Data & Knowledge Engineering 147 (2023) 102210

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

A FAIR catalog of ontology-driven conceptual models

Tiago Prince Sales [a], Pedro Paulo F. Barcelos [a,*], Claudenir M. Fonseca [a]
Isadora Valle Souza [b], Elena Romanenko [b], César Henrique Bernabé [c]
Luiz Olavo Bonino da Silva Santos [a,c], Mattia Fumagalli [b], Joshua Kritz [a]
João Paulo A. Almeida [d], Giancarlo Guizzardi [a]

[a] Semantics, Cybersecurity & Services (SCS), University of Twente, Enschede, The Netherlands
[b] KRDB Research Centre on Knowledge and Data, Free University of Bozen-Bolzano, Bolzano, Italy
[c] Biosemantics Group, Leiden University Medical Centre, Leiden, The Netherlands
[d] Ontology & Conceptual Modeling Research Group (NEMO), Federal University of Espírito Santo, Vitória, Brazil

ARTICLE INFO

Keywords:
Ontology-driven conceptual modeling
OntoUML
Unified Foundational Ontology
Model catalog
FAIR
Linked data

ABSTRACT

Multi-domain model catalogs serve as empirical sources of knowledge and insights about specific domains, about the use of a modeling language's constructs, as well as about the patterns and anti-patterns recurrent in the models of that language crosscutting different domains. They may support domain and language learning, model reuse, knowledge discovery for humans, and reliable automated processing and analysis if built following generally accepted quality requirements for scientific data management. More specifically, not unlike scientific (meta)data, models should be shared according to the FAIR principles (Findability, Accessibility, Interoperability, and Reusability). In this paper, we report on the construction of a FAIR model catalog for Ontology-Driven Conceptual Modeling research, a trending paradigm lying at the intersection of conceptual modeling and ontology engineering in which the Unified Foundational Ontology (UFO) and OntoUML emerged among the most adopted technologies. The catalog, publicly available at https://w3id.org/ontouml-models, currently includes over one hundred and forty models, developed in a variety of contexts and domains.

1. Introduction

Ontology-driven conceptual modeling (ODCM) is a trending paradigm that lies at the intersection of conceptual modeling and ontology engineering. ODCM often involves the use of foundational ontologies to guide the definition of conceptual models, modeling languages, and tools [1]. Within this paradigm, the Unified Foundational Ontology (UFO) [2] and the UFO-based conceptual modeling language OntoUML [3,4] have emerged among the most used approaches [1].

Over the years, UFO and OntoUML have been adopted by research, industrial, and governmental institutions worldwide to create *ontology-driven conceptual models* in a variety of domains [1,2]. These models are created either by directly extending UFO's categories or, more frequently, by using OntoUML (with its stereotypes that reflect the ontological distinctions of UFO). The UFO-grounded models could serve as important resources for reuse and scientific investigation if made openly available and properly curated.

Multi-domain model catalogs serve as sources of empirical knowledge and insights about (i) how specific domains are modeled, (ii) how a modeling language's constructs are used in practice, and (iii) how domain-independent patterns might emerge from language usage. To support domain and language learning, model reuse, and reliable automated processing and analysis by

* Corresponding author.
E-mail address: p.p.favatobarcelos@utwente.nl (P.P.F. Barcelos).

https://doi.org/10.1016/j.datak.2023.102210
Received 15 April 2023; Received in revised form 31 July 2023; Accepted 3 August 2023
Available online 9 August 2023

machines, these repositories must be built following generally accepted quality requirements for scientific data management. In particular, all scientific (meta)data—including models—should follow the FAIR principles, improving their Findability, Accessibility, Interoperability, and Reusability [5].

In this paper, we report on the construction of the FAIR Model Catalog for Ontology-Driven Conceptual Modeling Research, henceforth termed **OntoUML/UFO Catalog**. It is a structured, collaborative, and open-source catalog that contains UFO-grounded models, with the vast majority of them represented in OntoUML. We mainly seek to satisfy two goals with this catalog. First, we want to provide curated structured data to support empirical research in OntoUML and UFO, specifically, and on ontology-driven conceptual modeling, in general. For example, the catalog can provide high-quality data on *why*, *where*, and *how* these approaches are used, which can enable researchers to understand the evolution of the language and its foundations. It can also serve as a repository for the detection of patterns and anti-patterns [6], as well as a benchmark against which language transformation [7,8], complexity management algorithms [9,10], and other modeling techniques can be assessed. Second, it can support novice modelers who want to learn ODCM in OntoUML/UFO, as well as advanced users who want to reuse existing models in their entirety or as *seed models* [11].

The current release of the catalog, 20230414, contains a diverse collection of 141 models obtained from academic and industrial sources, created by modelers with varying modeling skills, for a range of domains, and for different purposes. These models are available in the two machine-readable formats, JavaScript Object Notation (JSON) and Turtle, and are accessible via persistent Uniform Resource Identifiers (URIs).

This paper is an extension of [12], which is evolved in four ways. First, we follow the GO-Plan method [13] to conduct a more systematic and in-depth FAIRness evaluation of the catalog, its models, and the vocabularies used to describe them. Second, we introduce our conceptual data schema—the OntoUML metamodel—and how it is implemented in JSON Schema [14] and in the Web Ontology Language (OWL) [15] to support the serialization of models in JSON and Turtle, respectively. Third, we extend our metadata schema with additional properties to improve the findability and reusability of the catalog and its models. Fourth, we introduce our FAIR Data Point deployment for the catalog, a new data service to facilitate model search and discovery.

This paper is further organized as follows. Section 2 presents the plan we followed to publish the catalog, including the identification of relevant stakeholders, their goals, and information needs. Section 3 discusses the process we have followed and the tools used to collect the models and their metadata. Section 4 reports on our data schema, i.e., the OntoUML metamodel and its implementations in JSON Schema and OWL. Section 5 reports on the metadata schema used for all the resources described in the catalog and exemplifies it with real data. Section 6 presents the two data services through which we store and publish the catalog and its models, namely a Git repository for data storage and a FAIR Data Point for data discovery. Section 7 evaluates the catalog with relation to the FAIR principles. Section 8 discusses some statistics on the current state of the catalog. Section 9 positions our work in relation to other catalogs and datasets available to the modeling community. Finally, Section 10 makes final considerations and discusses future work.

2. FAIRification planning

The FAIR principles emphasize optimizing datasets and other resources for machine discovery, access, interoperability, and reuse with minimal human intervention. The four main principles are further refined in a set of 15 subprinciples [5]. To improve *findability*, the principles prescribe the usage of globally unique and persistent identifiers (principle F1) for resources and their metadata. These resources should be described with rich metadata (F2), which should explicitly include the resource's identifier (F3) and be indexed in searchable platforms (F4). The *accessibility* principle stipulates the use of standardized communication protocols (A1, A1.1) and clear access conditions to resources (A1.2), while recommending high metadata longevity (A2). *Interoperability* is enhanced by publishing resources and their metadata in common knowledge representation formats (I1), using vocabularies that also follow the FAIR principles (I2), and including qualified references to other resources (I3). Finally, FAIR addresses *reusability* by recommending that resources are described with relevant attributes (R1), such as clear licensing (R1.1) and provenance information (R1.2), while following domain-relevant community standards (R1.3).

The process of making a resource FAIR is called FAIRification. This process can vary significantly across projects, organizations, and communities, given that the FAIR principles can be realized in various manners and achieved at different levels. To help us decide the ideal FAIRness level for the resources in our catalog and choose adequate FAIR enabling solutions, we adopted the GO-Plan method proposed by Bernabé et al. [13].

GO-Plan is a goal-based FAIRification planning method that provides a systematic approach to identify, refine, and prioritize FAIRification goals. The method is based on the distinction of two categories of stakeholders, namely project stakeholders and reuse stakeholders. The former refers to those who are involved in the FAIRification project and have their own goals and requirements for it (e.g., data custodians, ethics board members, and funders). The latter refers to those that would reuse the resource to be FAIRified. GO-Plan comprises six phases:

1. FAIRification preparation
2. Assessment of current FAIR supporting infrastructure and target resources
3. Preparation of project collaborators
4. Identification of domain scope and reuse stakeholders
5. Refinement of FAIRification goals and alignment to FAIR principles
6. Decision making

Phase 1: FAIRification preparation. The initial phase of GO-Plan is the onset of a FAIRification project. It consists in identifying key project stakeholders, as well as the resources to be FAIRified, the available FAIR-enabling infrastructure, and additional requirements that needed to be considered (e.g., budget, deadlines, regulatory requirements).

In our FAIRification project, the resources to be FAIRified were OntoUML and UFO-based models, some of which we had access to, but the majority of which we still needed to procure (see more on Section 3). The available infrastructure consisted of scalable virtual servers on which we had the right to install any application or database management system we needed. We did not have external project stakeholders. The only project stakeholder group whose goals we considered was that of the **catalog managers**—those responsible for maintaining the catalog and curating its contents. Their main drivers are maximizing the number of models submitted to the catalog, maximizing the number of cataloged models reused, and minimizing the effort to maintain the catalog.

Phase 2: Assessment of current FAIR supporting infrastructure and target resources. This phase involves verifying if the organization already has a FAIR enabling infrastructure to be reused and its current status, and assessing the resources that will undergo FAIRification. It aims to ensure that both the resources to be made FAIR and the current infrastructure intended to accommodate the FAIR resource do not pose any obstacles to FAIRification. Some FAIRification projects can face legal constraints regarding, for instance, access to privacy-sensitive data or requirements that mandate working with certain proprietary file formats. These obstacles have to be addressed in this phase before FAIRification continues. In our project, no noteworthy obstacles regarding the FAIR-enabling infrastructure or the models were identified.

Phase 3: Preparation of project collaborators. The goal of phase three is to fill eventual knowledge gaps between collaborators of a FAIRification project, such as FAIR and domain experts. GO-Plan recommends that FAIR experts become more knowledgeable about the domain of interest of the project and that domain experts learn a bit about FAIR. This is meant to facilitate communication throughout the project and to align expectations about the project's deliverables.

In our case, the domain of interest was ontology-driven conceptual modeling, with a focus on UFO and OntoUML. All the FAIR experts in our project already had hands-on experience with modeling in OntoUML, so no special activities were performed. This was also the case for the domain experts (i.e. the modeling experts) concerning FAIR.

Phase 4: Identification of domain scope and reuse stakeholders. The fourth phase involves delimiting the domain of the resources that will undergo FAIRification and identifying groups of stakeholders that could potentially reuse them. In our project, the domain was clearly delimited from the beginning—ontology-driven conceptual modeling with UFO and OntoUML.

We identified four groups of reuse stakeholders:

- **Newcomers**: Those who have a basic knowledge of UFO and OntoUML and are seeking to further increase their proficiency. This group includes students, researchers, and practitioners who are undergoing training, studying the language on their own, or working on their first models.
- **Modelers**: Those who want to have a domain of interest formalized via a UFO-based ontology-driven conceptual model. Modelers may be either researchers or practitioners and they may have varying knowledge of OntoUML or UFO. We assume that modelers are competent enough to build and assess models on their own.
- **Tool Developers**: Those who develop any kind of algorithm or supporting tool for UFO and OntoUML, such as:
  - Modeling tools, e.g., the OntoUML Plugin for Visual Paradigm [16] and the OpenPonk Modeling platform [17];
  - Automated transformations of OntoUML models into implementation artefacts, e.g., to relational databases [8], to object-oriented code [7];
  - Complexity management techniques, e.g., model clustering [9] and model abstraction [10];
  - Verification and validation techniques, such as visual model simulation [18] and story-based model assessment [19]; and
  - Modeling assistance techniques [20].
- **Researcher**: Those who are interested in studying how OntoUML is used in practice so that they can improve the language and its application method.

Phase 5.1: Refinement of FAIRification goals. Phase five starts by refining the goals of our stakeholders by asking "how" and "why" questions. For this purpose, GO-Plan recommends the use of goal modeling techniques. Here, we refine the goals of our stakeholders using iStar 2 [21].[1]

The goal refinement diagram for our first group of reuse stakeholders, the **newcomers**, is depicted in Fig. 1. Although there are many strategies they could follow to increase their proficiency in OntoUML, one in which they could leverage the models in our catalog is to examine how others have used OntoUML, both correctly and incorrectly. Observing "good models" is a fruitful way to learn a modeling language. First, they provide examples of how the constructs of the language are meant to be used. For instance, in OntoUML, kinds can be specialized by subkinds and generalized by categories. Second, one will see how common concepts have been formalized by others, such as classifying the type `Person` as a kind or the type `Agent` as a category. Third, good models will probably contain correct applications of the modeling patterns embedded in OntoUML, such as the relator pattern [22] being used

[1] iStar 2.0 defines goals (oval-shaped elements) as states of affairs to be achieved by executing tasks (hexagons). It defines qualities (cloud-like shapes) as desired attributes to which stakeholders want some level of achievement (e.g., performance, easiness), and resources (rectangles) as physical or informational entities to be used in tasks. For further details on iStar, please refer to [21].

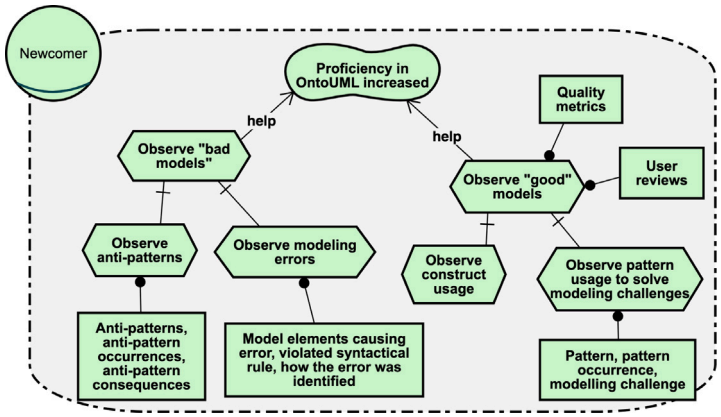


Fig. 1. *iStar* diagram depicting the motivation of newcomers.

to model employment relationships and the role mixing pattern to model `Costumer` as a role that is playable by both people and organizations. Observing "bad models" may be just as educational. In this case, newcomers learn how not to model using OntoUML. If syntactical errors are made explicit, newcomers may improve their understanding of syntactical rules of the language (e.g., that a sortal type cannot specialize multiple kinds). Additionally, by visualizing occurrences of ontological anti-patterns [6], they can learn modeling structures that are likely to misrepresent their domain of interest, either because they lack constraints or because there are too many of them. We highlight that, for this learning strategy to be viable, it is paramount that newcomers can distinguish good from bad models. To do that, they may rely on some quality measures or reviews given by other modelers. Alternatively, they may search for models built by people they consider good modelers.

Our second group of reuse stakeholders, the **modelers**, formalize domains by creating new models, or by reusing and adapting existing ones (as shown in Fig. 2). By reusing a model, we mean taking an existing model as-is or reusing fragments of it by analogy. A model built in such a way is the Digital Platform Ontology [23], which reuses domain-specific patterns from a service ontology entitled UFO-S [24]. By adapting a model, we mean refining an existing model by adding or removing elements until it becomes suitable for the modeler's intent. In this case, an example is the Reference Ontology for Security Engineering [25], which was created by extending the Common Ontology of Value and Risk [26]. By creating a new model, we mean building a model reusing no other model as a basis. Before reusing or extending a model, the modeler first needs to assess whether it is viable. This involves evaluating the license and use conditions set by the authors of the model to be reused. Then, the quality of the model needs to be assessed, which may be done via quality measurements, reviews, or by checking for syntactical errors and anti-patterns.

Additionally, we consider that modelers have a secondary goal of having their models reused and, ultimately, establishing them as references within their communities of interest. To do that, modelers need a platform via which they can share their models, as well as guidance on how to accomplish them effectively. A good example of a resource that became a reference within its community is the GoodRelations ontology [27], a vocabulary for e-commerce that started as an independent project and became very popular when major internet companies adopted it.

A fragment of the iStar diagram for our third reuse stakeholder group, the **tool developers**, is shown on the right side of Fig. 2. It focuses on the evaluation task required to develop algorithms. The purpose and subject of tests vary significantly, but it

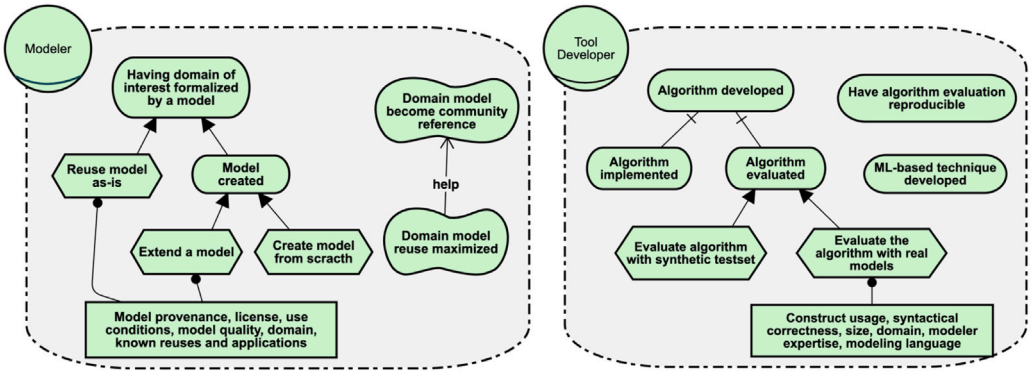


Fig. 2. *iStar* diagram depicting the motivation of modelers and developers.

