cdf"
     }
    ],
    "21.T11148/90ee0a5e9d4f8a668868": [
     {
       "key": "21.T11148/90ee0a5e9d4f8a668868",
       "name": "operationName",
       "value": "get related terms"
     }
    ],
    "21.T11148/2694e4a7a5a00d44e62b": [
     {
       "key": "21.T11148/2694e4a7a5a00d44e62b",
       "name": "requiredType", "value": "{'21.T11148/a976172668c68034d96c': [{'key': '21.T11148/a976172668c68034d96c', 'name': 'requiredKey', 'value':
              '21.T11148/b415e16fbe4ca40f2270'}], '21.T11148/dece113486d8c5ebcf8d': [{'key': '21.T11148/dece113486d8c5ebcf8d', 'name': 'requiredValue',
              'value': 'NoType'}]}"
     },
    "21.T11148/f5c452794e258e83e4ad": [
     {
       "key": "21.T11148/f5c452794e258e83e4ad",
       "name": "outputType",
       "value": "application/rdf+xml"
     },
     {
       "key": "21.T11148/f5c452794e258e83e4ad",
       "name": "outputType",
       "value": "http://www.w3.org/2004/02/skos/core#"
     }
    ],
    "21.T11148/a1fe3f60497302ae8b04": [
     {
       "key": "21.T11148/a1fe3f60497302ae8b04",
       "name": "webAPIExecutionProtocol",
       "value": "{'21.T11148/791a2295375e3cd9dfee': [{'key': '21.T11148/791a2295375e3cd9dfee', 'name': 'httpMethod', 'value': ['-G', 'https://
              vocabularies.unesco.org/sparql', 'false']}], '21.T11148/e92d9f23abd63ebea855': [{'key': '21.T11148/e92d9f23abd63ebea855', 'name': '
              httpHeader', 'value': ['-H', '\"Accept: application/sparql-results+json\"', 'false']}], '21.T11148/0c055f37730ce8b376e5': [{'key': '21.
              T11148/0c055f37730ce8b376e5', 'name': 'httpBody', 'value': ['--data-urlencode', '<21.T11148/b415e16fbe4ca40f2270>', ['true',[None, ' ',
              None ]]]}]}"
     }
    ]
   },
   "pid": "sandboxed/4c9353ef-5065-4f46-a839-014e1ffd4171"
    ]
}
```

A.3.8. Applied Use Case Examples Listings

Listing A.4: Validate_SKOS_RDF Operation overview.

```
Implementation technology: Python Script
Execution protocols:
 main_operation (Script execution protocol parameters):
   installations (1): ['OS command','python3, pip3']
   installations (2): ['pip install','skosify']
   script method: ['python3',sub_operation1]
   script argument: ['--files_dir',sub_operation2]

   sub_operation 1 (Web API execution protocol parameters):
    method: ['GET','../URL_to_script_file']

   sub_operation 2 (Web API execution protocol parameters):
    method: ['GET','21.T11148/b8457812905b83046284', 'false']
PIT for association conditions:
   digitalObjectType (21.T11148/1c699a5d1b4ad3ba4956)
   with value constellations: application/rdf+xml
      OR [application/zip AND application/rdf+xml]
   AND digitalObjectLocation (21.T11148/b8457812905b83046284)
Output type (for Validate SKOS RDF): string, boolean
```

A.4. Proof Trees

The following proof trees are derived on the basis of the syntax defined in Definitions 3 and 11 to 13. Each premise of a branch starts with no further assumptions from which the well formed typing context is concluded, or a premise already derived in an earlier proof tree.

