RESEARCH PAPER

A Comparative Analysis of Modeling Approaches for the Association of FAIR Digital Objects Operations

NICOLAS BLUMENRÖHR
JANA BÖHM
PHILIPP OST
MARCO KULÜKE
PETER WITTENBURG
CHRISTOPHE BLANCHI
SVEN BINGERT
ULRICH SCHWARDMANN

*Author affiliations can be found in the back matter of this article

ABSTRACT

The concept of FAIR Digital Objects represents a foundational step towards realizing machine-actionable, interoperable data infrastructures across scientific and industrial domains. As digital spaces become increasingly heterogeneous, scalable mechanisms for data processing and interpretability are essential. This paper provides a comparative analysis of various typing mechanisms to associate FAIR Digital Objects with their operations, addressing the pressing need for a structured approach to manage data interactions within the FAIR Digital Objects ecosystem. By defining and examining three core models of typing mechanisms—record typing, profile typing, and attribute typing—this work evaluates each model's quantitative quality indicators, using mathematical measures, and qualitative aspects. In particular, models are quantitatively evaluated with respect to their simplicity, efficiency, and flexibility, as well as being qualitatively assessed with respect to granularity, required client knowledge, and versatility, thereby shedding light on their strengths, limitations, and interoperability. With this assessment, our objective is to offer insights for the adoption of FDO frameworks that enhance data automation and promote the seamless exchange of digital resources across domains.

CORRESPONDING AUTHOR:
Nicolas Blumenröhr
Karlsruhe Institute of Technology, Scientific Computing Center, Germany
nicolas.blumenroehr@kit.edu

KEYWORDS:
FAIR Digital Objects; Metadata; FAIR Principles; Object-Oriented Programming; Machine-Actionability; Type System

TO CITE THIS ARTICLE:
Blumenröhr, N., Böhm, J., Ost, P., Kulüke, M., Wittenburg, P., Blanchi, C., Bingert, S. and Schwardmann, U. (2025) A Comparative Analysis of Modeling Approaches for the Association of FAIR Digital Objects Operations. *Data Science Journal* 24: 22, pp. 1–19. DOI: https://doi.org/10.5334/dsj-2025-022

1 INTRODUCTION

There is a growing trend in both science and industry to try to connect previously isolated domains, driven by the growing complexity of modern systems and the demand for interoperability. Hence, it becomes increasingly important to develop common approaches for automated acquisition, interpretability, and processing of digital data that can be considered digital resources (European Commission *et al.*, 2021; Jeffery *et al.*, 2021; Wilkinson *et al.*, 2016). Processing very large, heterogeneous, and diverse data sets from different domains using the existing sets of incompatible APIs applicable to those data sets is simply not possible (Soiland-Reyes, Goble, and Groth, 2024). However, it is widely agreed that the future of data processing must be highly automated to cope with the increasing amounts of digital resources that are of great importance for meeting the requirements of the UN Sustainable Development Goals (Madavarapu *et al.*, 2024). A foundational infrastructure that provides a common and more automatable approach to discovering and executing operations on data could have the same impact on data processing that the Internet and Web technologies have had on communication and multimedia information exchange (Schultes and Wittenburg, 2019; Wittenburg and Strawn, 2018). This could lead to large and necessary advances in scientific discovery, and industrial efficiency and sustainability.

The FAIR Digital Objects (FDOs) concept describes how such an infrastructure could be realized by representing digital resources of any type in a way that enables automated processing (Blumenröhr *et al.*, 2025; Schultes and Wittenburg, 2019; Smedt, Koureas, and Wittenburg, 2020). It does so by implementing the FAIR Principles (Wilkinson *et al.*, 2016), which provide guidelines for better data management and stewardship, using the Digital Object framework (Kahn and Wilensky, 2006). While different implementation strategies for FDOs exist, they all aim towards an automated processing by the machine-actionable characteristics of an FDO that is enabled by operations. An operation will in general be associated with an FDO by its typing mechanism and may be executed on different FDO levels, i.e., the metadata or the bit sequence of the digital resource (Blumenröhr *et al.*, 2025). Operations may range from basic Create, Read, Update and Delete (CRUD) operations to more advanced operations and can be implemented using various technologies. However, the exact specification of a type system for FDOs that enables a mechanism to associate the objects with applicable operations is not yet fully scoped (Blumenröhr *et al.*, 2025; Soiland-Reyes, Goble, and Groth, 2024). At this point, there exist different views and implementations for associating FDOs and operations by typing. In fact, 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 if and how these approaches are compatible with each other. Providing a structured analysis of these association models will support the adoption of FDOs by different communities. Associating FDOs with their operations is seen as the missing step in data processing automation by machine-actionability. Formalized type specifications and user intentions paired with formalized reuse conditions will be key in this regard.

In this work, we define and provide an assessment of typing mechanisms for associating FDOs with their operations based on different conceptual data models. We describe each data model along with an implementation example, and comparatively evaluate their characteristics with respect to these typing mechanisms. Based on the evaluation, we discuss the results in the larger context of FDO processability and perspectives for communities that want to adopt the concept.

2 BACKGROUND

2.1 FOUNDATIONS OF FDOS AND THE CORE MODEL

FDOs are persistent entities that bundle information for FAIR processing of a bit sequence including different kinds of metadata. They are referenced by a Persistent Identifier (PID), fulfill FAIR criteria in their core mechanisms, and can be protected against misuse in various dimensions (Smedt, Koureas, and Wittenburg, 2020). In the FDO core model given by (Blumenröhr *et al.*, 2025), each FDO represents a basic structure that allows for different configurations, i.e. configuration types (Lannom, Peters-von Gehlen, *et al.*, 2022), and has the following characteristics:

- A Handle PID will be resolved into an FDO information record that contains the Kernel Information.
- The Kernel Information describes the FDO core metadata attributes, such as its data type, location and additional metadata references.
- The Kernel Information is structured as a set of attributes expressed as a set of key-value pairs, aggregated by a Kernel Information Profile (Weigel *et al.*, 2019) that the information record must conform to.
- For compatibility reasons, only a minimal set of attributes are specified in the Kernel Information Profile as also proposed by the FDO Forum[1] and the Research Data Alliance[2].
- Each attribute included in the profile must be defined and registered in a public registry according to the specification of PID-Information Types (PITs) (Schwardmann, 2017), making it machine-interpretable.
- It is actionable through a set of operations that are associated with the Kernel Information via a typing mechanism.

This minimal definition of the FDOs follows the original idea of the Internet, which defines a basic package structure for information transfer and allows making use of a communication protocol for FDOs, the Digital Object Interface Protocol (DOIP) (DONA Foundation, 2018). FDOs can represent bit sequences with different kinds of content, such as data, metadata, configurations, semantic assertions, software, etc. As illustrated in Figure 1, due to their conceptual core model, FDOs have the potential to be used as a basic interoperability layer to connect different types of repositories and data spaces (Curry, Scerri, and Tuikka, 2022). For further technical details on FDOs, see the FDO Overview (Anders *et al.*, 2023a), and the FDO Requirement Specifications (Anders *et al.*, 2023b). Note that the term *profile* is used interchangeably with the term *Kernel Information Profile* in the subsequent sections.

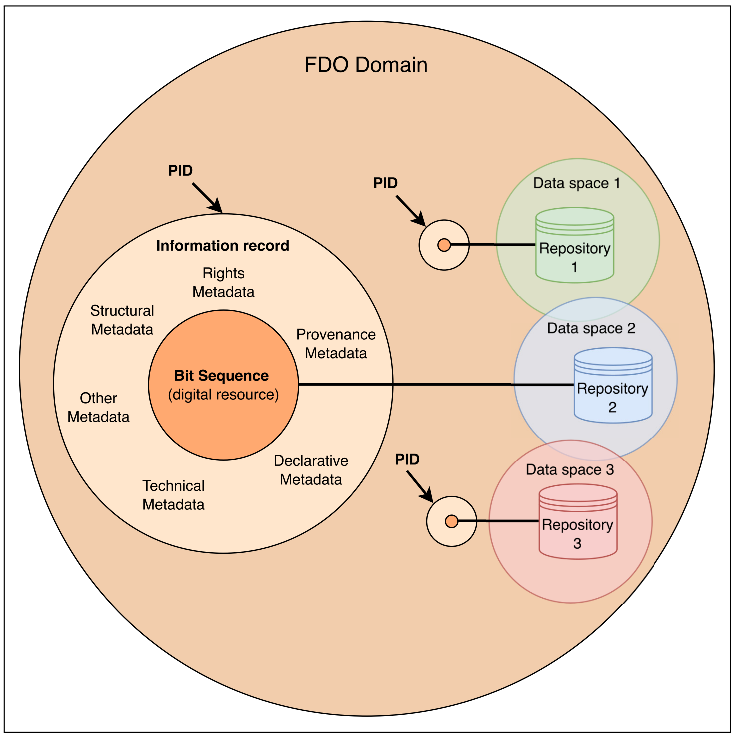

Figure 1 The conceptual FDO core model.

2.2 PROBLEM DESCRIPTION

Several works on FDO implementations have described the theoretical applicability of FDO operations, e.g. (Blanchi, Gebre, and Wittenburg, 2022; Blumenröhr *et al.*, 2025; Islam, 2023; Lannom, Koureas, and Hardisty, 2020) or have even implemented specialized systems that enable the execution of operations in their FDO ecosystem, e.g. (Blumenröhr and Aversa, 2023;

1 https://fairdo.org/.