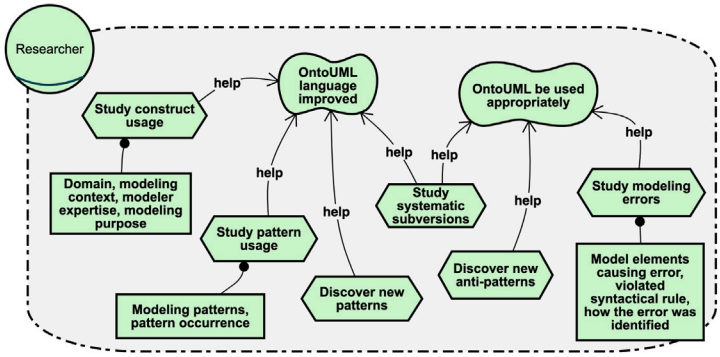


Fig. 3. *iStar* diagram depicting the motivation of OntoUML researchers.

often involves assessing the effectiveness of the algorithm, as well as its performance. To do so, developers need suitable test sets of machine-readable distributions of real models. If no such test set exists, they may rely instead on artificial toy examples or manually build their own test sets by reconstructing models published on the web. However, relying on artificial toy examples may lead to a biased test set that cannot be used to make claims about the algorithm based on empirical evidence. If they do the latter, they will waste a prohibitive amount of time just to prepare the test set. We also consider that developers may want to make the evaluations of their algorithms reproducible, particularly those who want to report their evaluations in scientific publications. For that, the test data must be (i) available for others to retrieve, and (ii) structured in a machine-readable format for others to repeat the evaluation automatically.

The iStar diagram for our fourth and last group of reuse stakeholders, the **researchers**, is shown in Fig. 3. Researchers can derive valuable empirical insights about OntoUML by observing how it has been used over the years across domains and by different people. Previous works have identified ways in which people systematically bent the syntax of the language, triggering its designers to evolve it [4,28]. We call these recurrent grammatically incorrect model fragments *systematic subversions*. OntoUML researchers also want to analyze how often the constructs and patterns of the language are used, so that they can help instructors to improve pedagogical strategies. The results of such analyzes could drive future research efforts that focus on the most used language constructs. Examples of these include [29], which proposes ontology-based rules for designing the concrete syntax of visual modeling languages, and [30], which proposed a canvas for ontology modeling. Both approaches aimed to address subsets of OntoUML/UFO categories. With this catalog, such design choices can be evidence-based. Additionally, researchers may leverage a model catalog as a source of empirical evidence for the discovery of patterns or anti-patterns [6]. Such patterns, which may be domain-independent or domain-specific, may be exploited in the language's evolution and disseminated to contribute to the modeling practice in OntoUML.

Lastly, the goals of our only project stakeholder group, the **catalog managers**, are refined in Fig. 4. Managers seek to maximize submissions to the catalog because, the more models it contains, the more the reuse stakeholders will benefit. Newcomers have more examples to learn from. Modelers have more options to reuse. Developers have larger and more diversified test sets. Researchers have more evidence to support OntoUML research. Additionally, managers seek to maximize model reuse, as this is the most direct metric to measure if it is fulfilling its central purpose—to support the OntoUML community. As this is a voluntary initiative, managers have limited resources to maintain the catalog. Thus, it is of utmost importance that maintenance efforts are minimized. One important strategy toward this aim having a simple submission process that is as automated as possible.

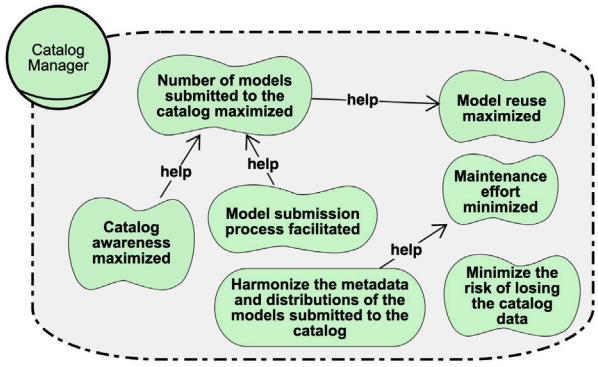
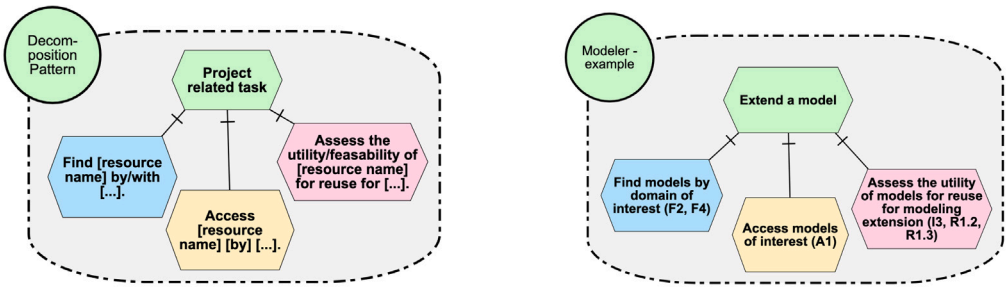


Fig. 4. *iStar* diagram depicting the motivation of catalog managers.



(a) (b)

Fig. 5. IStar diagram illustrating (a) the decomposition pattern and (b) exemplifying its use.

Phase 5.2: Alignment to FAIR principles. Phase five continues with the refinement of goals to actionable tasks and identifying the FAIR principles that are related to these tasks. Here, leaf-level tasks (i.e., those not further refined) are targeted. GO-Plan suggests the use of a decomposition pattern for the identification of related principles. The pattern attaches a set of FAIR-related tasks to the ones previously identified by adding a (i) "finding the resource" task, an (ii) "accessing the resource" task, and an (iii) "assessing the resource's reuse feasibility" task. Then, the principles that would support the realization of these tasks are defined within each one's context. For example, for a goal of improving the awareness of a dataset, a concrete task of providing descriptors for the dataset has been defined to facilitate, for instance, that a newcomer wishing to learn OntoUML via the catalog must *find* the catalog in a search engine (F2, F4), *access* the catalog (A1), and finally *assess* the utility of models for reuse for self-learning.

After identifying and refining the goals related to each stakeholder, we started mapping them to FAIR principles. For this step, we employed the decomposition pattern suggested by the method (depicted in Fig. 5(a)) and applied it to all leaf-level tasks of the previously identified goals. As an example, consider the Modeler's task to "Extend a model". The attached decomposition pattern would include the following tasks, as illustrated in Fig. 5(b):

- "Find models by domain of interest", aligned with principles F2 and F4, which emphasize the importance of rich metadata and indexed search engines to facilitate model discovery.
- "Access models of interest", aligned to principle A1 to allow for metadata to be easily retrieved.
- "Assess the utility of models for reuse for modeling extension", aligned to I3 (extra information to support assessment), R1.2 (clear use conditions for extension), and R1.3 (provenance to support assessing the quality of models).

The complete decomposition of all tasks and the aligned FAIR principles is available in the catalog's documentation on GitHub.[2]

Phase 6: Decision making. Finally, in the sixth phase, implementation decisions are made to realize the principles within the context of the goals and tasks previously identified. For instance, "use W3ID identifiers to achieve F1 in the context of a task to identify the metadata records of the target models".

Having identified the principles related to the goals and tasks of each stakeholder, the next step involved mapping each principle to solution candidates that would support its realization within its designated task context. For instance, to achieve F2 in the example presented in Fig. 5(b), we determined that collecting metadata about the model's title, description, language, violated syntactical rules, and pattern and anti-pattern occurrences would support a newcomer in searching for appropriate models. Additionally, for F4, we specified that this metadata should be indexed by the FAIR Data Point search engine.

Having followed the identification of the FAIRification objectives method, we elaborated a FAIRification plan that is aligned with the catalog's goals. The plan is summarized in Table 1 and further detailed in the remainder of this paper: the development of metadata and data models (F2, F3, I1, I3, R1.1, R1.2, R1.3) is described in Section 4 and Section 5, the publication of the FAIR catalog (F1, F4, A1.1) is detailed in Section 6. The implementation of the FAIRification plan is evaluated in Section 7.

3. Data collection

Given the goals we elicited for our stakeholders, we concluded that two metrics will impact the success of the catalog. The first one is the number of models available in the catalog. The more models it has, the better it supports our reuse stakeholders in achieving their goals. The second metric is the diversity of the models in the catalog, concerning the domain they formalized, the context in which they were developed, and the expertise of those who developed them.

To bootstrap the catalog so that it meets a minimum level of size and diversity, we coordinated a model collection and documentation effort that involved the authors of this paper and members of the OntoUML/UFO community. In this effort, we specified a set of requirements for all model submissions and a review system to evaluate them. These submission requirements establish, for instance, which files must be included (or compiled) on a submission, namely:

[2] https://w3id.org/ontouml-models/git

Table 1
Summary of FAIRification implementation decisions per principle for the catalog and for the models. Principles not listed here were not prioritized.

FAIR principle	Catalog	Models
F1 Unique ID	Catalog to be referenced via a W3ID	Models and their metadata to be referenced via W3IDs
F2 Rich Metadata	Metadata: catalog title, description, landing page, theme, bibliographic citation, license, access rights, and contact point	Metadata: model title, context, domain, description, language, model quality, user rating, (anti)pattern occurrences, syntactical issues, modeling language, modeling purpose, model size, modeler expertise, license, use conditions, known reuses, applications, editorial notes
F3 Linking Data and Metadata	Use W3ID URL's scope	Use W3ID URL's scope
F4 Metadata is searchable	Metadata to be indexed: title, contact points, alternative title, bibliographic citation	Metadata to be indexed: title, domain, language, description
A1.1 Open communication protocol	Metadata to be stored on GitHub and exposed using a FAIR Data Point (FDP)	Data to be stored on GitHub and exposed using an FDP
I1 Knowledge representation language	Data to be made available as RDF.	Models in PNG and Visual Paradigm Project (VPP) (human readable), RDF and JSON (machine-readable). Metadata in RDF.
I3 Qualified references	Metadata to include pointers to other metadata: ORCID (creators, contributors), CC (license), EU vocabularies (access rights)	Metadata to include pointers to other metadata: DBLP and ORCID (authors, contributors), IANA (mediatypes), DOI (original publication).
R1.1 License	License: CC-BY-SA 4.0	License to be described in each model's metadata (defined by the author)
R1.2 Provenance	Metadata: publishers, creators, contributors	Metadata: model provenance (authors, contributors, source)
R1.3 Adherence to community standards	Metadata described using DCAT, DCT, SKOS, MOD, FOAF, VANN, and VCARD	Metadata described using DCAT, DCT, SKOS, MOD, and FOAF. A vocabulary will be created for missing metadata properties.

(i) An image file for each diagram in the model.
(ii) A file containing the model built in a tool that supports OntoUML (e.g., as a Visual Paradigm file using the OntoUML Plugin for Visual Paradigm).
(iii) A serialization of the model in OntoUML's JSON format.
(iv) A serialization of the model in OntoUML's linked data format.
(v) A metadata file in linked data format.

As a visual language, OntoUML represents models through diagrams, which we catalog as image files (i). Unlike diagrams, which are human-readable resources, the serializations of models into standardized (iii) JSON and (iv) linked data files (detailed in Section 4) are machine-readable resources intended to support the needs of tool developers and researchers. Considering existing limitations of the serialization of models into JSON and linked data (e.g., regarding the serialization of formatting and other visual features), the (ii) project file used to develop the model is also required in the submission, enabling the recovery of all original information and promoting full access to modelers intending to reuse a model, and precise documentation to modelers interested in promoting the models they created. Finally, a (v) metadata file (detailed in Section 5) aggregates the provenance and description information elicited as necessary for achieving the relevant FAIR principles targeted in Section 2.

The models included in the catalog's current release were collected from three main sources: from published ontologies referring to UFO/OntoUML publications; from academic and industry projects developed by the involved collaborators; and from classroom assignments submitted by students. The different sources used to build the catalog contributed to the diverse nature of the produced catalog. To highlight how different sources affect submissions, we summarize three general submission paths in Fig. 6.

The first path is the most straightforward one, as it consists of submitting models that have been constructed within an OntoUML-supporting CASE tool. In this scenario, all submission files could be automatically generated, except for the metadata file, which needed to be created manually. The second path consists of submitting models that were built in a CASE tool that does not support OntoUML. In this scenario, models needed to be exported to an interchange format (e.g. XML Metadata Interchange (XMI)) and then imported in an OntoUML-supporting tool. This migration often introduced errors that required fixing before collecting metadata and compiling files. The third path consists of submitting models that were only available as diagram images in the publications we collected. For these models, a manual step of redrawing the diagrams into an adequate tool was necessary. This step was particularly challenging because we wanted to recreate diagrams as close as possible to the sources, including even the layout of elements in the diagram. In these cases, the guidelines provided to the participants of the data collection phase included instructions for handling recurrent issues like incomplete information (e.g., omitted cardinalities). For these models, their original diagrams were also included

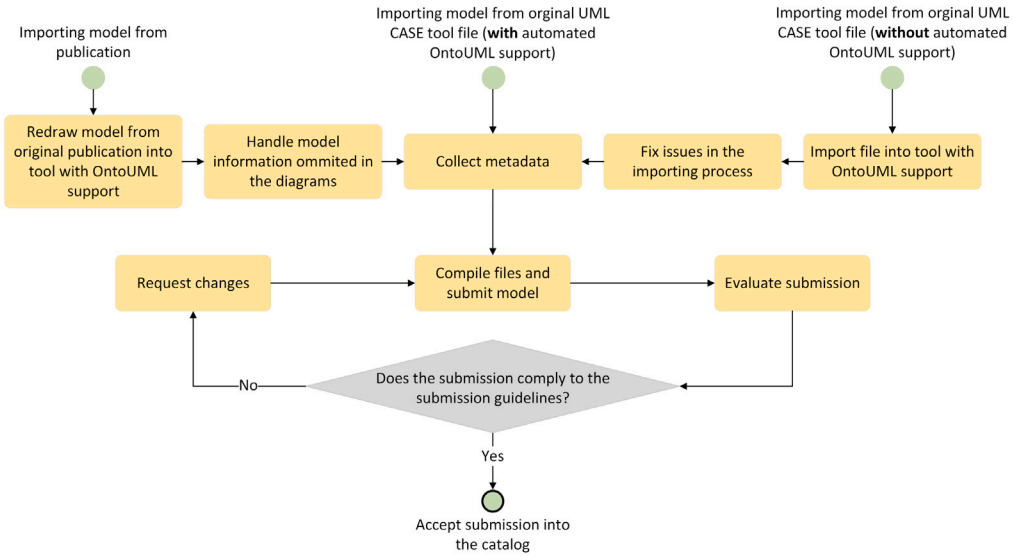


Fig. 6. Model submission and review process.

in the catalog alongside their reproductions, as an additional measure to clarify possible differences from the source material. All models from the data collection phase are included with the acknowledgment of their authors who selected individual licenses.

With all model-related files compiled, including the manually collected metadata, the collaborator would submit the model to the catalog. At this step, additional scripts would generate new complementing metadata files and, if approved by the designated reviewer, integrate the model into the catalog giving it a proper W3ID permanent URL according to the decisions of the FAIRification plan. We further explain the metadata collected, and the scripts/infrastructure we mention here in Section 5 and Section 6, respectively.

4. Data schemas

In Section 3, we describe three different representation formats of OntoUML models we are concerned with in data collection: the model's diagrams as image files, and the model serialization in standardized JSON and linked data OntoUML formats, also referred to as *data schemas* here. Each of these representations answers different needs of the reuse stakeholders, ranging from the needs of human readers such as modelers to applications tools developers and researchers are likely to use when interacting with models. We present on Section 4.1 the metamodel behind the OntoUML language, going into the details and design aspects of the JSON and linked data serializations in Section 4.2 and in Section 4.3, respectively.