A.4.1. Basic Type System

Array PIT Type formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau}}{\Gamma \vdash Ref(\tau)}}{\Gamma \vdash Nat \to Ref(\tau)}}{\Gamma \vdash Nat \times (Nat \to Ref(\tau))}}{\Gamma \vdash Array(\tau)} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat}}{\Gamma \vdash m : Nat} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau}}{\Gamma \vdash v : \tau}}{\Gamma \vdash array(m, v) : Array(\tau)}$$

Array Cardinality=1 PIT Type formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash m : Nat}}{\Gamma \vdash Fin(m)}}{\Gamma \vdash Fin(1)} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau}}{\Gamma \vdash Ref(\tau)}}{\Gamma \vdash Fin(1) \times Ref(\tau)}}{\Gamma \vdash Array_{Card.1}(\tau)} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash m : Nat}}{\Gamma \vdash Fin(m)}}{\Gamma \vdash Fin(1)} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau}}{\Gamma \vdash v : \tau}}{\Gamma \vdash array(1, v) : Array_{Card.1}(\tau)}$$

Array PIT KIP Type formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_1}}{\Gamma \vdash Ref(\tau_1)}}{\Gamma \vdash Nat \to Ref(\tau_1)}}{\Gamma \vdash Nat \times (Nat \to Ref(\tau_1))}}{\Gamma \vdash Array(\tau_1)} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_n}}{\Gamma \vdash Ref(\tau_n)}}{\Gamma \vdash Nat \to Ref(\tau_n)}}{\Gamma \vdash Nat \times (Nat \to Ref(\tau_n))}}{\Gamma \vdash Array(\tau_n)}}{\Gamma \vdash \{a_1 : Array(\tau_1)_1 \ldots a_n : Array(\tau_n)_n\}}$$

Array Cardinality=1 PIT KIP Type formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash m : Nat}}{\Gamma \vdash Fin(m)}}{\Gamma \vdash Fin(1)} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_1}}{\Gamma \vdash Fin(1) \times Ref(\tau_1)}}{\Gamma \vdash Array_{Card.1}(\tau_1)} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash m : Nat}}{\Gamma \vdash Fin(m)}}{\Gamma \vdash Fin(1)} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_n}}{\Gamma \vdash Fin(1) \times Ref(\tau_n)}}{\Gamma \vdash Array_{Card.1}(\tau_n)}$$

$$\Gamma \vdash \{a_1 : Array_{Card.1}(\tau_1)_1 \ldots a_n : Array_{Card.1}(\tau_n)\}_n$$

Array and Array Cardinality=1 PIT Mixture KIP Type formation

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash Nat} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_1}}{\Gamma \vdash Nat \to Ref(\tau_1)}}{\Gamma \vdash Nat \times (Nat \to Ref(\tau_1))}}{\Gamma \vdash Array(\tau_1)} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash m : Nat}}{\Gamma \vdash Fin(m)}}{\Gamma \vdash Fin(1)} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_n}}{\Gamma \vdash Fin(1) \times Ref(\tau_n)}}{\Gamma \vdash Array_{Card.1}(\tau_n)}$$

$$\Gamma \vdash \{a_1 : Array(\tau_1)_1 \ldots a_n : Array_{Card.1}(\tau_n)_n\}$$

Array PIT Record Value formation

$$\frac{\overline{\Gamma \vdash array(m, v_1) : Array(\tau_1)} \quad \overline{\Gamma \vdash array(m, v_n) : Array(\tau_n)} \quad \overline{\Gamma \vdash \{a_1 : Array(\tau_1)_1 \ldots a_n : Array(\tau_n)_n\}}}{\Gamma \vdash \{a_1 = array(m_1, v_1)_1, \ldots, a_n = array(m_n, v_n)_n\} : \{a_1 : Array(\tau_1)_1, \ldots, a_n : Array(\tau_n)_n\}}$$

Array Cardinality=1 PIT Record Value formation

$$\frac{\overline{\Gamma \vdash array(1, v_1) : Array_{Card.1}(\tau_1)} \quad \overline{\Gamma \vdash array(1, v_n) : Array_{Card.1}(\tau_n)} \quad \overline{\Gamma \vdash \{a_1 : Array_{Card.1}(\tau_1)_1 \ldots a_n : Array_{Card.1}(\tau_n)_n\}}}{\Gamma \vdash \{a_1 = array(1, v_1)_1, \ldots, a_n = array(1, v_n)_n\} : \{a_1 : Array_{Card.1}(\tau_1)_1, \ldots, a_n : Array_{Card.1}(\tau_n)_n\}}$$