2 https://www.rd-alliance.org/.

Islam *et al.*, 2023). However, to the best of our knowledge, a set of generic mechanisms for associating these operations with FDOs via a set of rules, i.e., a type system, in compliance with the description of the original concept, has not been worked out yet. This makes it hard to assess and to reproduce these use-case specific operation frameworks.

The authors of this work have developed typing mechanisms to associate FDOs and operations within their organizations, which were extensively discussed in the frame of the FDO Forum. At this point, there exist some reference implementations for these mechanisms as described in the following sections, but no detailed definition of their data models and how these compare to each other. We therefore see this paper as a step forward in assessing these association models and providing a baseline for implementing (inter-)operable FDO ecosystems.

3 MODELS FOR ASSOCIATING FDOS TO THEIR OPERATIONS

In this section, we first describe the different modeling approaches for the association of FDOs with operations and their underlying typing mechanisms. We assume that an FDO is specified according to the core model described in section 2.1. We first elaborate on the general idea of the typing mechanisms that we define as part of a type system for FDOs, and second on the rules of how they integrate with different FDO components. These typing mechanisms are related to well-known typing principles in computer science and are finally incorporated in each association model. Technical implementation details for these association models are not considered.

In the second part of this section, we go through several application examples that use these different association models based on the typing mechanisms.

3.1 TYPING MECHANISMS

The problem with the terms 'type' and 'typing' is that they are generic, and often have different definitions across disciplines and technologies. This work does not aim to provide an exhaustive description of these terms but it does require a more concrete description in the context of FDOs. It can be said at this point that many of the terms employed relate to ideas from the field of Object-oriented Programming (OOP), of which relations to other principles such as abstraction and encapsulation have already been described by the work of (Blumenröhr *et al.*, 2025; Schultes and Wittenburg, 2019). The next step is to infer mechanisms for associating operations on the basis of abstraction and encapsulation provided by FDOs. It is important to note that we consider the analogy between OOP and FDOs only on an abstract, conceptual level, whilst the implementation details of FDOs are a different aspect. The following terms also found in OOP are therefore defined in the context of FDOs as the following:

- Abstraction and Encapsulation: FDOs pack data and metadata into a single unit by definition, encapsulating internal details. The interface to the FDO is given by attributes that describe possible interactions. The set of attributes is given by its profile. The profile itself is therefore a class. It is an abstraction of all FDOs that satisfy the profile requirements.
- FDO Type: a characterization of an FDO through the set of its typed attributes (e.g. using PITs) that are bundled in a profile and are subject to syntactic and semantic specifications.
- Type System: inspired by the work of (Pierce, 2002), we define this as a set of rules for validating how FDOs are typed and associated with a set of operations by one or more typing mechanisms.
- Typing Mechanism: the exact procedure to determine if and how an operation is associated with a particular FDO via its kernel information elements, i.e., key-value pairs of typed attributes and profile.

The typing mechanisms to associate operations with FDOs are described below. The details and relations of these mechanisms to principles known from OOP are illustrated in Figure 2.

With respect to the association approach, there are two obvious possibilities. The first is to extend the FDO interfaces and to include operations as attributes in the FDO record by changing the profile (operation association to FDO). The other is to leave the interfaces of FDOs unchanged and to describe requirements for the interfaces of the operation representation (FDO association to operation). This can be represented as relations between operations and types (i.e., typed attributed using PITs) in a dedicated type system.

Even though object association to operations is also possible within OOP, operation association to objects of classes is encouraged there as part of the encapsulation. Operations behind Representational State Transfer (REST) services are also usually associated to the objects behind their interfaces. Object association to operation is more commonly used in the context of media types, in which the applicability of an operation is decided by the type of object. The type encapsulates the internal complexity of both the object and the operation. This results in three core mechanisms of typing that we detail in the following.

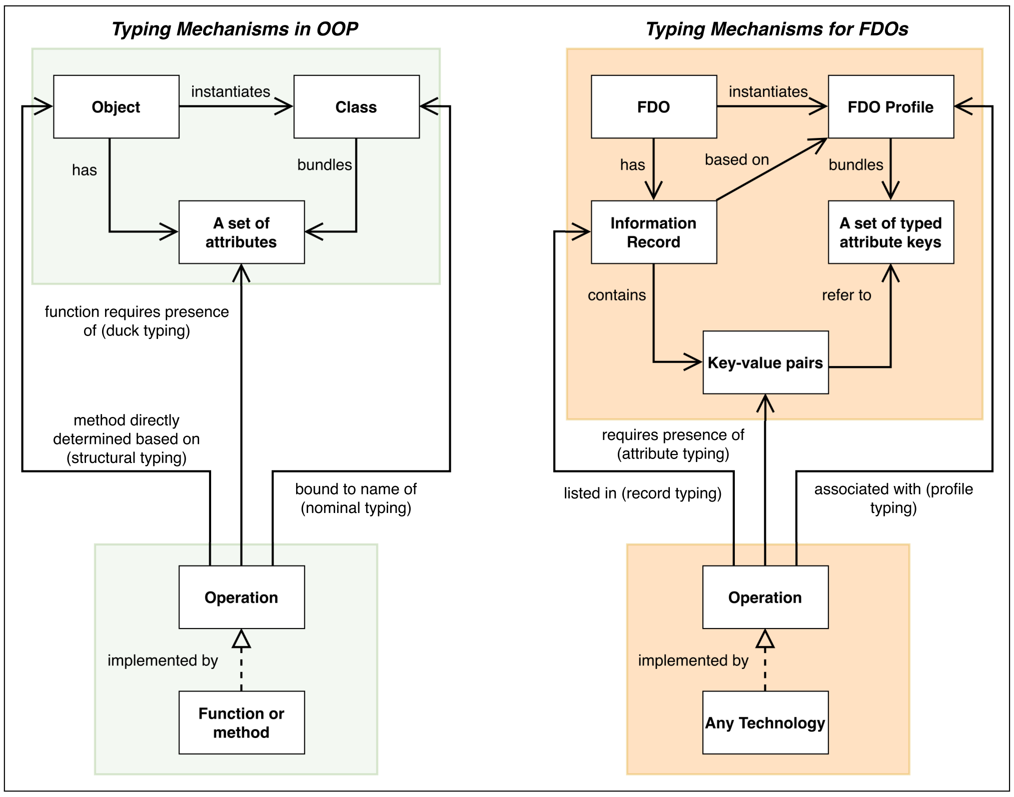
Typing Mechanisms in OOP
Object
instantiates
Class
has
bundles
A set of attributes
function requires presence of (duck typing)
method directly determined based on (structural typing)
bound to name of (nominal typing)
Operation
implemented by
Function or method
Typing Mechanisms for FDOs
FDO
instantiates
FDO Profile
has
based on
bundles
Information Record
A set of typed attribute keys
contains
refer to
Key-value pairs
requires presence of (attribute typing)
listed in (record typing)
associated with (profile typing)
Operation
implemented by
Any Technology

Figure 2 Typing Mechanisms. The conceptual typing mechanism to associate FDOs and their operations in analogy to OOP.

3.1.1 Record Typing

The most straightforward way of typing FDOs can be achieved by specifying an operation directly in the information record of the FDO as a key-value pair using typed attributes, thereby directly associating each operation with the individual object. The type is hereby purely defined by the constellation of applicable operations. Conceptually, this is similar to the principle of structural typing in OOP, in which the type of an object is determined by the methods it supports at compile time rather than by its explicit class. This focuses on what the object can do rather than what it is. All applicable operations are therefore also part of the attributes in the FDO information record and are fixed at instantiation time of the object.

3.1.2 Profile Typing

Profile typing means that operations that are associated with an FDO are inferred from the profile that is instantiated by this FDO and are therefore considered the type. Attaching the operations to FDO profiles is possible because each FDO has a profile as a mandatory typed attribute in its information record according to the kernel information requirements. This is comparable to nominal typing in OOP in which an operation in the form of a method is bound to a class and its name, meaning that it operates on instances of that class (objects) and has access to the class's attributes.

3.1.3 Attribute Typing

This typing mechanism considers the set of attributes in an FDO's information record, such that each operation is associated by the presence of one or more attributes that constitute the type in dependency of these requirements. This also relates to duck typing in OOP with the aspect that an object's usability can also be determined by the presence of specific attributes at runtime, rather than the object's class. 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. In principle, the association can be determined by considering one or more typed attributes, validating either only their key presence, or the presence of specific key-value pairs.

3.2 IMPLEMENTATION EXAMPLES

The examples described in this subsection originate from different projects and organizations the authors are involved in, using different types of data, technologies and service architectures. We concentrate here on the association models and the essential workflow, also considering information exchange between FDO services and the client side. Apart from a minimal necessary description, we do not therefore provide technical details of each implementation and the service components that are used in these projects. We also do not further explain the details of how these operations are ultimately applied to the contents of the FDOs they are associated with. For this, we refer to the references provided in each section. We also want to point out that different complexity levels of these implementations are not necessarily related to the complexity of the individual association model. These will be evaluated in section 4. However, according to the FDO core model, each FDO in these examples is registered at—and is thus resolvable via—the Handle Registry, has a typed information record, and complies to one of the known FDO configuration types.