4.1. OntoUML metamodel

Originally proposed as an extension of the Unified Modeling Language (UML) [31], OntoUML takes advantage of UML's abstract and concrete syntaxes, which are familiar to a large user base, and the well-established ecosystem of UML CASE tools that allow users to build their models with high-quality diagrammatic support. However, this approach presents some challenges as UML is in fact a family of languages for software engineering with a focus on implementation aspects of object-oriented programming. For instance, the complex metamodel of UML is heavily influenced by the need to support a large variety of diagrams (e.g., use case diagrams, message sequence diagrams), by constructs that are relevant in object-oriented programs (operations, visibility of attributes), and its scope goes far beyond those of a structural ODCM language. As such, we now define the OntoUML Metamodel as an artefact independent of UML and tailored to this ODCM language needs.

Several of the features present in the OntoUML Metamodel are imported from UML's class diagram language but in a much simpler way due to OntoUML's narrower focus. In Fig. 7, we exemplify the OntoUML Metamodel. OntoUML model elements (e.g., classes, relations, generalizations) are organized into modules called packages. For every project, a root package, referred to as `model`, contains all model elements in a tree structure. Alongside the `model` package, the project also contains all diagrams and visual elements, which we omit in this excerpt.

In Fig. 8, we present an example OntoUML model describing it in terms of the metamodel. In this example, we have three classes, `Agent`, `Person`, and `Organization`, which are `contents` of the package `Examples Package`. Two generalization arrows connect `specific` classes `Person` and `Organization` to the `general` class `Agent`, both of them part of the disjoint and complete generalization set `agent_types`, represented as a label next to the generalization arrows. The italicized name of the `Agent` class indicates that it is an abstract class. `Agent` also contains the attribute (i.e., class property) `name`, whose optional cardinality is indicated by the "[*]"

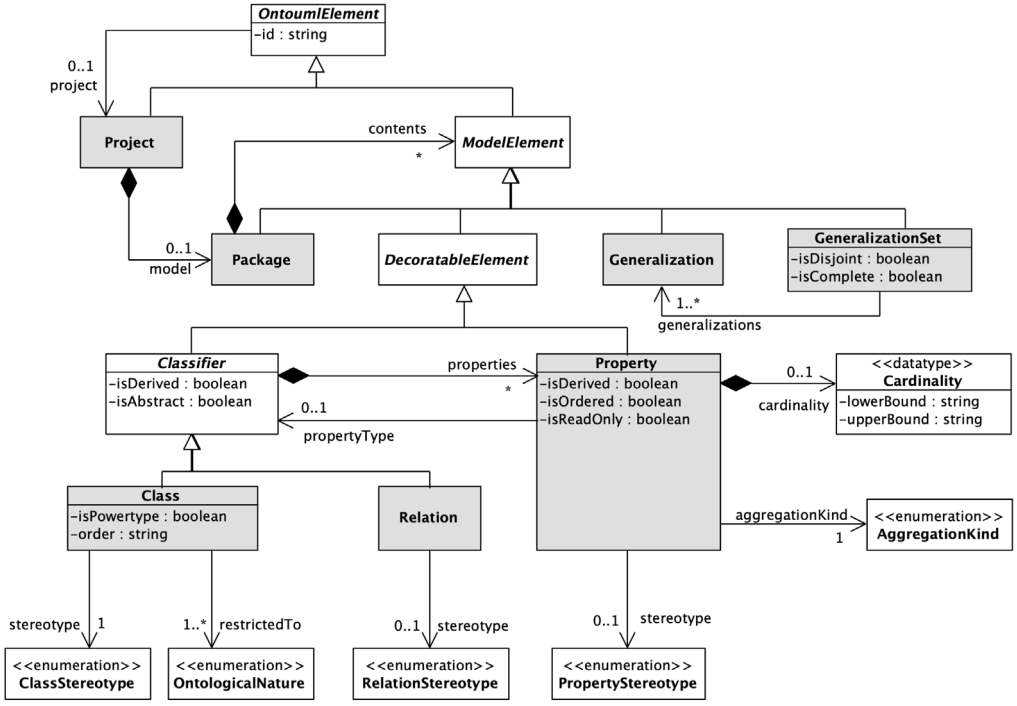


Fig. 7. An excerpt of the OntoUML Metamodel.

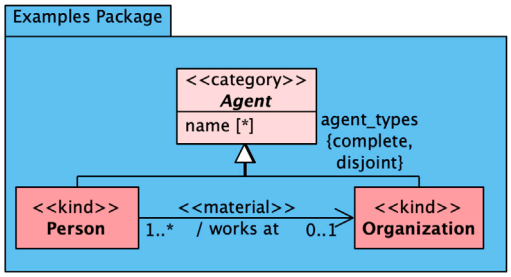


Fig. 8. An example of an OntoUML model.

symbol. The ontological nature of the three classes, all set to "functional-complex", is represented by a UML feature often omitted in diagrams called *tagged value*, which is here represented through the color red (both dark and light shades). The binary relation `works at` is represented as a line connecting `Person` to `Organization`, and it contains two relation end properties, one for the source of the relation (left side) with mandatory cardinality ("1..*"), and the other on the target (right side) with optional cardinality ("0..1"). The `works at` relation is a derived one as indicated by the symbol "/" preceding its name. Finally, all classifiers (i.e., classes and relations) in the example have one stereotype shown as a label between guillemets next to their names. We revisit this example later in Listing 1 and Listing 2 describing its serialization JSON and linked data schemas, respectively.

The metamodel presented here is the reference against which different implementations and syntaxes are developed. In other words, the OntoUML profile for UML (*lightweight extension* mechanism of UML) is now a compatibility layer designed to allow the development of OntoUML models in UML CASE tools. Note that OntoUML models represented in UML do have more leeway than what is intended in this metamodel, but this is exploited when we catalog certain unintended modeling patterns. For instance, certain models contain stereotypes that were never incorporated into the language but that express representation needs from modelers. Others, rather than using stereotypes replicate UFO's categories into their models and specialize them directly. Unintended reuse patterns like this configure some of the goals we foresee being explored by researchers when investigating the use of UFO to identify and differentiate genuine representation needs (also referred to as elephant paths) from anti-patterns and recurrent constraint violations.

```json
{
  "id" : "project_1",
  "name" : "Example Project",
  "type" : "Project",
  "model" : {
    "id" : "package_1",
    "name" : "Examples Package",
    "type" : "Package",
    "contents" : [ {
      "id" : "class_agent",
      "name" : "Agent",
      "stereotype" : "category",
      "isAbstract" : false,
      "restrictedTo" : [ "functional-complex" ],
      "properties" : [ {
        "id" : "att_name",
        "name" : "name",
        "type" : "Property",
        "cardinality" : "*",
        ...
      } ],
      ...
    }, {
      "id" : "gen_person_agent",
      "type" : "Generalization",
      "general" : {
        "id" : "class_agent",
        "type" : "Class"
      },
      "specific" : {
        "id" : "class_person",
        "type" : "Class"
      },
      ...
    }, {
      "id" : "gs_agent_types",
      "name" : "agent_types",
      "type" : "GeneralizationSet",
      "isDisjoint" : true,
      "isComplete" : true,
      "generalizations" : [ {
        "id" : "gen_person_agent",
        "type" : "Generalization"
      }, {
        "id" : "gen_organization_agent",
        "type" : "Generalization"
      } ],
      ...
    }, ... ],
    ...
  }
  ...
}
```

Listing 1: Fragment of the JSON serialization of the model depicted in Fig. 8.

```turtle
@prefix: <https://example.com/model#>.
@prefix ontouml: <https://w3id.org/ontouml#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

:project_1 rdf:type ontouml:Project;
  ontouml:name "Example Project";
  ontouml:model :package_1.


:package_1 rdf:type ontouml:Package;
  ontouml:name "Examples Package";
  ontouml:project :project_1;
  ontouml:containsModelElement
    :class_agent,
    :att_name,
    :gen_person_agent,
    :gs_agent_types .


:class_agent rdf:type ontouml:Class;
  ontouml:name "Agent";
  ontouml:project :project_1;
  ontouml:stereotype ontouml:category;
  ontouml:isAbstract "false"^^xsd:boolean;
  ontouml:restrictedTo ontouml:functionalComplex;
  ontouml:attribute :property_name.

:att_name rdf:type ontouml:Property;
  ontouml:name "name";
  ontouml:project :project_1;
  ontouml:cardinality :cardinality_1.

:cardinality_1 rdf:type ontouml:Cardinality;
  ontouml:cardinalityValue "*".

:gen_person_agent rdf:type ontouml:Generalization;
  ontouml:general :class_agent;
  ontouml:project :project_1;
  ontouml:specific :class_person.

:gs_agent_types rdf:type ontouml:GeneralizationSet;
  ontouml:name "agent_types";
  ontouml:project :project_1;
  ontouml:isComplete "true"^^xsd:boolean;
  ontouml:isDisjoint "true"^^xsd:boolean;
  ontouml:generalization
    :gen_person_agent,
    :gen_organization_agent.


# ...
```

Listing 2: Fragment of the linked data serialization of the model depicted in Fig. 8 in Turtle.

4.2. Models in JSON

Designed to support the development of model intelligence services in OntoUML [16], the ontouml-schema project specifies how to serialize OntoUML in JSON. In this format, OntoUML models can be easily exchanged between clients and servers communicating over HTTP. In addition, JSON has extensive support from all major tech stacks, including easy processing within browser applications. These characteristics favor most uses we foresee from tool developers, being this format the most friendly to the goals in our earlier analysis.

Listing 1 summarizes how the example of Fig. 8 would be serialized according to `ontouml-schema`. As of now, this schema is designed exclusively for the exchange of complete models, where the outermost container is always an object with `"type":"Project"`. In the example, the object representing the project contains an object with `"type":"Package"`, the `"Examples Package"`, as the root for objects corresponding to all other model elements, which are considered `"contents"` of the package (represented with a JSON array). Objects representing model elements can refer to other objects using the `"type"` and `"id"` of the referred object. For example,

the generalization object "gen_person_agent" has references to the general and specific classes ("class_agent" and "class_person" respectively).

In the serialization of the class object "class_agent", "stereotype" is represented as a unique string. The "restrictedTo" reference, represented as *tagged values* in the UML, is a dedicated array of enumerated ontological nature strings (having the nature "functional-complex" in this example). Not unlike UML, properties are commonly used in the representation of both ends in relations and attributes in classes, as these two uses have the same features as a whole (see "attribute_name"), like "cardinality". Finally, a third example of the serialization of model elements is described in "gs_agent_types", the generalization set object named "agent_types" that is also represented in the concrete syntax of OntoUML as simple sets of labels next to the generalizations it comprises. Bear in mind that we omit here many of the more specific properties of each OntoUML element, especially those requiring a greater understanding of UFO.

The specification of ontouml-schema uses a standard called JSON Schema, which allows the validation of JSON objects against specifications guaranteeing they follow the expected structure. This is crucial for this context, as it enables the development of software to process such models by standardizing their representation. Listing 3 demonstrates this through a JavaScript code snippet that validates the shape of a project object against ontouml-schema using a JSON Schema library called Ajv.

```javascript
const schemas = require('ontouml-schema');
const Ajv = require('ajv');

let validator = new Ajv().compile(schemas.getSchema("https://w3id.org/ontouml-schema/2021-02-26/Project"));
let project = {
  id : "project_1",
  name : "Example Project",
  type : "Project",
  model : null,
  diagrams : null
};
let isValid = validator(project);

if (isValid) {
    console.log(`The project ${project.name} is valid!`);
} else {
    console.log(`The project ${project.name} is NOT valid!`);
    console.log(validator.errors);
}
```

Listing 3: Example of ontouml-schema being used in a JavaScript script.

4.3. Models in linked data

While JSON offers a suitable solution for the needs of tool developers, the uses we foresee from researchers require the ability to query and analyze datasets of models. This is even more pronounced in the context of a catalog of models, where the significant number of models enables, for example, the generation of statistical reports and the detection of recurrent patterns. The serialization of OntoUML models in a linked data format allows us to feed them to a knowledge graph and perform complex analysis using the SPARQL querying language, all with no need for additional software. Listing 4 demonstrates this capability in a SPARQL query to retrieve the number of OntoUML classes in each project of the knowledge graph.

Similar to how ontouml-schema is used in the description of the JSON serialization, the OntoUML Vocabulary defines all necessary classes and properties involved in the linked data serialization of models. In the excerpt of Listing 2 we can see how this linked data serialization compares to its JSON counterpart (lst:json-example) for representing the model of Fig. 8. An important difference is that, unlike JSON, there is no notion of object containment/nesting in OWL, so all relations between objects are captured regularly with object properties and URIs. Further, some terms present in the linked data vocabulary may not have an equivalent feature in JSON. This is the case of ontouml:project, a property that connects every model element to the project it is contained in. This property facilitates the construction of common queries, especially within a knowledge graph containing several models. Despite these differences inherent to the context in which each serialization is employed, they are equivalent, with projects referring to a model package which is the root of a tree structure containing all model elements (e.g., classes, relations, generalizations, generalization sets). Even if omitted here, the linked data serialization also comprises the concrete elements (diagrams and diagram elements) of OntoUML projects and can also be used to fully reconstruct them.

```sparql
PREFIX ontouml: <https://w3id.org/ontouml#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?proj (COUNT (?c) AS ?classes)
WHERE {
  ?c rdf:type ontouml:Class .
  ?c ontouml:project ?proj .
}
GROUP BY ?proj
```

Listing 4: Example of a SPARQL to a knowledge graphs of models serialized with the OntoUML Metamodel Vocabulary.

5. Metadata schema

We designed the metadata schema for the catalog to satisfy the information needs of our stakeholders by reusing classes and properties from the following RDF/OWL vocabularies:

- Data Catalog Vocabulary (DCAT) - Version 2: A vocabulary that supports the representation of metadata about catalogs, datasets, and their distributions. DCAT is a W3C recommendation since 2014 and is frequently used in FAIRification initiatives.
- Dublin Core Terms (DCT): A vocabulary that defines properties to describe basic metadata about web resources. It is extensively reused by DCAT and is also commonly adopted in FAIRification initiatives.
- Metadata for Ontology Description and Publication (MOD) - Version 2: A vocabulary that extends DCAT by defining classes and properties specifically for the representation of metadata about ontologies and other semantic artefacts.
- Simple Knowledge Organization System (SKOS): A vocabulary for representing and linking knowledge organization systems. SKOS is also a W3R recommendation and is widely used by the semantic web community.
- Friend of a Friend (FOAF): A vocabulary that offers terms to describe people, groups, companies, and other types of agents.
- vCard: A vocabulary to describe contact information (e.g., email, phone number).
- OntoUML/UFO Catalog Metadata Vocabulary (OCMV) [32]: Our custom metadata vocabulary that specifies properties and property values we needed to declare the metadata of our catalog and of its models but that were not available in the other vocabularies.

The backbone of our metadata schema, depicted in Fig. 9, consists of five classes, namely `dcat:Resource`, `dcat:Dataset`, `dcat:Distribution`, `dcat:Catalog`, and `mod:SemanticArtefact`. We consider as a `dcat:Resource` any object that we keep track of in the catalog (including the catalog itself). Every resource should be described with basic metadata properties to improve its findability and discoverability, which include its names (via `dct:title` and `dct:alternative`) and descriptions. To support reusability, resources should be described with provenance data, which includes when they were created and modified (`dct:issued` and `dct:modified`), who created and modified it (`dct:contributor`, `dct:creator`, `dct:publisher`), and under which conditions it can be accessed and reused (`dct:accessRights`, `dct:license`).

When referring to an external resource, such as the author of a model or the license attributed to one, we recommend the use of a globally unique, persistent, and resolvable identifier. Whenever possible, we refer to the creators and contributors of a model (instances of `foaf:Agent`) via a URI provided by DBLP (e.g., https://dblp.org/pid/96/8280) or ORCID (e.g., https://orcid.