Array and Array Cardinality=1 PIT Mixture Record Value formation

$$\frac{\overline{\Gamma \vdash array(1, v_1) : Array(\tau_1)} \quad \overline{\Gamma \vdash array(1, v_n) : Array_{Card.1}(\tau_n)} \quad \overline{\Gamma \vdash \{a_1 : Array(\tau_1)_1 \ldots a_n : Array_{Card.1}(\tau_n)_n\}}}{\Gamma \vdash \{a_1 = array(m_1, v_1)_1, \ldots, a_n = array(1, v_n)_n\} : \{a_1 : Array(\tau_1)_1, \ldots, a_n : Array_{Card.1}(\tau_n)_n\}}$$

Array and Array Cardinality=1 PIT Mixture KIP Type uniform write as Array PIT KIP Type

$$\frac{\Gamma \vdash \{a_1 = array(m_1, v_1)_1, \ldots, a_n = array(1, v_n)_n\} : \{a_1 : Array(\tau_1)_1, \ldots, a_n : Array_{Card.1}(\tau_n)_n\}}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\} : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}$$

A.4.2. Record Typing

FDO Type by Record formation

$$\frac{\dfrac{\dfrac{\overline{\Gamma \vdash \diamond}}{\Gamma \vdash R} \quad \dfrac{}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash R : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}} \quad \frac{\dfrac{}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\}} \quad \dfrac{\overline{\Gamma \vdash \diamond}}{\Gamma \vdash T_{rec}}}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\} : T_{rec}}}{\Gamma \vdash R : T_{rec}}$$

Array PIT Record Subtype by PIT Subtypes formation

$$\frac{\dfrac{\dfrac{\overline{\Gamma \vdash \tau} \quad \overline{\Gamma \vdash \tau'}}{\Gamma \vdash \tau' <: \tau} \quad \dfrac{\overline{\Gamma \vdash Array(\tau)}}{\Gamma \vdash \xi} \quad \dfrac{\overline{\Gamma \vdash Array(\tau')}}{\Gamma \vdash \xi'}}{\Gamma \vdash \xi' <: \xi} \quad \frac{\overline{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash \{a_j : \xi_j\}} \quad \frac{\overline{\Gamma \vdash \{a_1 : \xi'_1, \ldots, a_n : \xi'_n\}}}{\Gamma \vdash \{a_j : \xi'_j\}}}{\Gamma \vdash \{a_j : \xi'_j\} <: \{a_j : \xi_j\}}$$

FDO by Record Type Subtype Formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\}}{\Gamma \vdash R_A} \quad \dfrac{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}{\Gamma \vdash T_{rec_A}}}{\Gamma \vdash R_A : T_{rec_A}} \quad \dfrac{\dfrac{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_z\}}{\Gamma \vdash R_B} \quad \dfrac{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_z : \xi_z\},\ z \geq n}{\Gamma \vdash T_{rec_B}}}{\Gamma \vdash R_B : T_{rec_B}}}{\dfrac{\Gamma \vdash T_{rec_B} <: T_{rec_A}}{\Gamma \vdash R_B, R_A : T_{rec_A}}}$$

A.4.3. Profile Typing

Array and Array Cardinality=1 PIT Mixture KIP Type with Profile PIT uniform write

$$\frac{\Gamma \vdash \{a_1 = array(m_1, v_1)_1, \ldots, a_p = array(1, v_p)_1 \ldots, a_n = array(1, v_n)_n\} : \{a_1 : Array(\tau_1)_1, \ldots, a_p : Array_{Card.1}(\tau_p)_p, \ldots, a_n : Array_{Card.1}(\tau_n)_n\}}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_p = [v]_p, \ldots, a_n = [v]_n\} : \{a_1 : \xi_1, \ldots, a_p : \xi_p, \ldots, a_n : \xi_n\}}$$

Record select Profile PIT formation

$$\frac{\dfrac{\dfrac{\overline{\Gamma \vdash \diamond}}{\Gamma \vdash R_A : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}} \quad \dfrac{\overline{\Gamma \vdash \diamond}}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_p = [v]_p, \ldots, a_n = [v]_n\} : \{a_1 : \xi_1, \ldots, a_p : \xi_p, \ldots, a_n : \xi_n\}}}{\Gamma \vdash R_A : \{a_1 : \xi_1, \ldots, a_p : \xi_p, \ldots, a_n : \xi_n\}}}{\Gamma \vdash R_A.a_p : \xi_p}$$

FDO Type by Profile formation