3.2.1 Record Typing in Interactive Computing Environments

This example considers a simple FDO information record that represents a catalog containing links to various climate model simulations described by domain-specific metadata key-value pairs. FDO-related information is statically implemented in the record. Hence, Figure 3 lays out how the implementation of an association mechanism for operations via record typing works in principle. The diagram shows a workflow illustrating the interaction between an FDO and a client using a computational environment, i.e., a Jupyter Notebook, to retrieve predefined operations (here labeled as operation 1 for opening the catalog and operation 2 for reading the catalog) that are bundled in the information record among other metadata required to execute the operation, such as the content type, the reference to the bit sequence, or other metadata.

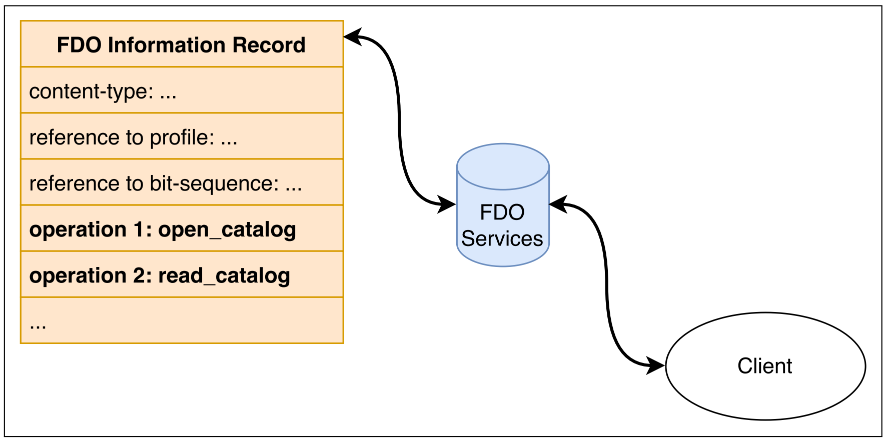
FDO Information Record
content-type: ...
reference to profile: ...
reference to bit-sequence: ...
operation 1: open_catalog
operation 2: read_catalog
...
FDO Services
Client

Figure 3 Record typing example. The conceptual workflow for interacting with an FDO based on record typing.

Depending on the FDO service, a client can either request the list of associated operations or directly retrieve them from the FDO's information record. In this example, these operations are a specification of code that is executable on the client side.

From a technical implementation perspective, the information record itself contains a set of operations that are in principle relevant to any object conforming to the content-type it represents. Note that these FDOs cannot dynamically change their operations or substitute them at runtime. The operations are fixed and cannot vary based on different FDO subtypes. The Jupyter Notebook can be found at (Kulüke, 2025).

3.2.2 Profile Typing with Multiple Registries

Within the FDO One project,[3] the focus is on providing basic operations for FDOs to build up a functional FDO ecosystem, e.g. CRUD operations (create and delete an FDO, get or update the (meta)data of an FDO) or copying an FDO and moving a distributed FDO from one storage location (data service) to another. For these types of operations, domain-specific attributes and content-types of bit sequences are irrelevant. Rather, the structure of the FDO itself is of importance, for example, whether it represents zero, one, or multiple (meta)data bit sequences and how those are stored. This information is determined by the FDO profile. Hence, the profile typing mechanism is used to associate those operations to FDO profiles. In particular, each FDO profile contains not only a list of mandatory and optional attributes which must be present in an FDO information record, but also a list of operations that can be applied to any FDO complying with this specific FDO profile. Profiles are registered in the profile registry, which is based on a Data Type Registry.[4]

As described in Figure 4, to find operations associated with an FDO, a client may retrieve the profile (either directly or through a software component) and receive a list of PIDs identifying operations that are associated to this FDO. The operations, in turn, are registered in the operation registry together with all necessary execution information[5]. For further reading and technical details of the FDO One testbed implementation, we refer to (*fairdo*, 2025).

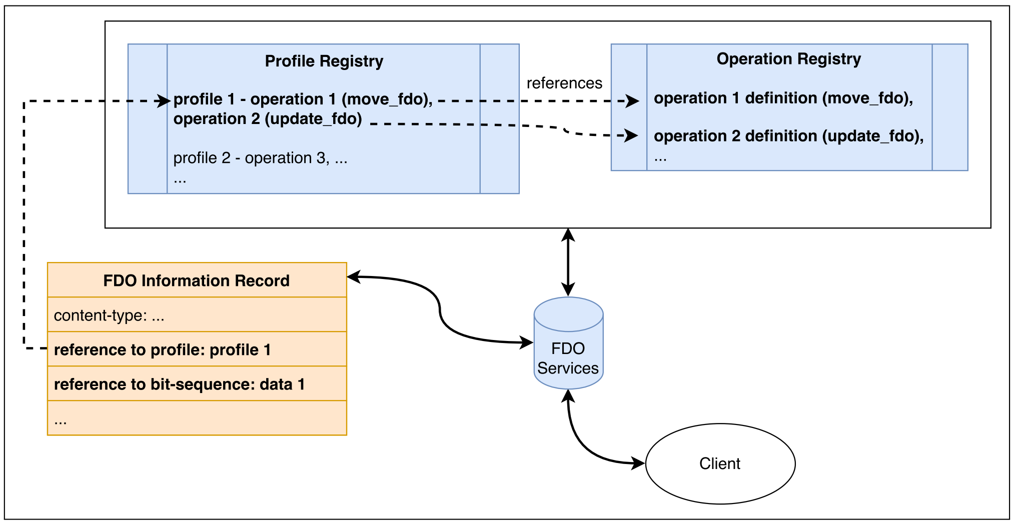
Profile Registry
profile 1 - operation 1 (move_fdo),
operation 2 (update_fdo)
profile 2 - operation 3, ...
...
references
Operation Registry
operation 1 definition (move_fdo),
operation 2 definition (update_fdo),
...
FDO Information Record
content-type: ...
reference to profile: profile 1
reference to bit-sequence: data 1
...
FDO Services
Client

Figure 4 Profile typing example. The conceptual workflow for interacting with an FDO based on profile typing. Irrespective of the service architecture that is used to implement and execute operations, such as the three registries in this example, the FDO service must infer the association between the profile of an FDO and its set of operations.

3.2.3 Attribute Typing with Operation FDOs

To realize the attribute typing mechanism, an operation must be represented in a way that allows it to be related to the attributes in the targeted FDO's information record that represents research data (i.e., labeled here as *target FDO*). This could be easily provided by representing the operation itself as an FDO as well, which we label here as *operation FDO*. This follows the concept's generic approach that each type of bit sequence can be represented as an FDO. The specific implementation of the operation is thus described in this *operation FDO* information record, detailing its implementation, possible execution mechanism, and the type-association requirements in the form of a typed attribute's key-value pair.

An example of this modeling approach is illustrated in Figure 5, where a *target FDO* and two *operation FDOs* are shown. Each *operation FDO* represents the implementation of the underlying operation that is either applied to the bit sequence, i.e., operation 1 for schema validation, or to the kernel metadata, i.e., operation 2 for license evaluation. The information

3 https://fdo-one.org.

4 https://typeregistry.lab.pidconsortium.net/.

5 Strictly speaking, in order to determine whether the above mentioned operations are executable on an FDO, the technical capabilities of the data service where the FDO is stored also need to be taken into account. This is because these types of operations do not just involve reading bit sequences, but potentially also require registering new PIDs or manipulating FDO information records, which requires actions of the data service itself. Therefore, a service registry is used to store which specific profiles and operations a data service supports. However, describing these mechanisms in detail is beyond the scope of this paper.

record of the *operation FDO* contains at least one key-value pair where the key expresses the *requiredInput* and the value references the PIT that indicates applicability of the operation to all FDOs that contain a typed attribute of this PIT in their information record. Depending on these requirements, only the corresponding key of the referenced PIT, or the key and a specific value in the form of a tuple (cf. operation 1) may be specified. This construction enables a dynamic typing mechanism, in which operations are 'aware' of the traits an FDO must have for their applicability to discover them at runtime. With respect to the infrastructure, additional services that know how to interpret and validate these type-based relations and subsequently execute the implemented operation, which is not detailed in this work, will be required. For further reading and technical details of this example, we refer to (Blumenröhr, 2025).