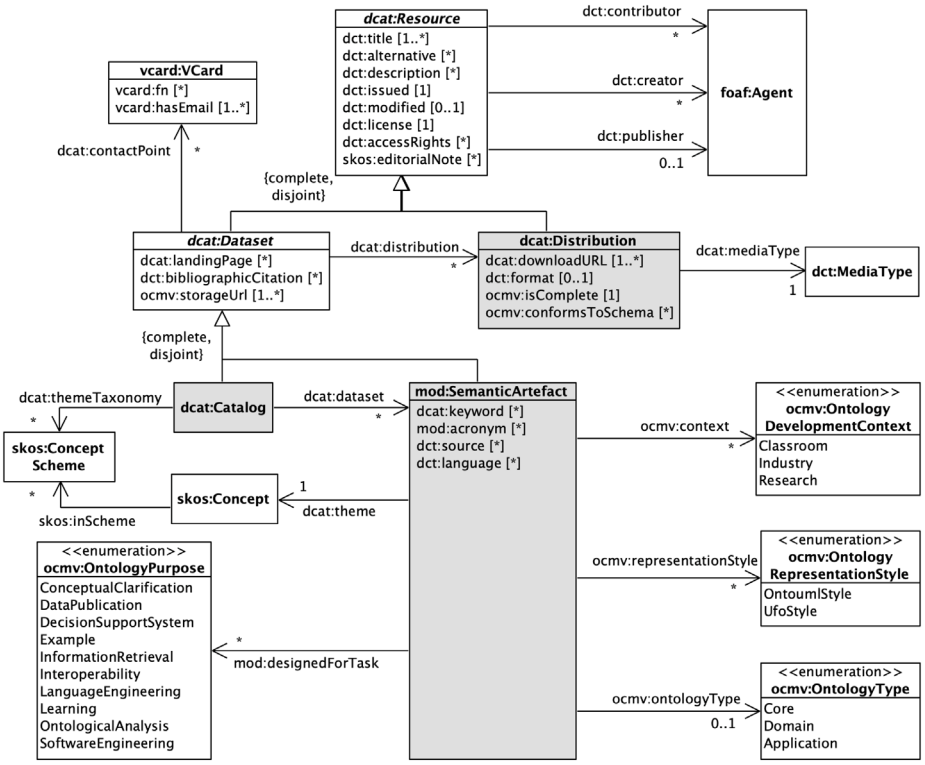


Fig. 9. The metadata schema for the OntoUML/UFO Catalog (`dcat:Catalog`), the conceptual models (`mod:SemanticArtefact`), and their distributions (`dcat:Distribution`) described as a UML class diagram.

org/0000-0003-2528-3118). If there is no such a URI, we suggest the use of a URL that references the agent's profile in a digital platform (e.g., https://github.com/tgoprince) or a webpage about the agent (e.g., http://www.mattspace.net).

A particular type of `dct:Resource` of interest to us is the `dcat:Dataset`. A dataset comprises a collection of data items and is available for access or download in one or more distributions (via `dct:distribution`). To foster findability and accessibility, datasets are associated with URLs pointing to web pages where one can find more information about the dataset (`dcat:landingPage`) and where one can access a data storage service to retrieve the dataset, its metadata, and its distributions (`ocmv:storageUrl`). We describe channels to reach out to the maintainers of a dataset with a `vcard` (via `dcat:contactPoint`). Lastly, to stimulate proper citation of the datasets, we added the `dct:bibliographicCitation` property.

A `dcat:Catalog` is a curated `dcat:Dataset` consisting of metadata about resources. Here, we naturally focus on a single catalog instance, the OntoUML/UFO Catalog, which contains metadata about conceptual models and their distributions. A fragment of its metadata record is shown in Listing 5. In addition to the properties defined for resources and datasets, a catalog lists the datasets (models in our case) it contains via the `dcat:dataset` property, as well as the knowledge organization system used to categorize them. To harmonize this categorization in our catalog, we opted to use the Library of Congress Classification (LCC) system, which has been published as a `skos:ConceptScheme`.

```
<https://w3id.org/ontouml-models> a dcat:Catalog ;
    dct:title "FAIR Model Catalog for Ontology-Driven Conceptual Modeling Research"@en ;
    dct:alternative "OntoUML/UFO Catalog"@en ;
    dcat:themeTaxonomy <http://id.loc.gov/authorities/classification> ;
    dct:license <https://creativecommons.org/licenses/by-sa/4.0/> ;
    dct:accessRights <http://publications.europa.eu/resource/authority/access-right/PUBLIC> ;
    dct:issued "2019-06-18"^^xsd:date ;
    dct:modified "2023-03-15"^^xsd:date ;
    ocmv:storageUrl "https://github.com/OntoUML/ontouml-models"^^xsd:anyURI ;
    dct:publisher <https://orcid.org/0000-0003-2528-3118> ;
    dct:creator <https://orcid.org/0000-0002-5385-5761> ,
                <https://orcid.org/0000-0002-1164-1351> ;
    dct:contributor <https://orcid.org/0000-0002-2384-3081> ,
                    <https://orcid.org/0000-0001-5010-3081> ;
    dcat:dataset <https://w3id.org/ontouml-models/model/5f85d9aa-67ac-4315-b60d-0212ea910320> ,
                 <https://w3id.org/ontouml-models/model/c6d2e61a-4904-47ec-8935-2e2e56669220> .
```

Listing 5: A fragment of the metadata describing the OntoUML/UFO Catalog.

The models in our catalog are categorized as instances of `mod:SemanticArtefact`, a type of `dcat:Dataset` whose data items are the formalization of concepts, properties, relations, and constraints about a domain of interest. The metadata record of a semantic artefact is exemplified in Listing 6 using the Reference Ontology of Trust [33]. To support our stakeholders in selecting and assessing models for reuse, we describe a model with the following properties: the domain it formalizes (`dcat:theme` and `dcat:keyword`), the purpose for which it was created (`mod:designedForTask`), the context in which it was created (`ocmv:context`), whether it is represented in OntoUML or extends UFO (`ocmv:representationStyle`), whether it is a core, domain, or application ontology model (`ocmv:ontologyType`). We also provide additional provenance to our models by connecting them with the publications in which they are described (`dct:source`). Here, we once more recommend the use of globally unique, resolvable, and persistent identifiers. For this type of resource, we use DOIs (e.g., https://doi.org/10.1007/978-3-030-33246-4_1).

```
<https://w3id.org/ontouml-models/model/d88fe48c-d574-43b4-85d6-a6e1aeaa6726> a dcat:Dataset, mod:SemanticArtefact ;
    dct:title "Reference Ontology of Trust" ;
    mod:acronym "ROT" ;
    dcat:keyword "trust"@en ;
    dcat:landingPage <https://github.com/unibz-core/trust-ontology> ;
    ocmv:ontologyType ocmv:Domain .
    ocmv:representationStyle ocmv:OntoumlStyle ;
    mod:designedForTask ocmv:ConceptualClarification ;
    ocmv:context ocmv:Research ;
    dct:issued "2019"^^xsd:gYear ;
    dct:modified "2022"^^xsd:gYear ;
    dcat:theme lcc:H ;
    dct:language "en" ;
    dct:license <https://creativecommons.org/licenses/by/4.0/> ;
    skos:editorialNote "Files retrieved from ROT'ss GitHub repository on 23 March 2022." ;
    dct:contributor <https://dblp.org/pid/81/4277>,
                    <https://dblp.org/pid/11/78>,
                    <https://dblp.org/pid/33/8219>,
                    <https://dblp.org/pid/75/5043> ;
    dct:source <https://doi.org/10.1007/978-3-030-33246-4_1> ,
               <https://doi.org/10.1007/978-3-030-62522-1_25> ;
    dcat:distribution
            <https://w3id.org/ontouml-models/distribution/78713112-cf7e-45f1-8c74-c7b5906f5b7c/>,
            <https://w3id.org/ontouml-models/distribution/7c83f03b-c170-49d2-9dd9-0a600be6cc96/>,
            <https://w3id.org/ontouml-models/distribution/669a3a1a-fd31-436e-8f31-932b4119de47> .
```

Listing 6: A fragment of the metadata describing the Reference Ontology of Trust [33].

A `dcat:Distribution` is a `dcat:Resource` that materializes a `dcat:Dataset`. The metadata of a distribution is exemplified in Listing 7. Noteworthy properties are `dcat:mediaType`, which identifies the format of the distribution (e.g. https://www.iana.org/assignments/media-types/application/json); `dcat:downloadURL`, which provides the address where to obtain the distribution; and `ocmv:isComplete`, a boolean property that identifies whether or not a distribution completely materializes a dataset. In our catalog, we use the latter property to state that the VPP, JSON, and Turtle distributions of a model are complete materializations, while the diagrams in PNG are not.

```
<https://w3id.org/ontouml-models/distribution/7c83f03b-c170-49d2-9dd9-0a600be6cc96/> a dcat:Distribution;
    dct:title "JSON distribution of the Reference Ontology of Trust"@en ;
    dct:license <https://creativecommons.org/licenses/by/4.0/> ;
    dcat:mediaType <https://www.iana.org/assignments/media-types/application/json> ;
    ocmv:isComplete "true"^^xsd:boolean ;
    ocmv:conformsToSchema <https://w3id.org/ontouml/schema> ;
    dcat:downloadURL <https://raw.githubusercontent.com/OntoUML/ontouml-models/master/models/amaral2019rot/ontology.json> .
```

Listing 7: A fragment of the metadata describing the JSON distribution of the Reference Ontology of Trust [33].

A detailed specification of the properties in our metadata schema and the guidelines on how to use them are provided in Appendix A.

6. Data services

In order to fulfill the goals of our reuse stakeholders, an adequate FAIR-supporting infrastructure is necessary to enable the principles of findability, accessibility, and reusability. Modelers, in particular, require features beyond reuse that enable the addition, update, and deletion of cataloged models and their distributions. We enable these capabilities on the OntoUML/UFO Catalog by employing tailored software platforms for the storage and the discovery of models which we refer to as *data services*. In this section, we discuss how FAIR Data Points [34] and online Git repositories are used as *data discovery* (Section 6.1) and *data storage* (Section 6.2) services, respectively, and how the features they provide help us fulfill our reuse stakeholders' goals.

6.1. Data discovery service

The OntoUML FAIR Data Point is a deployment of the FAIR Data Point (FDP) reference implementation [34], which is a FAIR-compliant platform designed to expose semantically rich metadata of FAIR digital objects. For the end-user (i.e., the reuse stakeholder), the OntoUML FAIR Data Point presents a website that turns every model and distribution (and even the catalog itself) into a dynamically generated website that shows all collected metadata in human-readable form. Moreover, all information contained in their metadata can be used for information discovery. In the example of Fig. 10, we present the filtered search for models with the context "industry" and designed for "conceptual clarification". This filtered search is also complemented with support for text-based search and SPARQL queries. For software agents, the endpoint of every resource is capable of proving the displayed information in linked data format (i.e., Turtle, RDF/XML, and JSON-LD). Finally, this server provides the URIs of all resources in our catalog.

However, the reference FAIR Data Point we employ has an important limitation as it is designed to manage only metadata. In addition, it is also limited when it comes to versioning the information it displays. To complement the capabilities of this data discovery service we employ a data storage service which we describe in Section 6.2.

6.2. Data storage service

To make the data (and metadata) available through our data discovery service accessible, we employ an online Git repository which serves as our data storage service. Git is a distributed version control system designed to support developers to collaborate and maintain large projects. With the help of online platforms, such as GitHub, these repositories become a way to share projects and enable collaboration between internal and external members, being highly adopted by users beyond the software engineering community.

The ontouml-models repository, presented in Fig. 11, was originally conceived as an online Git repository before the FAIRification process. The pull request mechanism of the used platform allowed for the collaboration required for modelers to submit and update models and distributions, with the review of each submission by a catalog manager. As every change in the repository is a commit in Git, the demand for the catalog's versioning is trivially resolved, with its whole history tracked. Individual commits can also be easily published and version releases of the catalog.

However, platforms such as GitHub cannot take advantage of the metadata collected in the FAIRification process, being this an unfit solution for many reuse stakeholders. Thus, the setup using two complementing services is still required. To solve the synchronization needs between the two services, a synchronization script is used, and improvements in the submission mechanism are among our future works. The document structure used in the current release of the OntoUML/UFO Catalog is presented in detail in Appendix B.

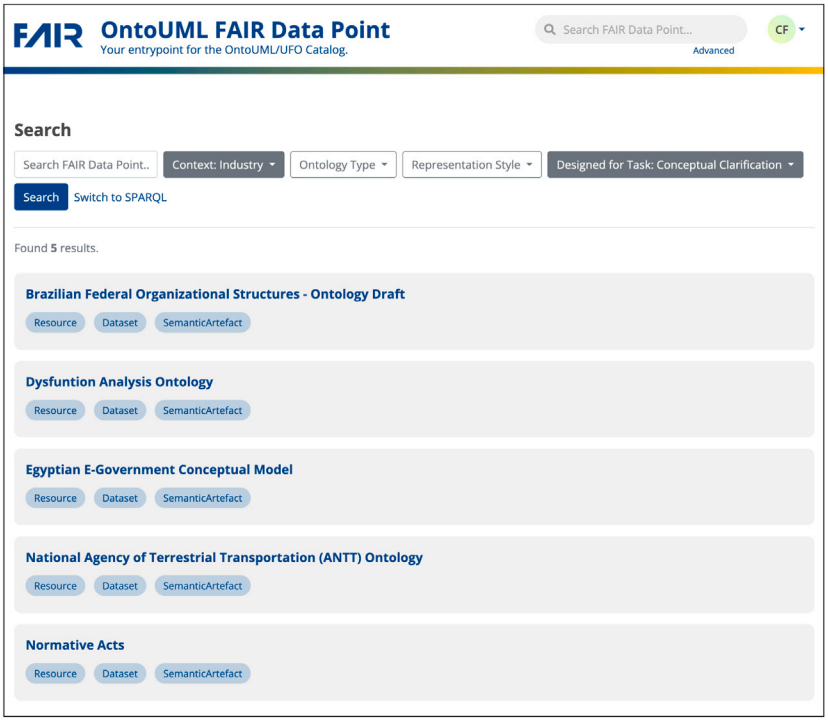


Fig. 10. OntoUML FAIR Data Point filtered search example.

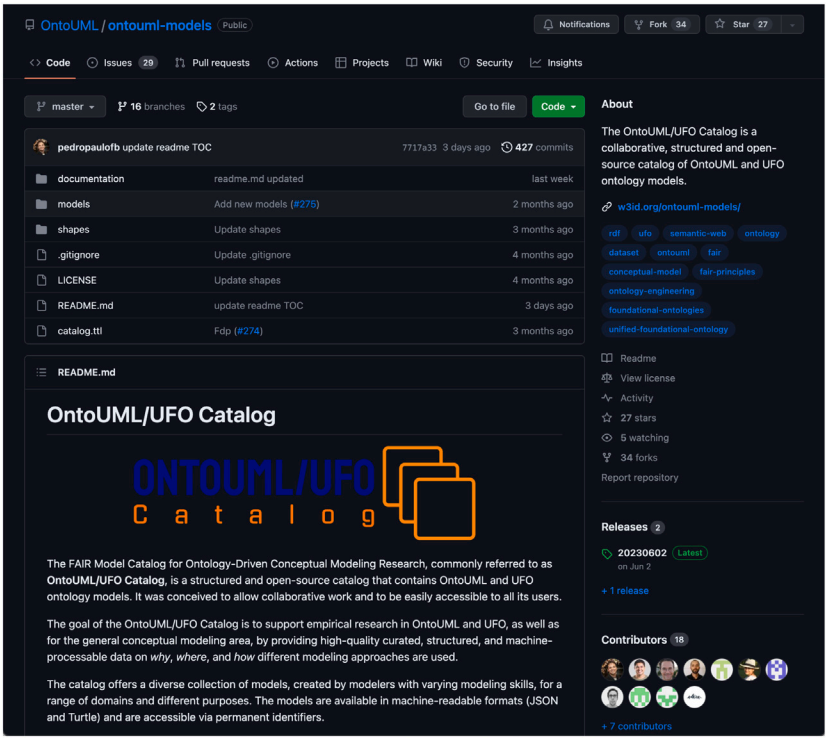


Fig. 11. The repository page of the OntoUML/UFO Catalog on GitHub.

7. FAIRness evaluation

As a result of the designed architecture and (meta)data models, and the reuse of standards, both the catalog and the data it provides meet a number of FAIR principles. Tables 2 and 3 summarize the evaluation of the catalog and the models' FAIRness, respectively.

7.1. Catalog

Findability The catalog can be reached through a global, unique, persistent, and resolvable identifier provided by the W3ID, thus fully achieving FAIR principle F1. Since the definition of what is "rich metadata" is not trivial, we aimed to fulfill the FAIR principle F2 by providing as much metadata as possible to facilitate the goals and tasks for the groups of stakeholders identified during FAIRification planning. For principle F3, it is possible to decompose the metadata URI and get back to the resource (i.e., the catalog) it describes. The provided metadata, which is exposed by using an FDP, is indexed using the FDP index service, thus fulfilling principle F4.
Accessibility The FDP provides an API that supports data exchange via HTTP, as well as a SPARQL endpoint, both of which are open communication protocols and fulfill A1.1. Given the goals of the catalog, no authentication protocol is necessary (i.e., A1.2 is not applicable). While the host of the FDP instance currently does not have a persistence plan in place, the metadata is also published on the catalog's GitHub repository, partially fulfilling A2.
Interoperability The catalog's metadata is published in RDF, a formal, accessible, shared, and broadly applied format, consequently realizing I1. Furthermore, the catalog's use of other FAIR vocabularies fulfills principle I2. These vocabularies provide additional metadata that enrich the description of the terms defined by them, thereby realizing I3: ORCID provides extra information about the author or collaborator, the EU vocabularies service offers detailed description about access rights, while CC provides additional metadata about the catalog's license.
Reusability The catalog is made available under the CC-BY-SA 4.0 license, as stated in its metadata (R1.1). Provenance information is also included in the metadata, with the related terms defined based on the goals of the various stakeholders (R1.2). Moreover, the catalog's metadata has been designed with a focus on reusing the most relevant and widely adopted standards, thus fulfilling the third aspect of the Reusability principle (R1.3).