$$\frac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash T_{prof}} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash m : Nat}}{\Gamma \vdash Fin(m)}}{\Gamma \vdash Fin(1)} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_p}}{\Gamma \vdash Ref(\tau_p)}}{\Gamma \vdash Fin(1) \times Ref(\tau_p)}}{\Gamma \vdash Array_{Card.1}(\tau_p)}}{\Gamma \vdash \xi_p}}{\Gamma \vdash \xi_p \longrightarrow T_{prof}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash res}}{\Gamma \vdash res : \xi_p \longrightarrow T_{prof}} \quad \dfrac{}{\Gamma \vdash R_A.a_p : \xi_p}}{\Gamma \vdash res(R_A.a_p) : T_{prof}}$$

Record select Profile (2) PIT formation

$$\frac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash R_B : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_p = [v]_p, \ldots, a_n = [v]_n\} : \{a_1 : \xi_1, \ldots, a_p : \xi'_p, \ldots, a_n : \xi_n\}}}{\Gamma \vdash R_B : \{a_1 : \xi_1, \ldots, a_p : \xi'_p, \ldots, a_n : \xi_n\}}}{\Gamma \vdash R_B.a_p : \xi'_p}$$

FDO Type by Profile (2) formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash m : Nat}}{\Gamma \vdash Fin(m)}}{\Gamma \vdash Fin(1)} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau'_p}}{\Gamma \vdash Ref(\tau'_p)}}{\Gamma \vdash Fin(1) \times Ref(\tau'_p)}}{\Gamma \vdash Array_{Card.1}(\tau'_p)}}{\Gamma \vdash \xi'_p} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash T_{prof}}}{\Gamma \vdash \xi'_p \rightarrow T'_{prof}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash res}}{\Gamma \vdash res : \xi'_p \rightarrow T'_{prof}} \quad \dfrac{}{\Gamma \vdash R_B.a_p : \xi'_p}}{\Gamma \vdash res(R_B.a_p) : T'_{prof}}$$

FDO Type by Profile Subtype formation

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash T_{prof}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash T'_{prof}} \quad \dfrac{}{\Gamma \vdash \xi' <: \xi}}{\Gamma \vdash T'_{prof} <: T_{prof}} \quad \dfrac{}{\Gamma \vdash res(R_A.a_p) : T_{prof}} \quad \dfrac{}{\Gamma \vdash res(R_B.a_p) : T'_{prof}}}{\Gamma \vdash res(R_A.a_p), res(R_B.a_p) : T_{prof}}$$

A.4.4. Attribute Typing

Record Requirement PITs to KIP Type formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_1}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_n}}}{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}}}{\Gamma \vdash \{r_1 = v_1, \ldots, r_n = v_n\} : \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_1}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_n}}}{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}} \quad \dfrac{}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\} \to \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash mapReq : \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\} \to \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash mapReq(\{r_1 = v_1, \ldots, r_n = v_n\}) : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash mapReq(R_{req}) : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}$$

Record Requirement PITs to KIP Subtype formation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_1}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_n}}}{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}} \quad \dfrac{}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\} \to \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}}{\Gamma \vdash \{r_j : \tau_{req_j}\} \to \{a_j : \xi_j\}}}{\Gamma \vdash \{r_j : \tau_{req_j}\} \to \{a_j : Array(\tau'_j)_j\}} \quad \dfrac{\dfrac{}{\Gamma \vdash \tau} \quad \dfrac{}{\Gamma \vdash \tau'}}{\Gamma \vdash \tau' <: \tau} \quad \dfrac{}{\Gamma \vdash Array(\tau)} \quad \dfrac{}{\Gamma \vdash Array(\tau)'}}{\Gamma \vdash Array(\tau') <: Array(\tau)}}{\Gamma \vdash \{r_j : \tau_{req_j}\} \to \{a_j : Array(\tau_j)_j\}}}{\Gamma \vdash mapReq : \{r_j : \tau_{req_j}\} \to \{a_j : Array(\tau_j)_j\}} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_1}} \quad \dfrac{\dfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_n}}}{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}}}{\Gamma \vdash \{r_1 = v_1, \ldots, r_n = v_n\} : \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}} \quad \dfrac{}{\Gamma \vdash \{r_j = v_j\} : \{r_j : \tau_{req_j}\}}}{\Gamma \vdash mapReq(\{r_j = v_j\}) : \{a_j : Array(\tau_j)_j\}}}{\Gamma \vdash mapReq(R_{req}) : \{a_j : Array(\tau_j)_j\}}}{\Gamma \vdash mapReq(R_{req}) : \{a_1 : Array(\tau_1)_1, \ldots, a_n : Array(\tau_n)_n\}}}{\Gamma \vdash mapReq(R_{req}) : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}$$

Record Requirement PITs to KIP Subtype formation (for Cardinality=1 Array Types)

$$
\cfrac{
  \cfrac{
    \cfrac{
      \cfrac{
        \cfrac{
          \cfrac{}{\Gamma \vdash \tau} \quad \cfrac{}{\Gamma \vdash \tau'}
        }{\Gamma \vdash \tau' <: \tau}
        \quad \cfrac{}{\Gamma \vdash Array_{Card.1}(\tau)}
        \quad \cfrac{}{\Gamma \vdash Array_{Card.1}(\tau)'}
      }{\Gamma \vdash Array_{Card.1}(\tau') <: Array_{Card.1}(\tau)}
      \quad
      \cfrac{
        \cfrac{
          \cfrac{
            \cfrac{
              \cfrac{\cfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_1}} \quad \cfrac{\cfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_n}}
            }{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}}
            \quad \cfrac{}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}
          }{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\} \to \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}
        }{\Gamma \vdash \{r_j : \tau_{req_j}\} \to \{a_j : \xi_j\}}
      }{\Gamma \vdash \{r_j : \tau_{req_j}\} \to \{a_j : Array_{Card.1}(\tau'_j)_j\}}
    }{\Gamma \vdash \{r_j : \tau_{req_j}\} \to \{a_j : Array_{Card.1}(\tau_j)_j\}}
  }{\Gamma \vdash mapReq : \{r_j : \tau_{req_j}\} \to \{a_j : Array_{Card.1}(\tau_j)_j\}}
  \quad
  \cfrac{
    \cfrac{
      \cfrac{
        \cfrac{\cfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_1}} \quad \cfrac{\cfrac{}{\Gamma \vdash \diamond}}{\Gamma \vdash \tau_{req_n}}
      }{\Gamma \vdash \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}}
    }{\Gamma \vdash \{r_1 = v_1, \ldots, r_n = v_n\} : \{r_1 : \tau_{req_1}, \ldots, r_n : \tau_{req_n}\}}
  }{\Gamma \vdash \{r_j = v_j\} : \{r_j : \tau_{req_j}\}}
}{
  \cfrac{\Gamma \vdash mapReq(\{r_j = v_j\}) : \{a_j : Array_{Card.1}(\tau_j)_j\}}{
    \cfrac{\Gamma \vdash mapReq(R_{req}) : \{a_j : Array_{Card.1}(\tau_j)_j\}}{
      \cfrac{\Gamma \vdash mapReq(R_{req}) : \{a_1 : Array_{Card.1}(\tau_1)_1, \ldots, a_n : Array_{Card.1}(\tau_n)_n\}}{\Gamma \vdash mapReq(R_{req}) : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}
    }
  }
}
$$

FDO Type by Required PITs Record Type

$$
\cfrac{
  \cfrac{\cfrac{}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_z : \xi_z\}}}{\Gamma \vdash R : \{a_1 : \xi_1, \ldots, a_z : \xi_z\}}
  \quad \cfrac{}{\Gamma \vdash mapReq(Req) : \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}
  \quad \cfrac{\cfrac{}{\Gamma \vdash T_{Req}}}{\Gamma \vdash \{a_1 = [v]_1, \ldots, a_n = [v]_n\} : T_{Req}}
  \quad \cfrac{\cfrac{}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_z : \xi_z\}} \quad \cfrac{}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_n : \xi_n\} \text{ with } z > n}}{\Gamma \vdash \{a_1 : \xi_1, \ldots, a_z : \xi_z\} <: \{a_1 : \xi_1, \ldots, a_n : \xi_n\}}
}{\Gamma \vdash R : T_{req}}
$$