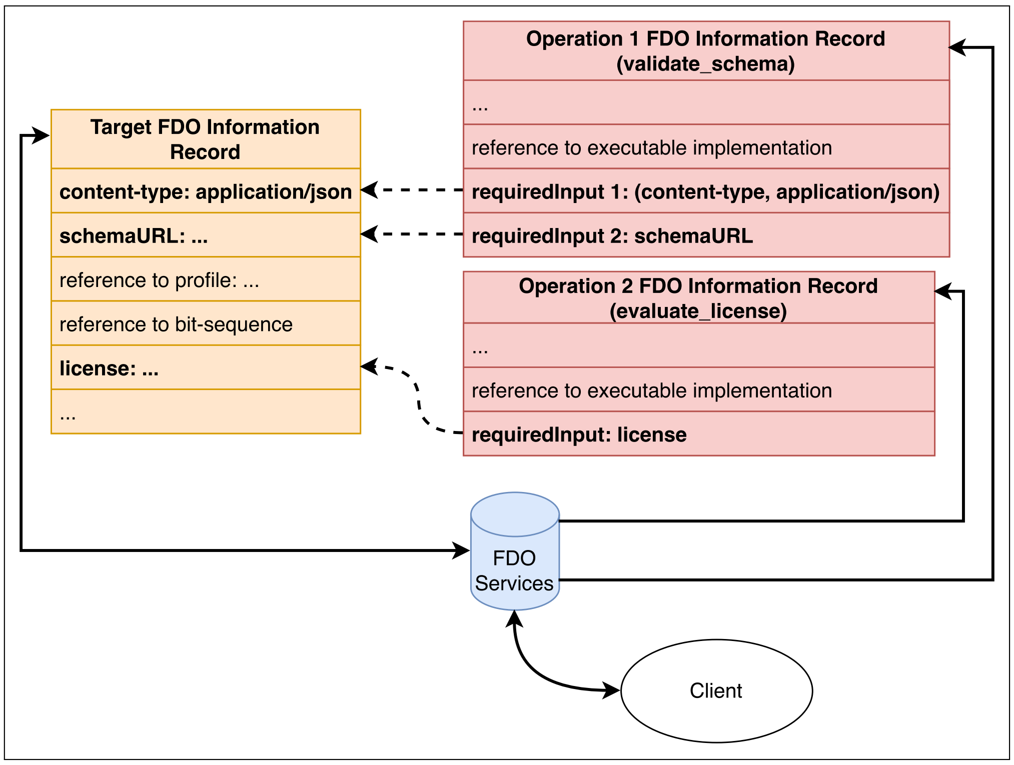
Operation 1 FDO Information Record (validate_schema)
...
reference to executable implementation
requiredInput 1: (content-type, application/json)
requiredInput 2: schemaURL
Target FDO Information Record
content-type: application/json
schemaURL: ...
reference to profile: ...
reference to bit-sequence
license: ...
...
Operation 2 FDO Information Record (evaluate_license)
...
reference to executable implementation
requiredInput: license
FDO Services
Client

Figure 5 The conceptual workflow for interacting with an FDO based on attribute typing. Irrespective of how the operation is ultimately performed (requested by the service in this example), the FDO service must infer the association based on the information record contents and references of the *target-* and *operation FDOs*.

4 MODEL EVALUATION AND DISCUSSION

To evaluate the different approaches for associating FDOs with operations based on the three typing mechanisms, we embed the association approaches into a mathematical context by modeling them as directed graphs (Section 4.1). Afterwards, a set of quality indicators is defined that are inspired by the methods used in the domain of entity-relationship modeling as described by (Moody, 1998) (Section 4.2). 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. To quantify the differences, we define metrics for these quality indicators that are evaluated on each graph model separately. In addition, we will also consider purely qualitative aspects.

However, in this work, we concentrate only on the comparison between the models rather than providing absolute numbers for the implementation examples we have introduced, as these are not relevant in the frame of a comparative analysis on the conceptual level. Furthermore, the examples will also be briefly discussed with respect to implementation aspects, limitations, and future work (Section 4.4).

4.1 MODELING THE ASSOCIATION MECHANISMS AS GRAPHS

To compare the association models not only qualitatively but also quantitatively, the three association approaches need to be put into a mathematical framework. For a distinct representation of all involved components, we model the association approaches first as Entity-Relationship (ER) Models, based on the work of (Chen, 1976), and then as directed graphs. This seems natural because associations between FDOs and operations are all based on references pointing from one entity to another entity. Those entities might be FDOs, operations, profiles (in the case of profile typing), or attribute definitions according to PITs (in the case of attribute typing). Instances of attribute definitions, i.e., typed attributes, are represented using the Attribute class in the ER model. The components of the ER model are then converted into mathematical graph components such that entities and their attributes (of the Attribute class) are represented as **vertices**, while relationships are represented as **edges**. Relationships such as 'FDO f contains attribute a in its information record' and 'attribute a points to operation o' directly translate into edges, while the entities named above, including instances of attribute definitions, translate into vertices in a graph. In this way, the ER model semantically specifies the components and structure of each association model generically, whilst the corresponding graph details the actual complexity to assess the number of elementary operations. This addresses especially the direct relationships between attribute instances and other components via edges, which can only be modeled implicitly in the ER diagram. This is further detailed in Definition 2 and visualized by Figure 6.

For the rest of this section, we index our association models with $i \in \{1, 2, 3\}$, such that $i = 1$ refers to record typing, $i = 2$ to profile typing, and $i = 3$ to attribute typing. In the following, we examine each association model separately under the assumption that the whole FDO ecosystem purely relies on a single association approach.

Definition 1 (Components)**.** Let F be the set of all FDOs representing data, O the set of all operations, P the set of all FDO profiles, and A_i^{def} the set of all attribute definitions (referring to PID-Information Types), in the whole FDO ecosystem. Attribute definitions are instantiated by typed attributes, from now on denoted only as attributes, (e.g., in FDO, operation, or profile information records) which are given by the set A_i. We denote the numbers of those quantities by $|F|$, $|O|$, $|P|$, $|A_i^{def}|$ and $|A_i|$, respectively. The set $C_i = F \cup O \cup P \cup A_i^{def}$ contains all **components** of the i-th association model.

Attribute definitions determine a key for an attribute together with a set of restrictions on the value of the attribute. Each attribute $a = (a_1, a_2) \in A_i$ is represented by a tuple that consists of a key a_1 and a value a_2. Two attributes $a = (a_1, a_2)$, $b = (b_1, b_2) \in A_i$ are considered to be the same element (i.e., $a = b$) if and only if they have the same key-value-pair (i.e., $a_1 = b_1$ and $a_2 = b_2$) and they are part of the same information record.

All components of the FDO ecosystem are uniquely identified by PIDs. Some components, such as the set of profiles and the set of attribute definitions or attributes, depend on the examined association approach. For example, attribute definitions might have different required keys and restrictions on the values depending on the model. In addition, the content of the profiles might differ according to the implementation and the chosen model. Hence, the set of attributes and the set of profiles are indexed by $i \in \{1, 2, 3\}$. The FDOs and operations are considered to be the same sets in all models (strictly speaking, we assume that there are bijective mappings $\mathcal{M}_{ij} : F_i \to F_j$ and $\mathcal{M}'_{ij} : O_i \to O_j$ between FDOs from different models and operations from different models, for $i \neq j$).

Definition 2 (Entity Relationship and Graph Models)**.** We define a simple ER and graph model for the three association approaches. The ER model is the basis specifying the elements of the set C_i as entities and their relationships, and the elements of the set A_i as attributes of these entities. Furthermore, for $i \in \{1, 2, 3\}$, we denote $G_i = (V_i, E_i)$ as the **graph** G_i, which consists of **vertices** $v_i \in V_i$ that are connected by **edges** $e_i = \{x_i, y_i\} \in E_i$ with $x_i, y_i \in V_i$.

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

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

- $i = 2$: In terms of profile typing, each FDO references a profile via an attribute in the information record. In turn, an attribute in the profile information record references an

FDO operation. Therefore,

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

- $i = 3$: For attribute typing, each *operation FDO* implicitly references a set of attributes within an FDO information record via their attribute definition and using attributes in the *operation FDO*. Hence,

$$\begin{aligned} V_3 &= F \cup A_3 \cup A_3^{def} \cup O, \\ E_3 &= \{\{o,a\} : \text{operation } o \in O \text{ has the attribute } a \in A_3\} \\ &\quad \cup \{\{a,a^{def}\} : \text{attribute } a \in A_3 \text{ references attribute } a^{def} \in A_3^{def}\} \\ &\quad \cup \{\{a',a^{def}\} : \text{attribute } a' \in A_3 \text{ references attribute } a^{def} \in A_3^{def}\} \\ &\quad \cup \{\{a',f\} : \text{attribute } a' \in A_3 \text{ is contained in FDO } f \in F\}. \end{aligned}$$