Table 2
FAIRness evaluation of the catalog.

FAIR principle achieved	Reasoning
F1 Unique ID	W3ID service offer global, unique, persistent, resolvable identifiers
F2 Rich Metadata	Metadata concepts defined to fulfill stakeholder's goals (our interpretation of rich metadata)
F3 Linking Data and Metadata	W3ID scope creates a path from metadata to data
F4 Metadata is searchable	The metadata in FDP is indexed by other search engines (i.e., Bing)
A1.1 Open Communication Protocol	FDP will allow communication using HTTP and SPARQL protocols, which are open, free, and universally implementable
A1.2 Authentication Protocol	Not applicable
A2 Metadata persistence	Metadata made persistent on the GitHub repo
I1 Knowledge representation language	Catalog metadata is exposed using RDF, which is formal, accessible, shared, and broadly applied
I2 Controlled vocabularies	Catalog metadata uses other vocabularies (e.g., ORCID, EU access right) that achieve some FAIRness
I3 Qualified references	Catalog metadata references services (ORCID, EU vocabularies, CC) that offer extra metadata for the referenced terms
R1.1 License	Catalog's license is defined in a machine-readable manner (RDF, CC terms)
R1.2 Provenance	Provenance information provided (in machine-readable formats) to fulfill stakeholder's goals
R1.3 Adherence to community standards	The catalog's metadata was designed using terms from, e.g., DCAT, DCT, SKOS, and FOAF, which are domain-relevant community standard standards for defining catalogs and datasets [34]

7.2. Models

The catalog's data includes the models themselves (in RDF, JSON, VPP, and PNG formats), as well as their specific metadata. These models are organized based on the OntoUML metadata model, which is beyond the scope of this evaluation.

Findability The models and their specific metadata are referred to using W3ID URIs that fall under the catalog's URI scope, thus primarily achieving F1 and consequently partially achieving F3, since the path to data can be decomposed to find its metadata. The metadata about the models includes relevant terms identified to achieve the stakeholders' goals, thereby fulfilling F2. Lastly, the catalog's metadata is made searchable through the FDP's search engine, which realizes F4.
Accessibility The metadata for the models is exposed through the FDP, which offers both HTTP and SPARQL endpoints for accessing them, fulfilling A1.1. Similarly, the models themselves are available on GitHub, which provides HTTP-based access to them. Since the data is open, A1.2 was not prioritized in this work. The data and metadata are under the persistency capabilities of GitHub, which partially achieves A2.
Interoperability The metadata for the models, as well as the models themselves, are made available in RDF, JSON, PNG, and VPP, thus achieving I1. The metadata about the models is described using other controlled vocabularies (I3) that are themselves FAIR (I2), including ORCID and DPL for model authorship, IANA for the distribution formats, EU vocabularies for access rights, and CC for licensing. Other specific terms are defined in the catalog's own controlled vocabulary to achieve greater consistency and interoperability, thus reinforcing I3.
Reusability Models submitted to the catalog are required to have their license specified (R1.1), and their provenance terms are collected to meet the stakeholders' goals defined during planning (R1.2). Finally, the metadata of the models was defined by reusing relevant standards: DCAT, DCT, SKOS, FOAF, VANN, and VCARD, while data is structured following the OntoUML metadata model (R1.3).

Table 3
FAIRness evaluation of the models.

FAIR principle achieved	Reasoning
F1 Unique ID	W3ID service offers global, unique, persistent, resolvable identifiers
F2 Rich Metadata	Metadata concepts defined to fulfill stakeholder's goals (our interpretation of rich metadata)
F3 Linking Data and Metadata	W3ID scope creates a path from metadata to the model instances
F4 Metadata is searchable	Models metadata indexed by the FDP search engine
A1.1 Open Communication Protocol	The implemented FDP allows communication using HTTP and SPARQL protocols, which are open, free, and universally implementable
A1.2 Authentication Protocol	Not applicable
A2 Metadata persistence	Persistence provided by GitHub
I1 Knowledge representation language	Models published in RDF, and JSON (formal, accessible, shared, and broadly applied machine-readable formats)
I2 Controlled vocabularies	Metadata uses other vocabularies (e.g., ORCID, EU access rights) that achieve some FAIRness. Controlled vocabularies are defined in machine-readable formats for terms not mapped to external ones.
I3 Qualified references	Metadata references services (ORCID, IANA, EU vocabularies, CC) that offer extra metadata for the referenced terms
R1.1 License	License is defined by the authors of the models, CC enforced.
R1.2 Provenance	Provenance information provided (in machine-readable formats) to fulfill stakeholder's goals
R1.3 Adherence to community standards	Metadata designed using terms from DCAT, DCT, SKOS, and FOAF, which are domain-relevant community standards for defining catalogs and datasets. The OntoUML (domain) metadata model reuses concepts from the OWL's metamodel.

8. Catalog statistics

The current release of the catalog contains 141 models. In this section, we briefly describe their composition and metadata. As we shall see, the results evidence that the catalog captures a diverse and heterogeneous sample of models, making it a valuable resource for supporting various analytical investigations.

8.1. Statistics on the data

Altogether, the 141 models in the catalog contain 733 diagrams, 7864 classes, 5987 relations, and 5839 generalizations. Among them, three outliers stand out for being significantly larger than the others, namely the National Agency of Terrestrial Transportation (ANTT) Ontology [35], the Ontology of Technology-Independent Multi-layer Transport Networks [36], and the Digital Platform Ontology [23]. By themselves, these three models contain 40% of all diagrams, 25% of all classes, 22% of all relations, and 34% of all generalizations. In the remainder of this subsection, we seek to characterize, from a structural perspective, the type of model one may expect to find in our catalog. Thus, we remove these three outliers and focus on the remainder 138 models.

We start with some basic statistics on the number of diagrams, classes, relations, and generalizations listed in Table 4. Given the wide range between the variable's minimum and maximum values, the catalog has quite a variety of models. However, it is limited with respect to models with higher numbers of diagrams, classes, relations, and/or generalizations, since the variables' mean values are much lower than their maximum values. Moreover, the fact that the median values of the variables are close to their mean values shows that the sample has a concentration of models with data close to the mean value and far from the maximum and

Table 4
Statistics on concepts from the catalog ontologies.

	Diagrams	Classes	Relations	Generalizations
Minimum	1	7	0	0
Maximum	20	190	165	155
Median	2.0	33.5	26.0	18.0
Mean	3.2	42.7	33.9	28.0
Mode	1	18	22	14

minimum values. Finally, it is possible to find in the catalog only 1 taxonomy, which is a model with no relations, and 6 models with no generalizations.

In the histogram depicted in Fig. 12(a), we group models by their number of classes. It shows that most (72) contain between 11 and 35 classes, a considerable number (45) contain between 36 and 70 classes and that very few (17) contain more than 70 classes. To better understand this distribution, we investigated which probability distribution best fitted it by applying goodness-of-fit techniques [37]. Using the Minitab Goodness-of-fit Test, we tested 14 probability distributions and 2 transformations. The results showed that the best fit was the lognormal distribution, with a p-value of 0.688.[3] We complement this assessment with the probability plot depicted in Fig. 12(b). In this type of plot, if the data distribution is a straight line (as in this case), it is reasonable to assume that the observed statistical sample comes from the specified distribution [38]. Continuing our analysis, we used Minitab to determine the parameters that specify the shape and location of the defined distribution. We identified that the data on the class number followed a lognormal distribution with a location of 3.519 and a scale of 0.675. Finally, we plotted the Histogram shown in Fig. 12(a) that compares the data grouped into intervals of the number of classes according to the frequency of observations (blue bars) to the probability density function of a lognormal distribution with the parameters determined (red line). The figure confirms a good fit between the analyzed data and the selected distribution.

A lognormal distribution is a continuous probability distribution in which the logarithm of a random variable is normally distributed. If a data set fits a lognormal distribution, it means that when the logarithm of all the data points is taken, the newly transformed points fit a normal distribution [39]. It is visually characterized by a long tail toward the right and a peak to the left. By following the lognormal distribution, our sample inherits its properties. Therefore, in the sample analyzed, there are no models with no classes or with a negative number of classes, and each model is an independent variable and independent regarding the number of classes. This means that if you randomly choose a model from the sample, you cannot infer anything about the number of classes in it. This could be attributed to the diversity of the models, which makes it unlikely for a common set of classes to exist between them.

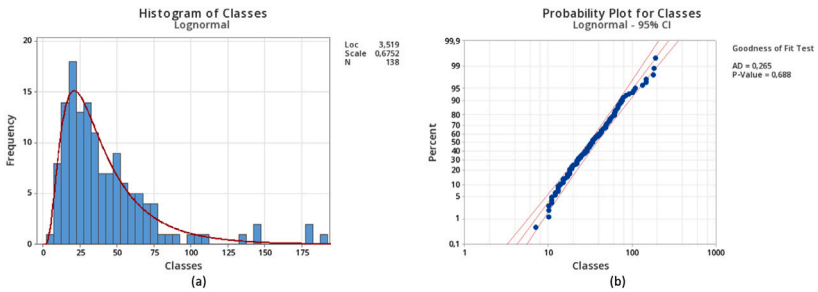


Fig. 12. (a) Histogram of classes and (b) probability plot for classes.

Deepening our analysis, we used Minitab to elaborate the matrix plots on Fig. 13 and calculate the Pearson's Correlation Coefficient (r) for the correlation between the number of (a) diagrams, (b) relations, and (c) generalizations of the models and their number of classes. The results show that there are positive and strong correlations between the variables [40], with $r = 0.764$, $r = 0.806$, and $r = 0.935$, respectively. Although the graphics give us some insightful information on the models' data, they should be carefully analyzed since the sample is limited.

By analyzing the graphics on Fig. 13, we observed that the ratios between the number of classes and the number of diagrams, relations, and generalizations vary greatly among models. For instance, the Ontology of Board Games [41] has 179 classes but only 4 diagrams (roughly 45 classes per diagram), while the Online Education Ontology [42] contains 65 classes and 12 diagrams (a little over 5 classes per diagram). The same phenomena can be observed when comparing the number of classes and relations. On one end of the spectrum, there is OntoBio [43], which has 147 classes and 43 relations, hence, 3.4 classes per relation. On the other end, we have models like the Turbidity Channel Migration Ontology [44], which has 69 classes and 99 relations, and hence, only 0.7 classes

[3] The Minitab Goodness-of-fit Test also showed that the Box–Cox Transformation (p-value = 0.688) and the Johnson Transformation (p-value = 0.890) have high p-values which indicate that we can successfully use these transformations if we need to transform our data to follow the normal distribution.

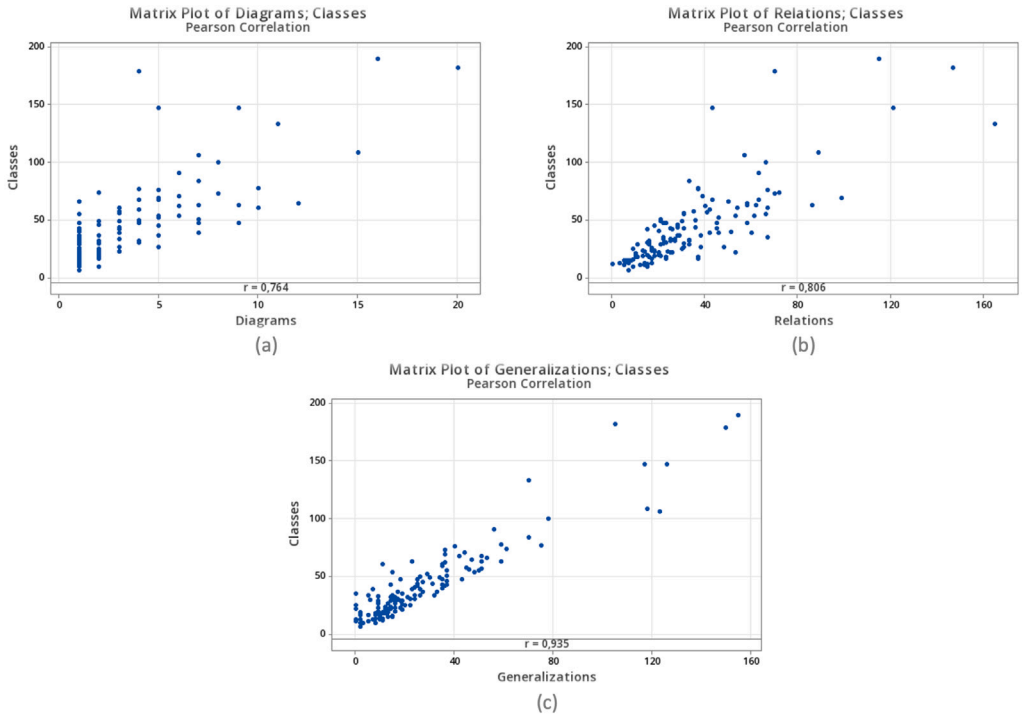


Fig. 13. Correlation between the number of classes and the number of (a) diagrams, (b) relations, and (c) generalizations.

per relation. The Electrocardiogram (ECG) Ontology [45] stood out during the analysis of the data on the number of generalizations for having 133 classes and 70 generalizations, which is considerably lower than the average number of generalizations (127.71) in models with more than 100 classes.

Lastly, we analyzed the use of OntoUML class stereotypes. We used the data from the 5426 classes of the 125 models described as having `ocmv:OntoumlStyle` as one of their representation styles (via the metadata property `ocmv:representationStyle`). The result is presented in Fig. 14 and shows that the most frequently used stereotypes are Kind (18.67%), Subkind (18.45%), Role (12.15%), Relator (8.29%), Category (6.89%), and Event (5.77%). These six classifications represent more than half of the stereotypes usage. The other stereotypes all have less than 5.00% of representativeness each. The graph in Fig. 14 is an excellent example of how the catalog can allow researchers to understand the evolution of OntoUML and its foundations. For example, the Event stereotype, introduced in OntoUML in 2019 [46], despite being relatively new, was well accepted by modelers being used in 32% (44) of the models included in the catalog, making it the sixth most used one.

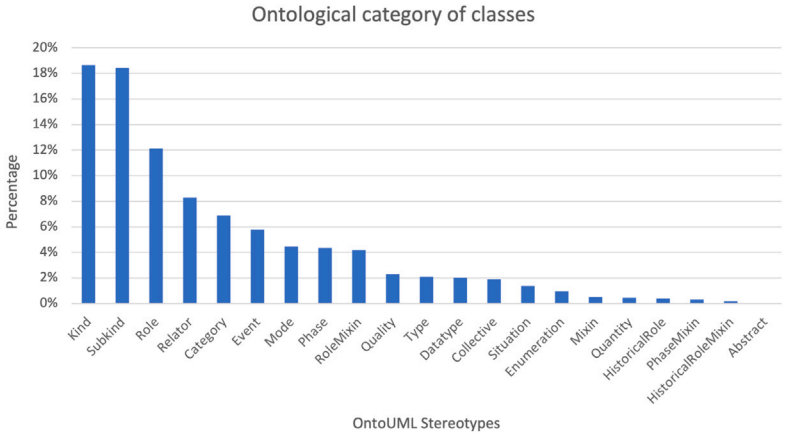


Fig. 14. Percentage frequency of OntoUML class stereotypes.

8.2. Statistics on the metadata

According to the data encoded in the `dct:issued` metadata field, the 141 models currently included in the catalog have been created between 2005 and 2023 (Fig. 15). The first model developed in OntoUML is the Example of Integrated Ontology [3], published in 2005 by Guizzardi in his Ph.D. thesis in which he first proposed the language. Therefore, it took some time for the language to become known and accepted in the community. It also took time for professionals and researchers to learn and start using it in their practices. This could explain why there is not a significant number of models cataloged until 2013—this year has a high number of models cataloged because of a set of models created by master and doctoral students for a course on Ontology Engineering given by a postgraduate program in Informatics. In the course, the students delivered two projects, one in which they had to take any ontology and remake it in OntoUML and another in which they had to make an ontology of a domain of their choice. Moreover, the release of the OntoUML Plugin for Visual Paradigm in March 2020 may have contributed to the increment in the number of models cataloged after 2020 since it can serialize models in JSON in compliance with the `ontouml-schema`. Finally, this analysis was developed at the beginning of 2023, which explains why so few models were cataloged this year.