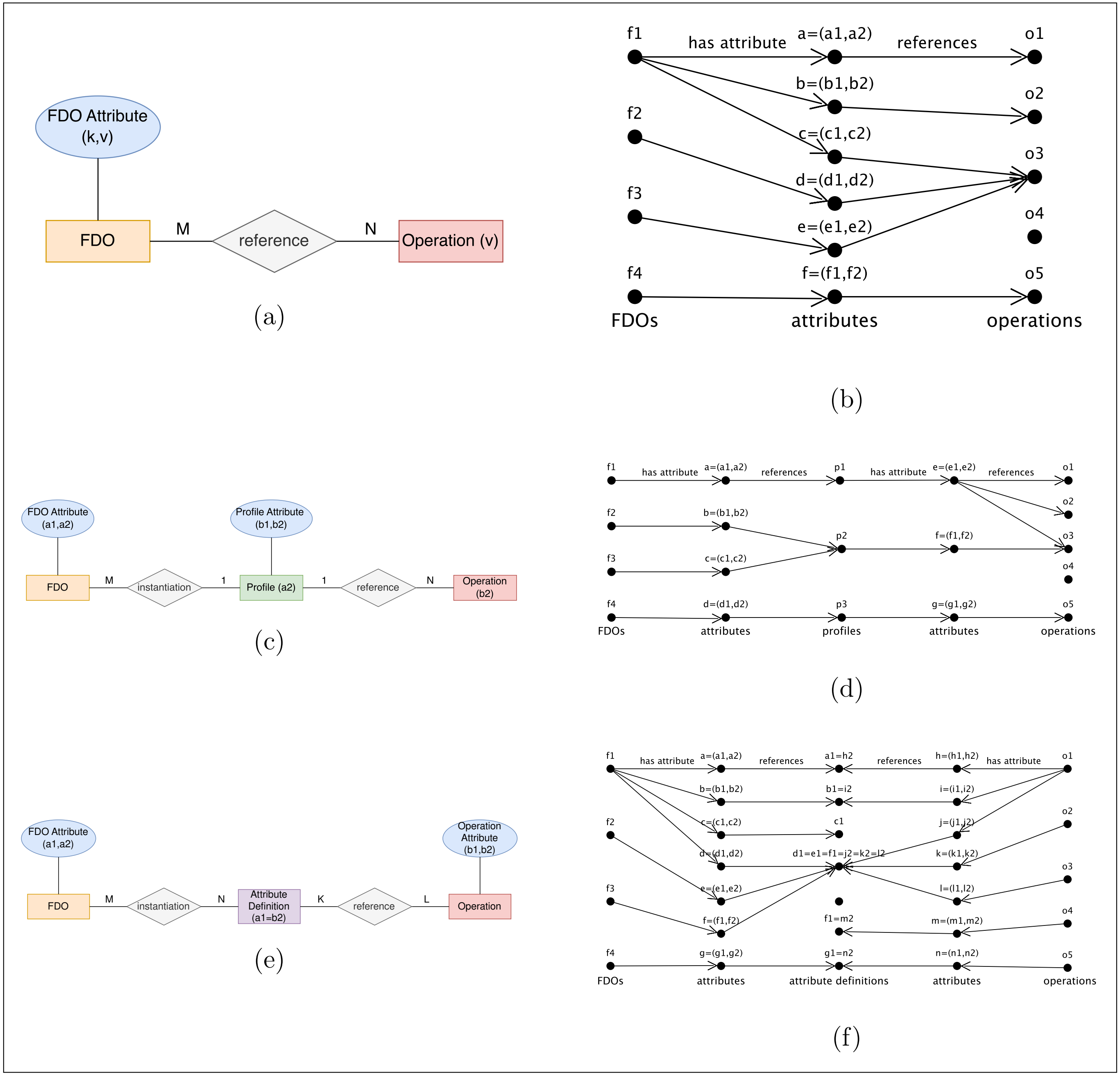

Figure 6 Entity Relationship **(a-c)** and corresponding exemplary graph representations **(d-f)**, modeling the three association approaches based on the typing mechanisms.

References point from the originating entity to the referenced entity. Note that references from attributes to attribute definitions in $i = 3$ arise from the instantiation itself. Consequently, the edges $\{x, y\} \in E_i$ are naturally ordered and may be modeled as directed edges (see Figures 6d-f where references are explicitly displayed as directed edges). However, 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. Hence, there is no need to differentiate between the orientation of edges, so we will work with simple graphs and adopt the notation as in Definition 2.

Figure 6 visualizes Definition 2. The ER models (labels a, b, c) constitute the generic constellation of the different typing mechanism models. The three graphs (labels d, e, f) derived from these ER models, respectively, illustrate an exemplary excerpt of a potential FDO ecosystem. They all contain the same FDOs $f_1, \dots, f_4$, the same operations $o_1, \dots, o_5$ 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 both associated with o_3, and f_4 is associated with o_5.

For record typing, each FDO might have several attributes for operation association, which contain the same key (i.e., $a_1 = b_1 = c_1 = d_1 = e_1 = f_1$). The attributes directly reference an operation via their value. In this example, attributes c, d, and e all have the same value (i.e., $c_2 = d_2 = e_2$) because they refer to the same operation. Each path connecting an FDO on the left side with an operation on the right side represents one FDO-operation-association.

In terms of profile typing, each FDO has exactly one attribute containing the profile reference. Those attributes have the same keys (i.e., $a_1 = b_1 = c_1 = d_1$). If two FDOs have the same profile, their attributes point to the same profile in the graph (i.e., $b_2 = c_2$). Each profile contains exactly one attribute ($e_1 = f_1 = g_1$) to specify a list of operations as its value. Similarly to record typing, each path from left to right represents one FDO-operation-association.

For attribute typing, each *target FDO* may contain multiple attributes. Similarly, each *operation FDO* may contain multiple attributes, with keys being all tantamount (i.e., $h_1 = i_1 = j_1 = k_1 = l_1 = m_1 = n_1$). The attributes in the *operation FDO* information record reference attribute definitions that are instantiated by attributes in the *target FDO* information record (i.e., in this example, we have $h_2 = a_1, i_2 = b_1, j_2 = d_1$, and so on). Note that this model is a simplification of attribute typing because we just consider the case that attributes in the operation record match with attributes in the FDO information record if the attribute in the FDO information record is present (i.e., has the desired key). We do not consider possible restrictions on the allowed values of attributes in the FDO information record and the resulting impact on granularity.

4.2 EVALUATION OF QUALITY INDICATORS AND METRICS

We examine quantitative quality indicators (simplicity, efficiency, flexibility) and qualitative aspects (granularity, required client knowledge and versatility). For the quantitative quality indicators, we define simple mathematical measures that are separately evaluated for each model under the assumption that the whole FDO ecosystem relies purely on a single association approach.

Throughout this work, we use big $\mathcal{O}$ notation to assess computational complexity of the conceptual models. Note that we generally make no assumptions about the data structure used in an implementation in which the information concerning the assessments would be stored.

Quantitative Quality Indicators

Simplicity refers to how complex it will be for a client to handle an FDO ecosystem that applies a given association model with respect to its structure. This can be measured using metrics such as the number of components involved and the number of their relations.

Efficiency takes into account how complex it will be to find all operations that are associated to an FDO or to assess whether a certain FDO is associated to a given operation. This can be measured using metrics such as the number of edges in the graph that make up an association.

Flexibility as a quality indicator relates to the question how many active modifications are required when new components are added to an existing FDO ecosystem that applies a particular association model. This can be measured using metrics such as the number of updates that must be performed when a new association between an FDO and an operation is made.

Qualitative Aspects

The qualitative aspects we consider are the *granularity* of the association models in comparison to the *amount of client knowledge* that is required to add the desired associations to a new FDO. In addition, the *versatility* of the models is discussed, which considers the possible processing options of an FDO through its associated operations in relation to the aspects imposed by *efficiency* and *flexibility*.

Definition 3. The following notation is introduced to evaluate the quantitative measures (see Theorem 5).

1. For any non-empty subset $F' \subseteq F$, $O_{F'}$ is the set of all operations that are associated with at least one FDO $f \in F'$. For the set containing a single element $F' = \{f\}$, we write O_f instead of $O_{\{f\}}$. For profile typing ($i = 2$) and a non-empty subset of profiles $P' \subseteq P$, we define the set of all operations that are referenced by at least one profile $p \in P'$ as $O_{P'}$.
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 containing a single element $O' = \{o\}$, we write F_o instead of $F_{\{o\}}$.
3. For $f \in F$ and $o \in O$, let A_f and A_o be the sets of all attributes in the information records of FDOs and operations, respectively.
4. Finally, the following definition only holds for $i = 2$: For subsets $F' \subseteq F$ and $O' \subseteq O$, 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 all profiles that are part of at least one FDO operation association between the elements of the set of F' and 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.

Definition 4 (Measures for Quantitative Quality Indicators)**.** For $i \in \{1, 2, 3\}$, we define the following metrics to assess the quality indicators:

1. $\mathcal{C}_i$ is the total number of components (see Definition 1) in the FDO ecosystem that are (potentially) part of each association mechanism. This includes not only those FDOs, operations, attribute definitions and profiles that are actually part of at least one FDO-operation-association, but also the total sets of the components that might be involved.
2. $\mathcal{A}_i$ is the total number of instantiated attributes that are present in FDO, profile, or operation information records, which are actually part of the association mechanism. Here, attributes are counted multiple times if the same key-value pair is present in multiple information records. Both $\mathcal{C}_i$ and $\mathcal{A}_i$ are indicators of the space complexity for each model.
3. $\mathcal{Q}_i$ is is an upper bound on the time complexity to decide whether an FDO $f \in F$ is associated to an operation $o \in O$.
4. $\mathcal{R}_i$ is an upper bound on the time complexity to find all FDOs that are associated with a single operation.
5. $\mathcal{S}_i$ is an upper bound on the time complexity to find all operations associated with a single FDO.
6. $\mathcal{T}_i$ is an upper bound on the time complexity to perform all required updates in the FDO ecosystem to associate a new operation with a set $F' \subseteq F$ of FDOs.
7. $\mathcal{U}_i$ is an upper bound on the time complexity to perform all required updates in the FDO ecosystem to associate a new FDO with a set of operations $O' \subseteq O$.

Theorem 5 (Evaluated Measures)**:** *The measures specified in Definition 4 are evaluated to the following quantities:*

1.
$$\mathcal{C}_1 = |F| + |O| + 1,$$
$$\mathcal{C}_2 = |F| + |O| + |P'| + 2,$$
$$\mathcal{C}_3 = |F| + |O| + |A_3^{def}|.$$

2. *For $i = 3$, let $b_1, \ldots, b_{|F_{O'}|} \in \mathbb{N}$ be the number of attributes being part of the association mechanism for the FDOs $f_1, \ldots, f_{|F_{O'}|}$, and let $d_1, \ldots, d_{|O_{F'}|} \in \mathbb{N}$ be the number of attributes taking part in the association mechanism for each operation $o_1, \ldots, o_{|O_{F'}|}$.*

$$\mathcal{A}_1 = \sum_{f \in F_{O'}} |O_f|,$$

$$\mathcal{A}_2 = |F_{O'}| + |P_{F'O'}|,$$

$$\mathcal{A}_3 = \sum_{j=1}^{|F_{O'}|} b_j + \sum_{j=1}^{|O_{F'}|} d_j.$$

3.
$$\mathcal{Q}_1 = \mathcal{O}(|A_f|),$$

$$\mathcal{Q}_2 = \mathcal{O}(|A_f| + |O_{P'_f}|),$$

$$\mathcal{Q}_3 = \mathcal{O}(|A_f| + |A_o|).$$

4.
$$\mathcal{R}_1 = \mathcal{O}\left(\sum_{f \in F} |A_f|\right),$$

$$\mathcal{R}_2 = \mathcal{O}\left(\sum_{f \in F} |A_f| + \sum_{p \in P_{F'}} |O_{\{p\}}|\right),$$

$$\mathcal{R}_3 = \mathcal{O}\left(\sum_{f \in F} |A_f| + |A_o|\right).$$

5.
$$\mathcal{S}_1 = \mathcal{O}(|A_f|),$$

$$\mathcal{S}_2 = \mathcal{O}(|A_f| + |O_{P'_f}|),$$

$$\mathcal{S}_3 = \mathcal{O}\left(|A_f| + \sum_{o \in O} |A_o|\right).$$

6.
$$\mathcal{T}_1 = \mathcal{O}(|F'|),$$

$$\mathcal{T}_2 = \mathcal{O}(|P_{\{o\}}|),$$

$$\mathcal{T}_3 = 0.$$

7.
$$\mathcal{U}_1 = \mathcal{O}(|O'|),$$

$$\mathcal{U}_2 = 0,$$

$$\mathcal{U}_3 = 0.$$

Proof. 1. According to Definition 1, the components involve the sets F, O, P and A_i^{def}. However, we just count those components that are potentially taking part in the association mechanism. For $i = 1$, this is the set of FDOs, the set of operations, and a single attribute definition (as all FDOs reference their operations via the same attribute key). For $i = 2$, there are two attribute definitions involved in the association mechanism, one to reference an FDO profile in all FDO information records, and one to reference a list of operations in all profile information records. For $i = 3$, there are no restrictions on the set of attributes that are being used in the FDO information records. Hence, all attribute definitions A_3^{def} are potentially taking part in the association mechanism.

2. Counting the number of attributes being part of the association mechanism means to count all edges with the label 'has attribute' as illustrated in Figure 6 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 associations. For $i = 2$, each FDO contains exactly one attribute to be connected to a profile (totaling $|F_{O'}|$ attributes), and each profile has exactly one attribute that connects it to a set of operations (totaling $|P_{F'O'}|$ attributes). For $i = 3$, the equation follows by definition of b_j and d_j.

3. Let $f \in F$ be any FDO and $o \in O$ be any operation. For $i = 1$, a client would need to search the whole FDO information record for the attribute containing the reference to o, taking time $\mathcal{O}(|A_f|)$. For $i = 2$, one needs to find the profile p in the FDO information record within time $\mathcal{O}(|A_f|)$. Since accessing the profile and its list of operations highly depends on the implementation but is not directly relevant for the comparative analysis, we assume access in constant time. Finally, the list of operations needs to be searched for the reference to o, taking time $\mathcal{O}(|O_{\{p\}}|)$. For $i = 3$, additionally, all attributes in the operation information record need to be found that determine the association, which is done in $\mathcal{O}(|A_o|)$. Afterwards, each of the associations that were found need to be matched against the attributes in the information record (after converting either the attributes in the FDO or the attributes in the *operation FDO* into a suitable format).

4. For $i = 1$, one has to search each FDO information record in the FDO ecosystem for its operations, which is $\mathcal{O}(\sum_{f \in F} |A_f|)$. For $i = 2$, similar time is required to find all profiles $P_{F'}$. A profile has one attribute containing a list of operations, and we assume that each list of operations can be accessed in constant time. Furthermore, checking whether those lists contain the operation requires reading the whole operation list within time $\mathcal{O}(\sum_{p \in P'_F} |O_{\{p\}}|)$. For $i = 3$, first find all operation attributes and convert them into a suitable format within time $\mathcal{O}(|A_o|)$. Then, read all attributes in all FDOs and check whether they match the operation attributes, taking time $\mathcal{O}(\sum_{f \in F} |A_f|)$.

5. For all $i \in \{1, 2, 3\}$, it is required to read all attributes in the information record. For $i = 2$, one then accesses the profile p within $\mathcal{O}(1)$ and the list of operations also within $\mathcal{O}(1)$. Reading all elements from that list takes time $\mathcal{O}(|O_{\{p\}}|)$. For $i = 3$, the FDO information record is converted into a suitable format (within $\mathcal{O}(|A_f|)$). Then, each *operation FDO* has to be checked against the *target FDO*, which requires time $\mathcal{O}(\sum_{o \in O} |A_o|)$.

6. For $i = 1$, relating a new operation to the set F' requires one to add one attribute in each FDO information record, yielding $\mathcal{O}(|F'|)$ updates in total. For $i = 2$, the new operation needs to be added to all profiles that it should be applicable to, which are $|P_{\{o\}}|$. For $i = 3$, no updates need to be done because the set F' is implicitly defined by the attributes in the *operation FDO*.

7. 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 with the same reason as detailed in 6. □

Note that the set F' in part 6 is defined by the client ($i = 1$) or is imposed by the model ($i = 2$ and $i = 3$). This is 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 $i = 2$, when a new operation is added, the associations are partly to be decided on by somebody who has the right to edit the required profiles and partly implied by the model itself (the associations between profiles and FDOs are already given and cannot be changed). For $i = 3$, the set F' is fully determined by the model in advance, depending on the attributes specified in the operation record. A similar observation applies to part 7: For $i = 1$, the client will define the set O', whereas for $i = 2$ and $i = 3$, the set O' is fully specified by the model. Such considerations need to be taken into account when assessing the quality measures.

4.3 COMPARISON OF MEASURES

We now compare the measures to evaluate the strengths and weaknesses of the different association models, starting with the quantitative measures. The overview of all measures is provided in Table 1.

- **Simplicity:** Both the number of components and the number of attributes that are part of the association mechanism are measures for the simplicity of the model. If few attributes are involved, the information records (of FDOs, profiles, and operations) can be kept comparatively short. If additionally few components are involved, the models are easier to understand for potential users. Regarding components, we have $\mathcal{C}_1 < \mathcal{C}_2$ and

$\mathcal{C}_1 \leq \mathcal{C}_3$, while for $\mathcal{C}_2$ and $\mathcal{C}_3$ the following cases are possible:

$$\mathcal{C}_2 \begin{cases} < \mathcal{C}_3 & \text{if } |P'| + 2 < |A_3^{def}| \\ = \mathcal{C}_3 & \text{if } |P'| + 2 = |A_3^{def}| \\ > \mathcal{C}_3 & \text{if } |P'| + 2 > |A_3^{def}| \end{cases}$$

In addition, there does not appear to be any general order of $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, for which we get the following estimates:

$$\mathcal{A}_1 = \sum_{f \in F_{O'}} |O_f| \leq \sum_{f \in F_{O'}} |O_{F'}| = |F_{O'}||O_{F'}|$$

$$\mathcal{A}_2 = |F_{O'}| + |P_{F'O'}| \overset{|F_{O'}| \geq 1}{=} |F_{O'}| \left(1 + \frac{|P_{F'O'}|}{|F_{O'}|}\right) \begin{cases} < 2|F_{O'}| & \text{if } |P_{F'O'}| < |F_{O'}| \\ \geq 2|F_{O'}| & \text{if } |P_{F'O'}| \geq |F_{O'}| \end{cases}$$

$$\mathcal{A}_3 = \sum_{j=1}^{|F_{O'}|} b_j + \sum_{j=1}^{|O_{F'}|} d_j \leq |F_{O'}||A_F| + |O_{F'}||A_O| \overset{|F_{O'}|=|O_{F'}|}{=} |F_{O'}|\,(|A_F| + |A_O|)$$

With that, we get $\mathcal{A}_2 \leq \mathcal{A}_1$ if $|O_{F'}| \geq 2$, which should be the most common case and assumes that the number of profiles is relatively small compared to the number of FDOs. We also get $\mathcal{A}_1 < \mathcal{A}_2$ if $|O_{F'}| < 2$, which implies $|O_{F'}| = 1$ (since $|O_{F'}| > 0$). This means that each FDO is associated to exactly one operation.

From $\mathcal{A}_1 < \mathcal{A}_3$, we get $|O_{F'}| < |A_F| + |A_O|$, which means that there are more attributes associated with FDOs and operations than there are operations associated to FDOs. Additionally, given $|F_{O'}| \leq |P_{F'O'}|$ and $|A_F| + |A_O| = 2$, we have $\mathcal{A}_3 \leq |F_{O'}|(|A_F| + |A_O|) = 2|F_{O'}| \leq \mathcal{A}_2$. This is the case when each operation relies on few attributes for association.