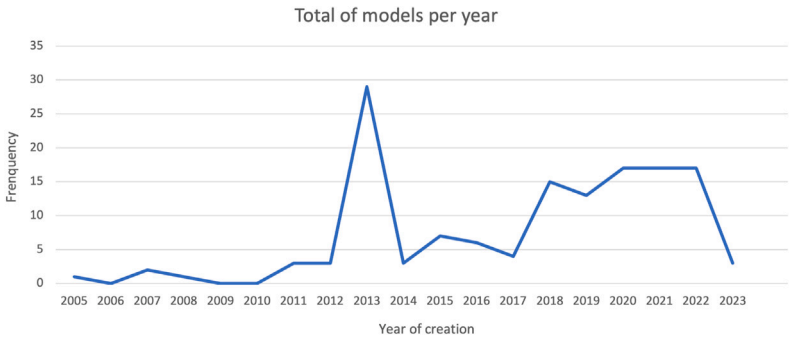


Fig. 15. Total of models per year.

The data encoded in the `dcat:keywords` field show that 187 different terms were used to express the domain represented by the models included in the catalog. The most frequent ones are *safety* and *software engineering*, with 6 occurrences each, followed by *education*, *finance*, and *value*, with 5 occurrences each. Keep in mind that these data come from an open metadata field where any term and more than one term could be included. Therefore, caution must be used when making inferences from these data.

The distribution of data from the metadata field `dct:language` shows that most of the models (129) use lexical terms in English (`en`), the others (9) use Brazilian Portuguese (`pt-br`), and there are still the multilingual models which simultaneously use English and Portuguese (`en, pt-br`) (2) and English and Dutch (`en, nl`) (1). The multilingual models can use more the one language in the same or in different diagrams.

The data encoded in the `dcat:theme` metadata field showed the catalog has models exemplifying 15 of the 21 LCC classes. As depicted in Fig. 16, the large representation is in *Class H - Social Science* and *Class T - Technology*, with 45 (32%) and 40 (28%) occurrences, respectively. Class H - Social Science encompasses models about finance, business, and trust. *Class T - Technology*, on the other hand, covers models about software engineering, aviation, construction, and robotics. The other 13 LCC classes exemplified

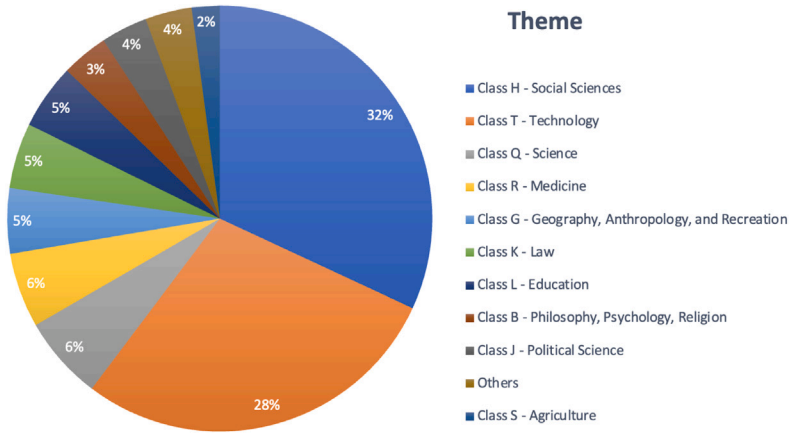


Fig. 16. Distribution of models by theme.

have less than 10 occurrences each. To facilitate the visualization of the data, we gathered 5 LCC classes (*Class A - General Works*, *Class M - Music*, *Class U - Military Science*, *Class V - Naval Science*, and *Class Z - Bibliography, Library Science, and General Information Resources*) that had one occurrence each in the category named *Others* in the chart.

The distribution of data from the metadata field `ocmv:representationStyle` shows that most of the models 126 (89.36%) are represented using OntoUML stereotypes, 13 (9.22%) of them directly extending UFO and 2 (1.42%) used both OntoUML stereotypes and UFO. When using both, authors usually used them mixed in the same diagram. This is because, initially, OntoUML only allowed the representation of (first-order) endurant types. There were no explicit provisions for event types, situation types, and higher-order types, for example. Hence, modelers appealed to workarounds, e.g., extending the set of OntoUML stereotypes with their own custom stereotypes or using plain UML class diagrams specializing UFO classes.

Most models in the catalog (105) were developed in the context of research, as can be seen in Fig. 17(a). This reflects the data collection described in Section 3 that had academic publications on UFO and OntoUML as the main source of models. There are only few models from the industry because in this context models are considered assets and are not usually publicly disclosed. The fact that most models were developed in a research context may explain their relatively small size (see Table 4). Academic publications have a space limitation, and models must adjust to this limitation while maintaining their distinctiveness. This may also explain why many models (61, 44% of the total) have one diagram. Since in industry, there is no such limitation, models can be larger. The National Agency of Terrestrial Transportation (ANTT) Ontology [35], for example, is the largest model in the catalog with 138 diagrams, 1222 classes, 655 associations, and 1119 generalizations.

The data encoded by the metadata property `mod:designForTask` reveals that conceptual clarification, with 103 occurrences, was by far the most frequent reason that triggered the construction of the models listed in our catalog (see Fig. 17(b)). The reason for creating a model can be related to the context in which it was developed. This relationship shows that ontological analysis is often developed in academia, where models are generally created to explain domains and develop conceptualizations. There are also more practical models developed to solve specific problems, such as supporting interoperability (22) and software engineering (18). We believe that these types of models would be more common among models developed in industry, but our analysis could not prove this hypothesis because of the limited number of models developed in this context in the analyzed sample. In addition, the models created for learning purposes (25) were developed in the classroom context.

Finally, as observed in Fig. 17(c), most of the models in the catalog (123) are categorized as domain ontologies according to the data in the metadata field `ocmv:ontologyType`, which can also be context-related. In research, domain, and core ontologies are typically used to represent how a community conceptualizes a phenomenon of interest. There are few cases in the sample where researchers have used applied ontologies. Those are more common among classroom and industry models, which are contexts that accept better ontological representations based on a more particular viewpoint of a user or a developer. This result, however, cannot be generalized because the sample under study only contains a few models created in the setting of classroom and industry.

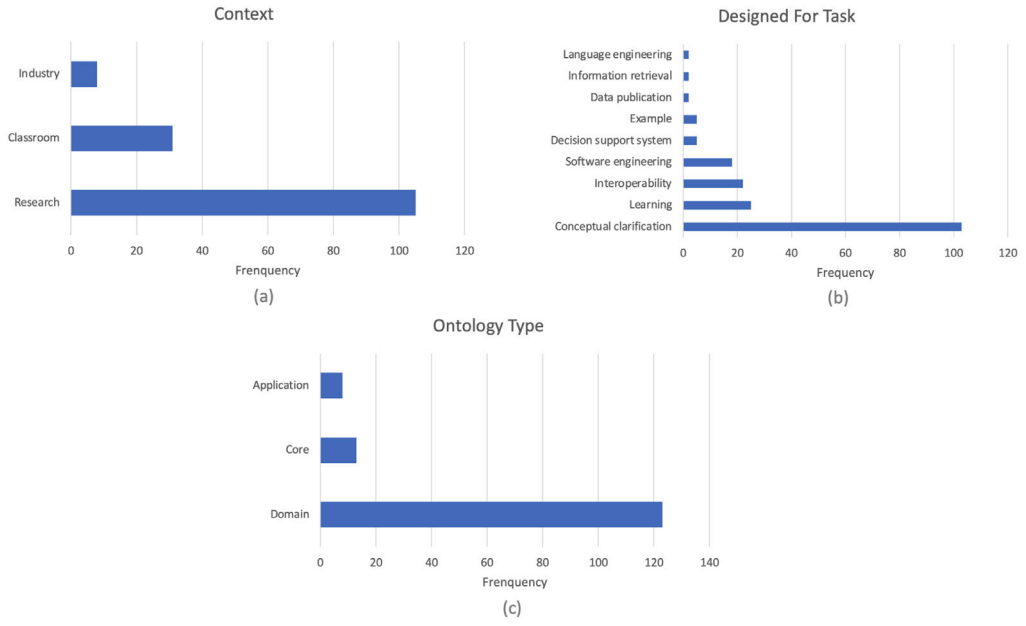


Fig. 17. Distribution of models by (a) context, (b) design for task, and (c) ontology type. As the model may have multiple values for these fields, the sum of the values presented in each graphic is greater than the number of models (141).

9. Related work

Our contribution here builds primarily on recent efforts in the generation of repositories for maintaining and reusing knowledge resources, such as ontologies, conceptual models, and vocabularies. Having analyzed the collected data and their organization, we identified certain initiatives related to ours that we will discuss in the sequel. Considering the different scopes, we excluded from this section catalogs for artefacts distinct from models or ontologies (e.g., design patterns catalogs).

In the past, some of us have made a first attempt to gather and organize OntoUML models [6]. This effort gathered 54 models, most of which were (or will eventually be) included in the catalog presented here. This repository of models—which is no longer available or maintained—-was created with the specific goal of supporting the empirical discovery of ontological anti-patterns. Differently from ours, it was not built intending to foster open and collaborative community participation.

Our initiative is also comparable to the Lindholmen Dataset [47], a comprehensive catalog of UML models. Their approach was to gather UML models automatically from open-source projects on GitHub and put them into a reference hub. The output is a catalog with over 93,000 UML diagrams from over 24,000 projects. This catalog includes a list of projects with UML files and their file format. In addition, they also provide a database that contains the metadata of the projects.

Another catalog of UML models was proposed by Shcherban et al. [48], but it contains only non-structured data. It comprises 3231 images that were manually collected and labeled to indicate they represent class diagrams (700), activity diagrams (454), use case diagrams (651), and sequence diagrams (706). They intended to create a dataset with the specific purpose of using a deep learning technique and, in particular, to evaluate neural network architectures for multi-class classification of UML diagrams. Their approach is used to determine whether an image is a UML diagram or not, and what type of UML diagram an image contains.

In a similar context, but with motivations that differ from those the mentioned datasets, which were created specifically for testing purposes, we must also mention the Data Set of OCL Expressions on GitHub [49], with 9188 Object Constraint Language (OCL) expressions originating from 504 EMF meta-models, built to facilitate empirical studies, like the replication of earlier studies on a larger and more diverse dataset, to replicate corpus-based studies conducted in traditional software engineering in a Model-driven Engineering (MDE) context, to provide a complementary perspective on the earlier results obtained through controlled experiments; and to evaluate practical limitations of previously proposed techniques.

Finally, we identified related works that had as main intention to present their methods for building a catalog, having the generated catalogs only as a consequent contribution. Here we cite the BPMN Artifacts Dataset 2021 [50] with 79,713 distinct BPMN models from 18,534 open-source projects, created through mining software projects on GitHub, and the ModelSet [51] with 5460 Ecore and 5120 UML models, built with a labeling method that includes features like automatic model grouping by similarity, visualizations, and label review.

These aforementioned catalogs are much larger than ours. They used data mining, web crawlers, and other automated data collection tools, emphasizing quantity while striving only for a minimal quality threshold for the models and the homogeneity of data. Our goals greatly diverge from theirs. First, in scope, they include UML, BPMN, and OCL information, whereas our focus is on ontology-driven conceptual models. All our models are available in the same formats and are described with rich and linked metadata, making our catalog more well-suited for reuse and in-depth study.

The Linked Open Vocabularies (LOV) [52] is a platform that provides access to a catalog of OWL vocabularies. Starting in 2011, LOV is now hosted by the Open Knowledge Foundation, and it currently offers almost eight hundred vocabularies with over 76 thousand terms. LOV is based on some quality requirements, including URI stability and availability on the web. It relies on standard formats and publication best practices, detailed metadata, and documentation. As a distinctive feature, LOV shows indicators that are not provided by other catalogs, such as the interconnections between vocabularies, the versioning history along with past and current editors (individual or organization). LOV is a catalog of vocabularies and/or lightweight (computational) ontologies, i.e., Semantic Web artefacts focused on web-based information sharing. Ours, in contrast, focuses on ontology-driven conceptual models (also termed ‘reference ontologies’), which contrary to lightweight ontologies, focus on expressiveness and domain appropriateness, often capturing the results of ontological analyzes [3].

An example of a repository of ontologies as logical specifications is OntoHub [53], which collects nearly 22,500 artefacts organized in more than 150 repositories—however, most of the ontologies there are also lightweight models. We can say the same for the LOV-inspired Linked Open Vocabularies for the Internet of Things (LOV4IoT) [54], and for the smartcity.linkeddata.es [55], which are both domain-specific repositories (the latter containing Smart City datasets). In this sense, they are like the BioPortal [56], which includes over 1000 life sciences ontologies. Unlike these other approaches, the OWL models in BioPortal usually are based on the Basic Formal Ontology (BFO) foundational ontology and, in principle, have a similar focus (e.g., with relation to domain appropriateness) to the ones in our catalog. However, despite their firm grounding, these models are subject to OWL’s expressivity limitations and, hence, omit many important ontological nuances (e.g., related to modality and multi-level structures). Since these ontologies are rendered as textual (sentential) logical specifications, they do not provide data for supporting the study of diagrammatic/visual aspects of domain representations (e.g., model layout, visual patterns and anti-patterns).

We see many other differences between our catalog and its related ones, the most important one being the fact that all catalogs here mentioned were not built to comply with the FAIR principles and they do not present this as a future development goal. Some of the related catalogs are only weakly documented (e.g., the BPMN Artifacts Dataset 2021), while we have examples of catalogs that are documented only through their publication papers (e.g., Shcherban et al. OCL Dataset).

Additionally, we regularly update our catalog, while others have received few or no updates since their release, causing them to become outdated over time (e.g., the Lindholmen Dataset, OCL Dataset, smartcity.linkeddata.es). Only a small set of them claim to be or are indeed curated (e.g., LOV, LOV4IoT, smartcity.linkeddata.es). Even though many declare to provide metadata, the information

is most of the time poor or not well-documented or they rely on data providers to do so (e.g., ModelSet and OntoHub). We can see the difference when checking the licensing information: while some works provide licensing only to their catalogs and not the contained datasets (e.g., BPMN Artifacts Dataset 2021, LOV4IoT), others do the opposite (e.g., smartcity.linkeddata.es, ModelSet), and, finally, some do not have licensing data at all (e.g., OCL Dataset).

10. Final considerations

In this paper, we presented the first FAIR Model Catalog for Ontology-Driven Conceptual Modeling Research. We provide a structured, collaborative, and open-source catalog of ODCM models designed with the OntoUML language (or by extending the UFO ontology). This resource supports the conceptual modeling and ontology engineering communities with many important empirical tasks. These include language design, understanding, and evolution; machine-learning research over model data; testing model manipulation (e.g., code generation, mining, modularization, and abstraction); and model reuse.

The catalog currently contains 141 models, but we expect it to grow, especially considering the UFO and OntoUML relevance to the ODCM field and considering that the catalog is open to contributions from the community. Instructions on how to collaborate are available on the repository's GitHub page. Over the years, a variety of experimental modeling tools and prototypes have been developed for OntoUML. To be included in the catalog, the models originally built with these tools must be reconstructed in the currently supported OntoUML Visual Paradigm plugin. Considering the difficulty of this task, we intend to research potential methods for partially automating it (e.g., by automating the normalization of data and by partially generating models from diagram images). Additionally, we intend to create a service in the OntoUML plugin for modelers to submit their models directly to the catalog.

Inspired by the LOV initiative (which is intended as a high-quality catalog of reusable vocabularies), we envisage the creation of a Linked Open OntoUML Models (LOOM) initiative. Unlike LOV, LOOM would organize the space of ontologically well-founded domain models, i.e., a space of conceptual models grounded on a foundational ontology and, thus, having deeper ontological semantics by design.

Even though our catalog is restricted to UFO/OntoUML conceptual models, its metadata schema could be easily adapted to accommodate models built following other foundational ontologies. Nonetheless, to the best of our knowledge, UFO is the only mainstream foundational ontology [1] that has an ODCM language that is explicitly associated with it, i.e., in a technical sense: (i) having the modeling primitives of the language directly reflecting the distinctions of the ontology; (ii) having the grammatical constraints of the language explicitly reflecting the axiomatization of the ontology.

Finally, as previously discussed, the models are included in this catalog in their original form, i.e., preserving the original modeling choices made by their creators. This is important to study how the language is used in practice, what are the most common modeling errors and anti-patterns, how different users subvert the grammatical rules of the language signaling possible evolution trends, etc. However, as a direct consequence, the catalog shall contain models that are of a variety of quality levels, including models bearing syntactic, semantic, and pragmatic problems. This hinders the potential (re)use of some of these models (e.g., as seed models or reusable modeling components). In future work, we intend to address this issue by investigating methodological and computational mechanisms for assessing some quality aspects of these models (e.g., with relation to syntactical correctness, presence of anti-patterns, and visual pragmatics, among others).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank Accenture Israel Cybersecurity Labs for supporting this work, as well as Ítalo Oliveira, Thomas Derave, Tim van Ee, Cristiano Silva, and Lucas Maddalena for their contributions to the catalog. We especially thank András Komáromi for his contribution and inspiring passion for modeling in OntoUML.