- **Efficiency:** All measures $\mathcal{Q}_i$, $\mathcal{R}_i$ and $\mathcal{S}_i$ quantify the effort for a client to find certain FDO-operation-associations within the FDO ecosystem. To compare those measures, we note that all upper bounds are sharp upper bounds.

  $\mathcal{Q}_i$ quantifies the effort to decide whether a certain FDO is associated to an operation. This is obviously smallest for record typing. For $\mathcal{Q}_2$ and $\mathcal{Q}_3$, the following cases are possible:

$$\mathcal{Q}_2 \begin{cases} \lesssim \mathcal{Q}_3 & \text{few operations are associated with } f\text{'s profile for } i = 2 \\ \gtrsim \mathcal{Q}_3 & \text{otherwise} \end{cases}$$

  Considering $\mathcal{R}_i$, it is obvious that $\mathcal{R}_1$ is smallest. For the other two association models, the following two cases are possible:

$$\mathcal{R}_2 \begin{cases} \lesssim \mathcal{R}_3 & \text{if (very) few operations are associated with } F \\ \gtrsim \mathcal{R}_3 & \text{otherwise} \end{cases}$$

  For example, $\mathcal{R}_2 \lesssim \mathcal{R}_3$ occurs when all $f \in F$ have the same profile. For $\mathcal{S}_i$, we observe that $\mathcal{S}_1$ is smallest, while $\mathcal{S}_2$ also scales with the number of operations related to the given profile and $\mathcal{S}_3$ scales with the number of attributes in all operations, which is considerably larger.

  Overall, this shows that record typing is the best approach in terms of efficiency. Profile typing and attribute typing are less efficient in terms of measures $\mathcal{Q}_i$ and $\mathcal{R}_i$. However, the measure $\mathcal{S}_3$ reveals the high costs of attribute typing in comparison to the other models, because one has to iterate over all attributes of all operations in the FDO ecosystem to find all operations associated to one FDO.
- **Flexibility:** Assuming that the number of FDOs associated to the new operation is much larger than the number of profiles (for $i = 2$) associated to this operation, it trivially follows that $\mathcal{T}_1 \gtrsim \mathcal{T}_2 > \mathcal{T}_3 = 0$. For $\mathcal{U}_i$, obviously $\mathcal{U}_1 > \mathcal{U}_2 = \mathcal{U}_3 = 0$. Hence, in terms of required updates, attribute typing is most efficient, followed by profile typing. In comparison, record typing is relatively inefficient.

Finally, we will comment on qualitative aspects, that is, granularity and client knowledge, as well as versatility.

- **Granularity and client knowledge:** For record typing, each FDO can be associated with any operation as desired by the client. This is the most granular approach, as any combination of FDOs and operations is possible. However, for each newly defined FDO, the client who has introduced the FDO information record has to think of which operations to include into the information record. This requires both domain knowledge regarding the content information in the FDO, and knowledge about the association mechanism. The price for higher granularity is therefore that for each new FDO a careful individual inspection might be required to make an informed decision on operation association.

  Attribute typing has a slightly smaller granularity as not every FDO can be seamlessly associated to any operation. In turn, the association mechanism works out automatically, which means that clients just need to include all information they have available into the information record, without deciding for specific operations or attributes. However, in case a client has a specific operation in mind that was not automatically associated to the FDO but which one wants to be associated, one still needs to figure out which additional attributes to include into the FDO information record.

  Profile typing is the least granular approach. As each operation is associated to a whole class of FDOs, there is a need for many different profiles to be made available to the client to reach a granularity that is comparable with the other models. The advantage of profile typing is that the client just has to make an informed choice as to which profile to use, and then will be instructed which attributes are required in the information record due to the profile definition. Hence, one does not have to think at all about associating their FDO to any operations.
- **Versatility:** In contrast to its high granularity, the overall versatility of record typing is considered to be the lowest, as each FDO-operation association must be explicitly declared to increase the possible processing options for an FDO.

  Profile typing has much greater versatility compared to record typing because a profile is typically reused several times to create a set of FDOs, and all of these FDOs automatically have the possible processing options defined by the operations associated with that profile.

  Compared to attribute typing, the versatility of profile typing is potentially lower because attribute definitions that constitute an association condition are typically reused across profiles and may occur in multiple *target 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 simply because the association between the operation and the profile was not explicitly made. In addition, an operation associated via profile typing may assume the presence of specific, not necessarily mandatory, attribute definitions in the profile. This could result in incompatibilities when executing the operation in case these attribute definitions were not instantiated for a particular FDO. With attribute typing, this cannot happen since the instantiation of all required attribute definitions is assured as part of the association process.

Table 1 Overview of measures between Record, Profile, and Attribute Typing approaches and corresponding metrics.

MEASURES	RECORD TYPING (i=1)	PROFILE TYPING (i=2)	ATTRIBUTE TYPING (i=3)	METRIC OVERVIEW
Simplicity	high	moderate	low-moderate	$\mathcal{C}_1 < \mathcal{C}_2, \mathcal{C}_1 \leq \mathcal{C}_3$ and, in general, $\mathcal{C}_2 \neq \mathcal{C}_3, \mathcal{A}_2 \leq \mathcal{A}_1$ in most cases, $\mathcal{A}_3 \leq \mathcal{A}_2$ for few attributes
Efficiency	high	moderate	low	$\mathcal{Q}_1 < \mathcal{Q}_2$ and $\mathcal{Q}_2 \lesssim \mathcal{Q}_3$ for few operations in f's profile or $\mathcal{Q}_2 \gtrsim \mathcal{Q}_3$; $\mathcal{R}_1 < \mathcal{R}_2$ and $\mathcal{R}_2 \lesssim \mathcal{R}_3$ for few operations being associated with FDOs or $\mathcal{R}_2 \gtrsim \mathcal{R}_3$; $\mathcal{S}_1 < \mathcal{S}_2 < \mathcal{S}_3$
Flexibility	low	moderate	high	$\mathcal{T}_1 \gtrsim \mathcal{T}_2 > \mathcal{T}_3, \mathcal{U}_1 > \mathcal{U}_2 = \mathcal{U}_3$
Versatility	low	moderate	moderate–high	None
Granularity and Required Client Knowledge	high	low	low–moderate	None

4.4 INTEROPERABILITY OF ASSOCIATION MODELS

In contrast to the other quality indicators, we do not define specific metrics to quantify different levels of interoperability, which is out of scope for this work. Instead, we describe the implications for interoperability of FDOs and their operations by the compatibility of the introduced association models.

Interoperability of FDO operations refers to the ability to perform consistent, standardized operations on FDOs across different systems and platforms, ensuring that actions such as accessing, processing, or transforming the objects can be executed reliably and requested uniformly by a client, regardless of the environment. From our point of view, different association models should therefore be consistent and compatible with respect to a standardized FDO type system that utilizes one or more typing mechanisms.

This ensures that when an FDO is accessed or manipulated across different platforms, its type definitions and associated operations are consistently interpreted and executed. The type system provides a common language for using the standardized structure of FDOs based on one or more typing mechanisms, enabling seamless interaction between systems. Regardless of which typing mechanisms are implemented within a service, it is critical that all clients accessing an FDO obtain the same set of associated operations independent from the underlying model.

Profiles are essential for this as they provide a minimal, standardized metadata structure for all FDOs. Because 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, they maintain a basic level of compatibility. Different association models using either record, profile or attribute typing can therefore be applied to the same FDO since the type system serves as a common language that allows different models to 'understand' which operations are valid. This also means that 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.

4.5 IMPLEMENTATION CONSIDERATIONS

Based on the results of our comparative evaluation and considering the approach of modeling the different association models using directed graphs, we conclude that implementations such as described in section 3.2.1–3.2.3 would highly profit from storing the interconnected components and the rules of each association model in proper graph data structures. This way, the assessment of the information about associations as described by the metrics is done once at ingestion time and stored as vertices and nodes according to the model, yielding the structure as illustrated in Figure 6. The repetitive procedures described by the quality indicators and quantified by their metrics are then facilitated. For example, assessment about which operations are associated with a given FDO and vice versa can be performed with simple graph queries. More complex procedures, such as is the case for attribute typing (cf. $\mathcal{S}_3$), could be also compensated this way by caching and integrating rules for inferring information. This could take place on the level of the object entities, as well as on the level of the services which store additional information about some components, for example the profiles.

5 CONCLUSIONS

In this paper, we defined and assessed multiple modeling approaches for associating FAIR Digital Objects with their operations through different typing mechanisms based on three example implementations. Our analysis underlines that each model—record typing, profile typing, and attribute typing—has distinct advantages and trade-offs for FDO ecosystems concerning simplicity, efficiency, flexibility, versatility, and granularity in conjunction with required client knowledge. While record typing offers simplicity and a high granularity, profile typing and attribute typing provide enhanced flexibility, versatility and few required client knowledge. Our findings also indicate that these association models are so far compatible with each other that a particular FDO entity could incorporate all approaches at the same time. This is also relevant with respect to interoperability between different FDO ecosystems. Ultimately, adopting an association model will depend on the specific requirements of the data environment, including client expectations and computational constraints. Future work will need to consider how to manage FDO ecosystems at scale and which technologies are most suitable for implementing different models, ensuring a robust foundation for machine-actionable data infrastructures.