César H. Bernabé is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 825575. Mattia Fumagalli is supported by the *"Dense and Deep Geographic Virtual Knowledge Graphs for Visual Analysis - D2G2"* project, funded by the Autonomous Province of Bolzano (CUP ID: I53C21000180003). João Paulo A. Almeida is funded by CNPq (313687/2020–0) and FAPES (281/2021, 1022/2022).

Appendix A. Metadata schema specification

The metadata schema of the OntoUML/UFO catalog consists of five main classes, namely `dcat:Resource`, `dcat:Dataset`, `dcat:Distribution`, `dcat:Catalog`, and `mod:SemanticArtefact`. In this appendix, we define these classes and their associated properties and provide guidelines on how to use them.

A **`dcat:Resource`** is an object that we keep track in the catalog (including the catalog itself). A resource can be described by the following properties:

- `dct:title`: Determines a title for the resource. Accepts `xsd:string` and `rdf:langString` literals. E.g., `"Common Ontology of Value and Risk"@en`, `"FAIR Model Catalog for Ontology-Driven Conceptual Modeling Research"@en`. There must be at most one title per language.
- `dct:alternative`: Determines an alternative title for the resource. Accepts `xsd:string` and `rdf:langString` literals. E.g., `"OntoUML/UFO Catalog"@en`.
- `dct:description`: Determines a free-text account of the resource. Accepts `xsd:string` and `rdf:langString` literals.
- `dct:issued`: Determines when the resource was created. Accepts literals of the types `xsd:dateTime`, `xsd:date`, `xsd:gYearMonth`, and `xsd:gYear`. e.g., `"2018"^^xsd:gYear`, `"2018-01-15"^^xsd:date`. When cataloging a model from documents, we recommend using the publication date from the first one.
- `dct:modified`: Determines when the resource was last modified. Accepts literals of the types `xsd:dateTime`, `xsd:date`, `xsd:gYearMonth`, and `xsd:gYear`. When cataloging a model based on documents, we recommend using the publication date from the latest one.
- `dct:license`: Identifies a legal document under which the resource is made available. E.g., https://creativecommons.org/licenses/by/4.0/.
- `dct:accessRights`: Identifies a `dct:RightsStatement` or a text concerning who and how the resource can be accessed. E.g., the statement http://publications.europa.eu/resource/authority/access-right/PUBLIC informs that something is "publicly accessible by everyone".
- `skos:editorialNote`: Determines a general note relative to the resource documentation process. Accepts `xsd:string` and `rdf:langString` literals. E.g., `"The model was originally designed in Portuguese and translated by the publisher."@en`.
- `dct:contributor`: Identifies a `foaf:Agent` who contributed to the development of the resource.
- `dct:creator`: Identifies a `foaf:Agent` who contributed to the creation of the resource.
- `dct:publisher`: Identifies the `foaf:Agent` who added the resource to the catalog. The publisher does not need to have created or contributed to the resource.

A **`dcat:Dataset`** is a `dct:Resource` that comprises a collection of data items and is available for access or download in one or more distributions. Here, we describe datasets using the following properties:

- `dcat:landingPage`: Identifies a web page where one can access the dataset, its metadata, its distributions, and additional information about it. E.g., https://www.model-a-platform.com is the landing page of the Digital Platform Ontology.
- `dct:bibliographicCitation`: Determines a bibliographic reference for the dataset in textual format. Accepts `xsd:string` and `rdf:langString` literals. E.g., `"Weigand, H., Johannesson, P., & Andersson, B. (2021). An artefact ontology for design science research. Data & Knowledge Engineering, 133."@en`
- `ocmv:storageUrl`: Determines a URL of a service in which the data and metadata of the dataset are stored. Accepts values in `xsd:anyURI`.
- `dcat:distribution`: Identifies an available `dcat:Distribution` of the dataset.
- `dcat:contactPoint`: Identifies a `vcard:vCard` that contains information on how to contact an agent responsible for the dataset. We required that at least an email is provided.

A **`dcat:Catalog`** is a curated `dcat:Dataset` consisting of metadata about resources. Here, we naturally focus on single catalog instance, the OntoUML/UFO Catalog, which contains metadata about conceptual models and their distributions. The additional properties we used are:

- `dcat:themeTaxonomy`: Identifies a knowledge organization system used to classify the semantic artefacts in the catalog. In our catalog, we use the Library of Congress Classification (LCC) system, which exists as a `skos:ConceptScheme`.
- `dcat:dataset`: Identifies a `mod:SemanticArtefact` listed in the catalog.

The conceptual models in our catalog are categorized as instances of **`mod:SemanticArtefact`**, a type of `dcat:Dataset` whose data items are the formalization of concepts, properties, relations, and constraints about a domain of interest. The metadata record of a semantic artefact extends that of a `dcat:Dataset` with the following properties:

- `dcat:keyword`: Determines a domain (partially) described by the semantic artefact. Accepts `xsd:string` and `rdf:langString` literals. E.g., the User Feedback Ontology is described with the keywords `"online user feedback"`, `"software engineering"`, and `"requirements engineering"`.
- `mod:acronym`: Determines an acronym one can use to refer to the semantic artefact. Accepts only `xsd:string` literals. E.g., `"RDBS-O"`, `"COVER"`, `"ROT"`.
- `dct:source`: Identifies resources that contain, present, or significantly influenced the development of the semantic artefact. We recommend the use of persistent and resolvable identifiers to refer to these resources, such as the Digital Object Identifier (DOI) or DBLP's URI. E.g., https://doi.org/10.3233/AO-150150, https://dblp.org/rec/journals/ao/Morales-Ramirez15.
- `dct:language`: Determines a language in which the lexical labels of the semantic artefact are written. We require the use of values listed in the IANA Language Sub Tag Registry. E.g., `"en"`, `"pt"`.
- `dcat:theme`: Identifies the central theme of the semantic artefact according to a theme taxonomy. In our catalog, the theme of an artefact must be a `skos:Concept` from the LCC. E.g., `"Class S - Agriculture"`, `"Class T - Technology"`.

- `mod:designedForTask`: Identifies a goal that motivated the development of the semantic artefact. To standardize the use of this property, we documented some recurrent modeling goals:
  - `ocmv:ConceptualClarification`: The artefact was created as the result of an ontological analysis of a concept, language, or domain of interest that sought to conceptually clarify and untangle complex notions and relations.
  - `ocmv:DataPublication`: The artefact was created to support the publication of some datasets. For instance, a conceptual model used to generate an OWL vocabulary to publish tabular data as linked open data on the web.
  - `ocmv:DecisionSupportSystem`: The artefact was created during the development of a decision support system.
  - `ocmv:Example`: The artefact was created to demonstrate how OntoUML can be used to solve a certain modeling challenge, to support an experiment involving OntoUML, or to exemplify how a generic model can be reused in more concrete scenarios.
  - `ocmv:InformationRetrieval`: The artefact was created to support the design of an information retrieval system.
  - `ocmv:Interoperability`: The artefact was created to support data integration, vocabulary alignment, or the interoperability of software systems.
  - `ocmv:LanguageEngineering`: The artefact was created for the design of a domain-specific modeling language.
  - `ocmv:Learning`: The artefact was created so that its authors could learn UFO and OntoUML. This usually applies to models developed by students as part of their course assignments.
  - `ocmv:SoftwareEngineering`: The artefact was created during the development of an information system. For instance, a conceptual model used to generate a relational database.
- `ocmv:context`: Identifies an `ocmv:Context` in which the artefact was developed, namely:
  - `ocmv:Research`: The artefact was developed as part of a research project. This usually implies that the artefact was featured in a scientific publication.
  - `ocmv:Industry`: The artefact was developed for a public or private organization.
  - `ocmv:Classroom`: The artefact was developed within the context of a course on conceptual modeling, most likely as a course assignment.
- `ocmv:representationStyle`: Identifies an `ocmv:OntologyRepresentationStyle` representation styles adopted in the artefact. We account for the existence of two values:
  - `ocmv:OntoumlStyle`: Characterizes a model that contains at least one class, relation, or property using a valid OntoUML stereotype.
  - `ocmv:UfoStyle`: Characterizes a model that contains at least one class or relation from UFO without an OntoUML stereotype.
- `ocmv:ontologyType`: Identifies the categorization of the ontology according to its scope. For this catalog, we adopt the following three categories proposed by Roussey et al. [57]:
  - `ocmv:Core`: An ontology that grasps the central concepts and relations of a given domain, possibly integrating several domain ontologies and being applicable in multiple scenarios. E.g., UFO-S, a commitment-based ontology of services, can be considered a core ontology because it applies to services in multiple domains, such as medical, financial, and legal services.
  - `ocmv:Domain`: An ontology that describes how a community conceptualizes a phenomenon of interest. In general, a domain ontology formally characterizes a much narrower domain than a core ontology does. E.g., OntoBio is a domain ontology of biodiversity.
  - `ocmv:Application`: An ontology that specializes a domain ontology where there could be no consensus or knowledge sharing. It represents the particular model of a domain according to a single viewpoint of a user or a developer.

Finally, a **`dcat:Distribution`** is a `dcat:Resource` that materializes a `dcat:Dataset`. The characterization of a distribution accounts the following additional properties:

- `dcat:downloadURL`: Identifies a URL that can be used to download the distribution.
- `dcat:mediaType`: Identifies the media type of the distribution. Valid media types are only those defined by IANA, such as https://www.iana.org/assignments/media-types/application/json. If there is no media type for the format of the distribution, the type `application/octet-stream` must be used for binary files and the type `text/plain` must be used for text files.
- `dct:format`: Identifies the format of the distribution. This property should be used to complement `dcat:mediaType` when the distribution format is not listed by IANA. We limit the use of this property with URIs so that more context about how to manipulate a distribution can be provided. E.g., https://www.file-extension.info/format/vpp.
- `ocmv:isComplete`: Determines if a distribution contains all the data from the *dcat:Dataset* it materializes. In the catalog, the distributions of models in VPP, JSON, and Turtle are complete, while those in image format are not.
- `ocmv:conformsToSchema`: Identifies a schema upon which the distribution can be validated against. E.g., a JSON Schema document, a SHACL shape, and an XML Schema document. The identified schema should be compatible with the media type of the distribution. That is, if a distribution is in JSON, the schema cannot be an XML Schema.

Appendix B. Catalog files

The Git repository of the OntoUML/UFO Catalog functions as its data storage service, hosting the datasets and resources that users rely on to conduct their research activities. Fig. B.18 provides a visual representation of the files' organization employed within the catalog's repository.

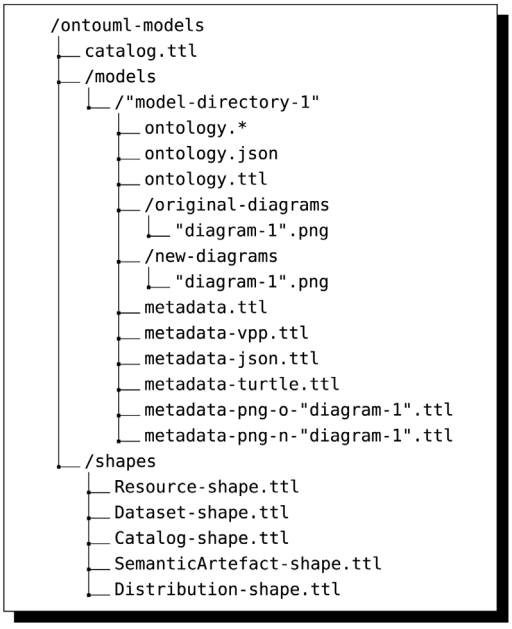


Fig. B.18. Catalog files' structure.

The description of each of the files contained in the catalog, including the ones sent by users and those automatically generated, is provided as follows.

- `catalog.ttl`: the turtle file that contains all metadata about the catalog in linked data format.
- `/models`: the directory containing all cataloged models.
- `/"model-directory-1"`: a directory containing a single model and distributions that materialize it. Most model directories' names are composed of information about the model itself, such as the name of the first author, the year of publication, or the name of the model.
- `ontology.*`: the file generated by the modeling tool used to create or reproduce the model. The modeling tool in question must be able to serialize the model in JSON format in conformance with the `ontouml-schema` (e.g., the Visual Paradigm UML CASE, `.vpp` extension, with the OntoUML Plugin for Visual Paradigm).
- `ontology.json`: the JSON serialization of the model in conformance with the `ontouml-schema`.
- `ontology.ttl`: the turtle serialization of the model in linked data format described with the OntoUML Metamodel Vocabulary. This file is automatically generated from the JSON serialization.
- `/original-diagrams`: the directory containing all diagrams of the model in PNG format. These diagrams are either (i) created from the original file generated by the modeling used, or (ii) extracted from the source where the model was published (e.g., screenshots from the original publication).
- `/new-diagrams`: the directory containing all diagrams of the model in PNG format. This directory must only be used when the original model was previously built in a non-supported modeling editor and the original diagrams have been extracted from the model's source.
- `metadata.ttl`: the turtle file that contains all metadata about the model in linked data format.
- `metadata-vpp.ttl`: the turtle file that contains all metadata about the model file, the `ontology.vpp` distribution, in linked data format. This file is automatically generated.
- `metadata-json.ttl`: the turtle file that contains all metadata about the model's JSON serialization file, the `ontology.json` distribution, in linked data format. This file is automatically generated.
- `metadata-turtle.ttl`: the turtle file that contains all metadata about the model's turtle serialization file, the `ontology.ttl` distribution, in linked data format. This file is automatically generated.
- `metadata-png-o-"diagram-1".ttl`: a turtle file that contains all metadata about one of the model's original diagrams, a `/original-diagrams/"diagram-1".png` distribution, in linked data format. A file is automatically generated for each original diagram.
- `metadata-png-n-"diagram-1".ttl`: a turtle file that contains all metadata about one of the model's new diagrams, a `/new-diagrams/"diagram-1".png` distribution, in linked data format. A file is automatically generated for each new diagram.

- `/shapes`: the directory containing the SHACL shapes used in the validation of metadata schemas.
- `Resource-shape.ttl`: the turtle file that contains the SHACL shape used to validate metadata about resources of type `dcat:Resource`.
- `Dataset-shape.ttl`: the turtle file that contains the SHACL shape used to validate metadata about resources of type `dcat:Dataset`.
- `Catalog-shape.ttl`: the turtle file that contains the SHACL shape used to validate metadata about resources of type `dcat:Catalog`.
- `SemanticArtefact-shape.ttl`: the turtle file that contains the SHACL shape used to validate metadata about resources of type `mod:SemanticArtefact`.
- `Distribution-shape.ttl`: the turtle file that contains the SHACL shape used to validate metadata about resources of type `dcat:Distribution`.

References

[1] M. Verdonck, F. Gailly, Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling, in: Conceptual Modeling. ER 2016, 9974, Springer, 2016, pp. 83–97, http://dx.doi.org/10.1007/978-3-319-46397-1_7.

[2] G. Guizzardi, A.B. Benevides, C.M. Fonseca, D. Porello, J.P.A. Almeida, T.P. Sales, UFO: Unified Foundational Ontology, Appl. Ontol. 17 (1) (2022) 167–210, http://dx.doi.org/10.3233/ao-210256.

[3] G. Guizzardi, Ontological foundations for structural conceptual models (Ph.D. thesis), in: CTIT Ph.D. thesis Series, (05–74) Telematica Instituut / CTIT, 2005.

[4] G. Guizzardi, C.M. Fonseca, J.P.A. Almeida, T.P. Sales, A.B. Benevides, D. Porello, Types and taxonomic structures in conceptual modeling: A novel ontological theory and engineering support, Data Knowl. Eng. 134 (2021) 101891, http://dx.doi.org/10.1016/j.datak.2021.101891.

[5] M.D. Wilkinson, et al., The FAIR guiding principles for scientific data management and stewardship, Sci. Data 3 (1) (2016) 160018, http://dx.doi.org/10.1038/sdata.2016.18.

[6] T.P. Sales, G. Guizzardi, Ontological anti-patterns: empirically uncovered error-prone structures in ontology-driven conceptual models, Data Knowl. Eng. 99 (2015) 72–104, http://dx.doi.org/10.1016/j.datak.2015.06.004.

[7] R. Pergl, T.P. Sales, Z. Rybola, Towards OntoUML for software engineering: From domain ontology to implementation model, in: Model and Data Engineering. MEDI 2013, 8216, Springer, 2013, pp. 249–263, http://dx.doi.org/10.1007/978-3-642-41366-7_21.

[8] G.L. Guidoni, J.P.A. Almeida, G. Guizzardi, Transformation of ontology-based conceptual models into relational schemas, in: Conceptual Modeling, Springer, 2020, pp. 315–330.

[9] G. Guizzardi, T.P. Sales, J.P.A. Almeida, G. Poels, Automated conceptual model clustering: a relator-centric approach, Softw. Syst. Model. 21 (4) (2022) 1363–1387, http://dx.doi.org/10.1007/s10270-021-00919-5.

[10] E. Romanenko, D. Calvanese, G. Guizzardi, Abstracting ontology-driven conceptual models: Objects, aspects, events, and their parts, in: Research Challenges in Information Science, Springer, 2022, pp. 372–388.

[11] M. Blaha, Patterns of Data Modeling, vol. 1, first ed., CRC Press, 2010, p. 261, http://dx.doi.org/10.1201/9781315380131.

[12] P.P.F. Barcelos, T.P. Sales, M. Fumagalli, C.M. Fonseca, I.V. Sousa, E. Romanenko, J. Kritz, G. Guizzardi, A FAIR model catalog for ontology-driven conceptual modeling research, in: Conceptual Modeling. ER 2022, vo. 13607, Springer, 2022, pp. 3–17, http://dx.doi.org/10.1007/978-3-031-17995-2_1.

[13] C. Bernabé, T.P. Sales, E. Schultes, N. van Ulzen, A. Jacobsen, L.O. Bonino da Silva Santos, B. Mons, M. Roos, A goal-oriented method for FAIRification planning, in: ER Forum. International Conference on Conceptual Modeling, CEUR-WS.org, 2023.

[14] A. Wright, H. Andrews, B. Hutton, G. Dennis, JSON Schema: A media type for describing JSON documents, 2017, URL https://json-schema.org/draft/2020-12/json-schema-core.html Internet-Draft. Internet Engineering Task Force.

[15] P. Hitzler, M. Krötzsch, B. Parsia, P.F. Patel-Schneider, S. Rudolph, et al., OWL 2 Web Ontology Language primer, 2009, URL https://www.w3.org/TR/owl2-overview/ W3C recommendation.

[16] C.M. Fonseca, T.P. Sales, V. Viola, L.B.R. Fonseca, G. Guizzardi, J.P.A. Almeida, Ontology-driven conceptual modeling as a service, in: JOWO 2021. the Joint Ontology Workshops, 2021.

[17] P. Uhnák, R. Pergl, The OpenPonk modeling platform, in: 11th International Workshop on Smalltalk Technologies, Association for Computing Machinery, 2016, pp. 1–11, http://dx.doi.org/10.1145/2991041.2991055.

[18] A.B. Benevides, G. Guizzardi, B.F.B. Braga, J.P.A. Almeida, Validating modal aspects of OntoUML conceptual models using automatically generated visual world structures, J. Univers. Comput. Sci. 16 (20) (2010) 2904–2933.

[19] B.F.B. Braga, J.P.A. Almeida, Modeling stories for conceptual model assessment, in: Advances in Conceptual Modeling, vol. 9382, Springer, 2015, pp. 293–303, http://dx.doi.org/10.1007/978-3-319-25747-1_29.

[20] S.J. Ali, G. Guizzardi, D. Bork, Enabling representation learning in ontology-driven conceptual modeling using graph neural networks, in: Advanced Information Systems Engineering. CAiSE 2023, Springer, 2023.

[21] F. Dalpiaz, X. Franch, J. Horkoff, iStar 2.0 language guide, 2016, http://dx.doi.org/10.48550/arxiv.1605.07767.

[22] F.B. Ruy, G. Guizzardi, R.A. Falbo, C.C. Reginato, V.A. Santos, From reference ontologies to ontology patterns and back, Data Knowl. Eng. 109 (2017) 41–69, http://dx.doi.org/10.1016/j.datak.2017.03.004.

[23] T. Derave, T.P. Sales, F. Gailly, G. Poels, Towards a reference ontology for digital platforms, in: Conceptual Modeling. ER 2020, Springer, 2020, pp. 289–302.

[24] J.C. Nardi, R. de Almeida Falbo, J.P.A. Almeida, G. Guizzardi, L.F. Pires, M.J. van Sinderen, N. Guarino, C.M. Fonseca, A commitment-based reference ontology for services, Inf. Syst. 54 (2015) 263–288.

[25] Í. Oliveira, T.P. Sales, R. Baratella, M. Fumagalli, G. Guizzardi, An ontology of security from a risk treatment perspective, in: Conceptual Modeling. ER 2022, Springer, 2022, pp. 365–379.

[26] T.P. Sales, F. Baião, G. Guizzardi, J.P.A. Almeida, N. Guarino, J. Mylopoulos, The common ontology of value and risk, in: Conceptual Modeling. ER 2018, vol. 11157, Springer, 2018, pp. 121–135, http://dx.doi.org/10.1007/978-3-030-00847-5_11.

[27] M. Hepp, Goodrelations: An ontology for describing products and services offers on the web, in: 16th International Conference on Knowledge Engineering, Springer, 2008, pp. 329–346.

[28] C.M. Fonseca, D. Porello, G. Guizzardi, J.P.A. Almeida, N. Guarino, Relations in ontology-driven conceptual modeling, in: Conceptual Modeling, Springer, 2019, pp. 28–42.

[29] M.G.S. Teixeira, An ontology-based process for domain-specific visual language design (Ph.D. thesis), Ghent University, 2017.

[30] G. Guizzardi, T.P. Sales, "As simple as possible but not simpler": Towards an ontology model canvas, in: 3rd Joint Ontology Workshops, vol. 2050, 2017.

[31] O.M. Group, Unified Modeling Language (OMG UML) Version 2.5.1, 2017.

[32] T.P. Sales, C.M. Fonseca, P.P.F. Barcelos, OntoUML Catalog Metamodel Vocabulary Specification - Version 1.1.0, URL https://w3id.org/ontouml-models/vocabulary/docs.
[33] G. Amaral, T.P. Sales, G. Guizzardi, D. Porello, Towards a reference ontology of trust, in: On the Move to Meaningful Internet Systems: OTM 2019 Conferences, Springer, 2019, pp. 3–21, http://dx.doi.org/10.1007/978-3-030-33246-4_1.
[34] L.O. Bonino da Silva Santos, K. Burger, R. Kaliyaperumal, M.D. Wilkinson, FAIR Data Point: A FAIR-oriented approach for metadata publication, Data Intell. 5 (1) (2022) 1–21, http://dx.doi.org/10.1162/dint_a_00160.
[35] C.A.M. Bastos, L.S. de Rezende, M.F. Caldas, A.S. Garcia, S.M. Filho, et al., Building up a model for management information and knowledge: The case-study for a Brazilian regulatory agency, in: 2nd International Workshop on Software Knowledge, 2011, pp. 3–11.
[36] P.P.F. Barcelos, Ontology-based provisioning for technology-independent multi-layer transport networks (Ph.D. thesis), Federal University of Espírito Santo, Vitória, Espírito Santo, 2015.
[37] R. D'Agostino, Goodness-of-Fit-Techniques, Routledge, 2017.
[38] J. Ferré, 3.02 - Regression diagnostics, in: Comprehensive Chemometrics, Elsevier, Oxford, 2009, pp. 33–89.
[39] N.L. Johnson, S. Kotz, N. Balakrishnan, Continuous univariate distributions, John wiley & sons, 1995.
[40] N.J. Gogtay, U.M. Thatte, Principles of correlation analysis, J. Assoc. Phys. India 65 (3) (2017) 78–81.
[41] J.S. Kritz, An ontology of board games based on the MDA framework (Master's thesis), Universidade Federal Do Rio De Janeiro, 2020, p. 124.
[42] F. Sikora, Modeling and data management in the domain of online education, (Bachelor thesis), Czech Technical University in Prague, 2021.
[43] A.C.F. Albuquerque, Desenvolvimento de uma ontologia de domínio para modelagem de biodiversidade [Development of a domain ontology for biodiversity modeling] (Ph.D. thesis), Federal University of Amazonas, 2011, p. 147.
[44] F.H. Rodrigues, L.F. Garcia, R. dos Santos Alvarenga Kuchle, M. Abel, An ontological model for turbidite channel migration, in: 12th Seminar on Ontology Research in Brazil, vol. 2519, CEUR-WS.org, 2019.
[45] B. Gonçalves, G. Guizzardi, J.G. Pereira Filho, An electrocardiogram (ECG) domain ontology, in: 2nd Workshop on Ontologies and Metamodels for Software and Data Engineering, 2007, pp. 68–81.
[46] J.P.A. Almeida, R.A. Falbo, G. Guizzardi, Events as entities in ontology-driven conceptual modeling, in: Conceptual Modeling. ER 2019, 11788, Springer, 2019, pp. 469–483, http://dx.doi.org/10.1007/978-3-030-33223-5_39.
[47] R. Hebig, T.H. Quang, M.R.V. Chaudron, G. Robles, M.A. Fernandez, The quest for open source projects that use UML: Mining GitHub, in: ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, 2016, pp. 173–183, http://dx.doi.org/10.1145/2976767.2976778.
[48] S. Shcherban, P. Liang, Z. Li, C. Yang, Multiclass classification of four types of UML diagrams from images using deep learning, in: 33rd International Conference on Software Engineering and Knowledge Engineering, KSI Research Inc., 2021, pp. 57–62, http://dx.doi.org/10.18293/seke2021-185.
[49] J. Noten, J.G. Mengerink, A. Serebrenik, A data set of OCL expressions on GitHub, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017, pp. 531–534, http://dx.doi.org/10.1109/msr.2017.52.
[50] J. Türker, M. Völske, T.S. Heinze, BPMN in the wild: A reprise, in: 14th Central European Workshop on Services and their Composition (ZEUS), in: CEUR Workshop Proceedings, vol. 3113, CEUR-WS.org, 2022, pp. 68–75.
[51] J.A.H. López, J.L. Cánovas Izquierdo, J.S. Cuadrado, ModelSet: a dataset for machine learning in model-driven engineering, Softw. Syst. Model. 21 (3) (2022) 967–986, http://dx.doi.org/10.1007/s10270-021-00929-3.
[52] P.-Y. Vandenbussche, G.A. Atemezing, M. Poveda-Villalón, B. Vatant, Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the web, Semantic Web 8 (3) (2017) 437–452, http://dx.doi.org/10.3233/sw-160213.
[53] M. Codescu, E. Kuksa, O. Kutz, T. Mossakowski, F. Neuhaus, Ontohub: A semantic repository engine for heterogeneous ontologies, Appl. Ontol. 12 (2017) 275–298, http://dx.doi.org/10.3233/ao-170190, 3-4.
[54] A. Gyrard, C. Bonnet, K. Boudaoud, M. Serrano, LOV4IoT: A second life for ontology-based domain knowledge to build semantic web of things applications, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), 2016, pp. 254–261, http://dx.doi.org/10.1109/FiCloud.2016.44.
[55] M. Poveda Villalón, R. García Castro, A. Gómez-Pérez, Building an ontology catalogue for smart cities, in: EWork and EBusiness in Architecture, Engineering and Construction: ECPPM 2014, vol. 1, CRC Press, 2014.
[56] M. Salvadores, P.R. Alexander, M.A. Musen, N.F. Noy, BioPortal as a dataset of linked biomedical ontologies and terminologies in RDF, Semantic Web 4 (3) (2013) 277–284, http://dx.doi.org/10.3233/sw-2012-0086.
[57] C. Roussey, F. Pinet, M.A. Kang, O. Corcho, An introduction to ontologies and ontology engineering, in: Ontologies in Urban Development Projects, Springer, 2011, pp. 9–38, http://dx.doi.org/10.1007/978-0-85729-724-2_2.

Tiago Prince Sales is a Senior Researcher at the University of Twente The Netherlands, where he is a member of the Semantics, Cybersecurity, and Services group. He received his Ph.D. from the University of Trento, Italy, with a cum laude distinction and the additional qualification of doctor europaeus. His main research focus is on ontology-driven conceptual modeling. In particular, he works on the development and evolution of the OntoUML language and of gUFO, a lightweight implementation of the the Unified Foundational Ontology for the semantic web. He investigates and develops engineering tools for conceptual modeling, such as model simulation and anti-patterns,and he develops well-founded reference ontologies for business and social domains, such as value, risk, competition, and trust. He has over ten years of experience in industrial and technology transfer projects in a wide range of domains, such as tourism, media asset management, public healthcare, transportation, and oil and gas.

Pedro Paulo F. Barcelos is a Researcher at the Semantics, Cybersecurity, and Services group at the University of Twente, the Netherlands. He holds a M.Sc. and Ph.D. in Electrical Engineering (Telecommunications and Information Technology) at the Federal University of Espírito Santo (UFES), Brazil, an MBA in Project Management at the University of Vila Velha, Brazil, and is graduated in Computer Engineering at UFES. His main interest areas are ontologies, applied ontologies, conceptual modeling, knowledge graphs, semantic web, and FAIR data.

Claudenir Morais Fonseca is a Researcher at the University of Twente, the Netherlands. He is a member of the Semantics, Cybersecurity, and Services group. He holds degrees of Computer Engineer and M.Sc. in Informatics at the Federal University of Espírito Santo—Brazil, and a Ph.D. degree at the Faculty of Computer Science, in the Free University of Bozen-Bolzano. Claudenir has been working in the latest years on building, formalizing, and validating core ontologies. He is currently interested in the development of solutions for ontology-driven conceptual modeling involving language design and model-driven engineering.

Isadora Valle Sousa is a Ph.D. candidate at the Faculty of Engineering, in the Free University of Bozen-Bolzano, Italy, and a member of the Conceptual and Cognitive Modeling Research Group. She holds a bachelor's degree in Industrial Engineering at Mercer University and PUC Minas, Brazil, an MBA in Business Administration, Management and Operations at Ibmec, a M.Sc. degree in informatics at FUMEC University, Brazil.

Elena Romanenko is a Ph.D. candidate at the Free University of Bozen-Bolzano, Italy, and a member of the KRDB Research Centre for Knowledge and Data. She holds a joint master's degree in Computational Science from the Perm State University, Russia, and the University of Reading, the UK. Her current research interests lie in pragmatic explanations of ODCMs and ontologies.

César Henrique Bernabé is a Ph.D. candidate in the Biosemantics group at Leiden University Medical Centre, in the Netherlands. He holds degrees of Computer Science and M.Sc. in Informatics from the Federal University of Espírito Santo, Brazil, where he was part of the Ontology & Conceptual Modeling Research Group (Nemo). His current research interests include ontologies, goal-oriented requirements engineering, conceptual modeling, FAIR and FAIRification, and Semantic Web.

Luiz Olavo Bonino da Silva Santos is an Associate Professor of the Semantics, Cybersecurity and Services group at the University of Twente and an Associate Professor of the BioSemantics group at the Leiden University Medical Centre, the Netherlands. Luiz has a background in ontology-driven conceptual modeling, semantic interoperability, service-oriented computing, requirements engineering and context-aware computing. In the past 9 years, Luiz has worked on designing and developing technologies, methods and tools to support making, publishing, indexing, searching and annotating FAIR (meta)data.

Mattia Fumagalli is a Fixed-term Assistant Professor, and a member of the In2Data and CORE groups, at the Free University of Bozen-Bolzano, Italy, where he is a lecturer for the Information Systems Design and Research Methodology courses. He has a Master's Degree in Philosophy at Università Cattolica del Sacro Cuore, and a Ph.D. in Computer Science at Università di Trento. His research concerns primarily Artificial Intelligence, with a particular focus on the automated support for Knowledge Representation and Conceptual Modeling.

Joshua Kritz holds a master's degree in Systems and Computing Engineering and a bachelor's degree in Applied Mathematics at the Federal University of Rio de Janeiro, Brazil. His main interests are conceptual modeling, knowledge discovery, interoperability, ontologies, game design, game development, game studies, algorithm development, and algorithm analysis.

João Paulo A. Almeida is a Full Professor at the Computer Science Department of the Federal University of Espírito Santo, Brazil. He is a founding member of the Ontology and Conceptual Modeling Research Group (NEMO). Since 2007, João Paulo has been working on the application of ontologies in enterprise architecture and enterprise modeling at NEMO/UFES, having had a key role in the development of OntoUML model-driven tools and core ontologies based on UFO (including UFO-S and UFO-L). More recently, he has led the development of the gUFO implementation of UFO. He has also made contributions to multi-level conceptual modeling with the MLT theory and the ML2 language. He is currently principal researcher in a FAPES-funded project on semantic technologies and their applications.

Giancarlo Guizzardi is a Full Professor of Software Science and Evolution as well as Chair and Department Head of Semantics, Cybersecurity & Services (SCS) at the University of Twente, The Netherlands. He is also an Affiliated Full Professor at the Department of Computer and Systems Sciences (DSV) at Stockholm University, in Sweden. He has been active for more than two decades in the areas of Applied Ontology, Conceptual Modeling, Business Informatics, and Information Systems Engineering, working with a multi-disciplinary approach that combines results from Formal Ontology, Cognitive Science, Logics and Linguistics. He is currently an associate editor for the Applied Ontology journal and for Data & Knowledge Engineering, a co-editor of the Lecture Notes in Business Information Processing series, and a member of a number of international journal editorial boards. Finally, he is a member of the Steering Committees of ER, EDOC, and IEEE CBI, and of the Advisory Board of the International Association for Ontology and its Applications (IAOA).