ACKNOWLEDGEMENTS

The authors thank the FDO Forum group participants who contributed valuable insights through discussions and remarks during this work. Special thanks go to Larry Lannom from the Corporation for National Research Initiatives (CNRI) and Yudong Zhang from GESIS–Leibniz-Institut für Sozialwissenschaften.

FUNDING INFORMATION

This work is funded by the Helmholtz Metadata Collaboration Platform (HMC), NFDI4ING (DFG – project number 442146713), and is supported by the research program 'Engineering Digital Futures' of the Helmholtz Association of German Research Centers. Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) under grant agreement No 101093054.

AUTHOR CONTRIBUTIONS

Supervision: Nicolas Blumenröhr; Conceptualization: Nicolas Blumenröhr, Jana Böhm, Marco Kulüke, Christophe Blanchi, Peter Wittenburg, Ulrich Schwardmann, Sven Bingert; Methodology and Evaluation: Nicolas Blumenröhr, Jana Böhm, Philipp Ost; Revision of State-of-the-Art analysis and background: Nicolas Blumenröhr, Jana Böhm, Philipp Ost, Peter Wittenburg, Ulrich Schwardmann, Sven Bingert, Christophe Blanchi.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Nicolas Blumenröhr orcid.org/0009-0007-0235-4995
Karlsruhe Institute of Technology, Scientific Computing Center, Germany

Jana Böhm orcid.org/0009-0004-9802-113X
GWDG, Germany

Philipp Ost orcid.org/0000-0002-7198-0566
Karlsruhe Institute of Technology, Scientific Computing Center, Germany

Marco Kulüke orcid.org/0000-0003-0611-2567
German Climate Computing Center, Germany

Peter Wittenburg orcid.org/0000-0003-3538-0106
Max Planck Institute for Psycholinguistics, Netherlands

Christophe Blanchi orcid.org/0000-0003-2277-5176
DONA Foundation, Switzerland

Sven Bingert orcid.org/0000-0001-9547-1582
GWDG, Germany

Ulrich Schwardmann orcid.org/0000-0001-6337-8674
GWDG, Germany

REFERENCES

Anders, I. *et al.* (2023a) *FAIR Digital Object Technical Overview*. Available at: https://doi.org/10.5281/zenodo.7824714 (Accessed: 02/19/2025).

Anders, I. *et al.* (2023b) *FDO Forum FDO Requirement Specifications*. Available at: https://doi.org/10.5281/zenodo.7782262 (Accessed: 02/19/2025).

Blanchi, C., Gebre, B. and **Wittenburg, P.** (2022) 'Canonical Workflow for Machine Learning Tasks', *Data Intelligence*, 4(2), pp. 173–185. Available at: https://doi.org/10.1162/dint_a_00124

Blumenröhr, N. (2025) *kit-data-manager/DOIP-Client-and-TPM-Adapter- Service-for-FDOs: V 1.0.0*. Available at: https://doi.org/10.5281/zenodo.14886300 (Accessed: 02/18/2025).

Blumenröhr, N. and Aversa, R. (2023) 'From implementation to application: FAIR digital objects for training data composition', *Research Ideas and Outcomes*, 9, e108706. Available at: https://doi.org/10.3897/rio.9.e108706

Blumenröhr, N. *et al.* (2025) "FAIR Digital Objects for the Realization of Globally Aligned Data Spaces". *IEEE International Conference on Big Data (BigData)*. Washington, DC, USA, 14–18 December 2024. IEEE Xplore, pp. 374–383. Available at: https://doi.org/10.1109/BigData62323.2024.10825796

Chen, P.P.-S. (1976) 'The entity-relationship model—toward a unified view of data', *ACM Trans. Database Syst.*, 1(1), pp. 9–36. Available at: https://doi.org/10.1145/320434.320440

Curry, E., Scerri, S. and **Tuikka, T.** (2022) 'Data Spaces: Design, Deployment, and Future Directions', in E. Curry, S. Scerri, and T. Tuikka (eds.) *Data Spaces: Design, Deployment and Future Directions*. Cham: Springer International Publishing, pp. 1–17. Available at: https://doi.org/10.1007/978-3-030-98636-0

DONA Foundation (2018) *Digital Object Interface Protocol Specification*. Available at: https://www.dona.net/sites/default/files/2018-11/DOIPv2Spec_1.pdf (Accessed: 07/28/2021).

European Commission *et al.* (2021) *EOSC interoperability framework: report from the EOSC Executive Board Working Groups FAIR and Architecture*. LU: Publications Office of the European Union. Available at: https://doi.org/10.2777/620649

fairdo (2025) Available at https://gitlab.com/fairdo (Accessed: 02/19/2025).

Islam, S. (2023) 'FAIR digital objects, persistent identifiers and machine actionability', *FAIR Connect*, 1(1), pp. 29–34. Available at: https://doi.org/10.3233/FC-230001

Islam, S. *et al.* (2023) 'Assessing the FAIR Digital Object Framework for Global Biodiversity Research', *Rio*, 9. Available at: https://doi.org/10.3897/rio.9.e108808

Jeffery, K. *et al.* (2021) 'Not Ready for Convergence in Data Infrastructures', *Data Intelligence*, 3(1), pp. 116–135. Available at: https://doi.org/10.1162/dint_a_00084

Kahn, R. and **Wilensky, R.** (2006) 'A framework for distributed digital object services', *International Journal on Digital Libraries*, 6(2), pp. 115–123. Available at: https://doi.org/10.1007/s00799-005-0128-x

Kulüke, M. (2025) *Marco-DKRZ/example-record-typing: auto*. Available at: https://doi.org/10.5281/zenodo.14860533 (Accessed: 02/18/2025).

Lannom, L., Koureas, D. and **Hardisty, A.R.** (2020) 'FAIR Data and Services in Biodiversity Science and Geoscience', *Data Intelligence*, 2(1–2), pp. 122–130. Available at: https://doi.org/10.1162/dint_a_00034

Lannom, L. *et al.* (2022) *FDO Forum FDO Configuration Types*. Available at: https://doi.org/10.5281/zenodo.7825703 (Accessed: 12/13/2024).

Madavarapu, J.B. *et al.* (2024) 'AI-Powered Solutions Advancing UN Sustainable Development Goals: A Case Study in Tackling Humanity's Challenges', in W. Leal Filho *et al.* (eds.) *Digital Technologies to Implement the UN Sustainable Development Goals*. Cham: Springer Nature Switzerland, pp. 47–67. Available at: https://doi.org/10.1007/978-3-031-68427-2_3

Moody, D.L. (1998) 'Metrics for Evaluating the Quality of Entity Relationship Models'. en, *Conceptual Modeling – ER '98, Singapore, 16–19 November, 1998*. Berlin, Heidelberg: Springer, pp. 211–225. Available at: https://doi.org/10.1007/978-3-540-49524-6_18

Pierce, B.C. (2002) *Types and Programming Languages*. 1st ed. Cambridge, Massachusetts: MIT Press.

Schultes, E. and **Wittenburg, P.** (2019) 'FAIR Principles and Digital Objects: Accelerating Convergence on a Data Infrastructure', in A. Pozanenko (ed.) *Data Analytics and Management in Data Intensive Domains*. Moscow: Springer Cham, pp. 3–16. Available at: https://doi.org/10.1007/978-3-030-23584-0_1

Schwardmann, U. (2017) 'Automated schema extraction for PID information types', *Proc. IEEE International Conference on Big Data (Big Data)*. Washongton D.C., USA, 5–8 December 2016: IEEE Xplore, pp. 3036–3044. Available at: https://doi.org/10.1109/BigData.2016.7840957

Smedt, K., Koureas, D. and **Wittenburg, P.** (2020) 'FAIR Digital Objects for Science: From Data Pieces to Actionable Knowledge Units', *Publications*, 8, 21. Available at: https://doi.org/10.3390/publications8020021

Soiland-Reyes, S., Goble, C. and **Groth, P.** (2024) 'Evaluating FAIR Digital Object and Linked Data as distributed object systems', *PeerJ Computer Science*, 10, e1781. Available at: https://doi.org/10.7717/peerj-cs.1781

Weigel, T. *et al.* (2019) *RDA Recommendation on PID Kernel Information*. Available at: https://doi.org/10.15497/rda00031

Wilkinson, M.D. *et al.* (2016) 'The FAIR Guiding Principles for scientific data management and stewardship', *Scientific Data*, 3(1), 160018. Available at: https://doi.org/10.1038/sdata.2016.18

Wittenburg, P. and **Strawn, G.** (2018) *Common Patterns in Revolutionary Infrastructures and Data*. Available at: https://doi.org/10.23728/b2share.4e8ac36c0dd343da81fd9e83e72805a0

TO CITE THIS ARTICLE:
Blumenröhr, N., Böhm, J., Ost, P., Kulüke, M., Wittenburg, P., Blanchi, C., Bingert, S. and Schwardmann, U. (2025) A Comparative Analysis of Modeling Approaches for the Association of FAIR Digital Objects Operations. *Data Science Journal* 24: 22, pp. 1–19. DOI: https://doi.org/10.5334/dsj-2025-022

Submitted: 24 February 2025
Accepted: 06 August 2025
Published: 29 August 2025