Preprint of:
E. Tzortzakakis & P. Fafalios.
Drawio2Triples: Semantic Validation and RDF Transformation of CIDOC CRM Example Diagrams.
In International Conference on Theory and Practice of Digital Libraries (TPDL 2025). Springer.

Drawio2Triples: Semantic Validation and RDF Transformation of CIDOC CRM Example Diagrams

Elias Tzortzakakis[1] and Pavlos Fafalios[1,2]

[1] Information Systems Laboratory, FORTH-ICS, Heraklion, Greece
[2] Technical University of Crete, Chania, Greece
{tzortzak,fafalios}@ics.forth.gr

Abstract. Drawing diagrams is a common practice in ontology documentation, used either to better communicate and explain the context of specific functional units of a model or to demonstrate use cases through examples based on well-known, meaningful stories and events. Drawio2Triples is a tool tailored for CIDOC CRM, a formal ontology widely used in the cultural heritage domain for documenting museum collections, historical archives, and other cultural resources. The tool supports the validation of ontology example diagrams created with diagrams.net (a.k.a. draw.io) and their transformation into RDF triples, suitable for storage in triplestore databases or use with RDF data management and ontology development tools. The validation process can detect semantic errors, such as invalid domain or range specifications of properties, incorrect use of property direction, or simple typos in the names of classes and properties. The tool is especially useful for pedagogical purposes, such as ontology learning, and is currently being extended to support any formal ontology.

Keywords: CIDOC CRM · Semantic Validation · Diagram Transformation · Ontology Learning · Draw.io · RDF

1 Introduction

Drawing diagrams is a common and long-established practice in ontology documentation and communication. Diagrams are often used to explain the structure and semantics of a model, to focus on specific functional units, or to illustrate practical use cases through examples. They support better understanding by providing a visual context that complements the formal definition of the ontology, can facilitate its communication with domain experts who are less familiar with formal notations, and can be particularly helpful in pedagogical settings, such as ontology learning and training activities, where diagrams of ontology instances facilitate the comprehension of the ontology concepts.

An example of a formal ontology is the CIDOC Conceptual Reference Model (CIDOC CRM) which is an ISO standard (ISO 2117:2023) and has been extensively used for information integration in the field of cultural heritage [3]. Due

to its event-centric nature and complex hierarchy of classes and properties, diagrams can greatly facilitate the understanding of the semantics of core classes and properties, clarify modeling patterns, and provide users with intuitive entry points to the ontology.

In recent years, the diagrams.net tool (also known as draw.io) has become widely used for creating such visual representations, due to its ease of use, browser-based interface, and support for collaborative editing. It allows ontology engineers and practitioners to manually construct diagrams that refer to ontology elements (instances, classes, properties), either for conceptual design, examples depiction, teaching, or documentation purposes.

However, the process of manually creating ontology instantiation diagrams inevitably introduces the possibility of semantic errors. Common mistakes include assigning properties to instances with incorrect domain or range classes, using inverse directions unintentionally, or including invalid or mistyped class and property names. These issues can lead to confusion, misinterpretation, or inconsistency between the diagram and the formal ontology, especially in cases where the diagram is used as a basis for further modeling or implementation.

To address this need, we introduce Drawio2Triples, a tool tailored for the CIDOC CRM ontology that supports the semantic validation and transformation of ontology instances diagrams created with draw.io. The tool parses the diagram content and checks for logical consistency with respect to the latest official version of CIDOC CRM. It reports semantic issues such as invalid domain-range usage, directionality errors, and unrecognized elements. Furthermore, it transforms the content of the diagram into RDF triples, enabling its use in triplestore databases or related RDF data management or ontology development applications. The tool also supports embedding of one or more images within a diagram. Each image is extracted and encoded in base64, allowing it to be included directly in the resulting RDF representation.

The tool is publicly available through the web[3]. While it has been initially designed for diagrams based on CIDOC CRM use cases, it is currently being extended to support any formal ontology.

The rest of this paper is organised as follows. Section 2 provides the required background and discusses related works. Section 3 details the functionality of the application. Finally, Section 4 concludes the paper and discusses future work.

2 Background and Related Work

2.1 Draw.io

The diagrams.net tool (also known as draw.io)[4] is a graph drawing software application used to create diagrams such as flowcharts, process diagrams, UML diagrams, network diagrams, and others. It is available both as a web application (for online use) and desktop application (for offline use), and has been

[3] https://isl.ics.forth.gr/cidoc_services/drawioXMLToTriples/
[4] https://diagrams.net/

widely used for creating visual representations of ontologies due to its ease of use, intuitive interface, and support for both online and offline editing.

One of its core features is the use of a structured XML format for serializing and storing diagrams. Each diagram created in draw.io is represented as an XML document that encodes the layout, shapes, labels, styles, and interconnections of the graphical elements. This XML structure enables programmatic access to the contents of a diagram, making it suitable for automated processing, transformation, or validation. In the context of ontology documentation, this format allows for the extraction of semantic elements from the visual layer, such as classes, properties, and instances, which can then be interpreted and converted into machine-readable formats.

2.2 Use Case: CIDOC CRM

The CIDOC Conceptual Reference Model (CIDOC CRM)[5] is a formal ontology developed for the cultural heritage domain, aiming to facilitate the integration, mediation, and exchange of heterogeneous documentation about museum collections and other cultural objects [3]. It has been an ISO standard since 2006 (ISO 21127) and has since undergone two official (ISO) revisions, in 2014 and 2023.[6] The latest stable community version is 7.1.3 (February 2024).[7]

The ontology is fundamentally event-centric, modeling the world in terms of events and spatiotemporal processes that connect actors, objects, places, and concepts. This approach enables the representation of rich contextual information and the dynamic aspects of cultural heritage data. It is extensively used by cultural institutions, research projects, and heritage information systems as a semantic framework for data integration and semantic interoperability. However, its expressive power comes with considerable complexity, as it includes a deep class hierarchy, a rich network of properties, and a variety of modeling constructs that require careful application. Indicatively, its latest release (7.1.3) consists of 81 classes, 160 properties, and a class hierarchy with a depth of nine levels.

To support understanding and adoption, diagrams play a key role in CIDOC CRM documentation. They are used to illustrate modeling examples, demonstrate use cases, and explain core notions of the ontology in a visual and pedagogically effective manner. To facilitate the creation of diagrams, a dedicated CIDOC CRM library for draw.io has been developed by the Canadian Heritage Information Network (CHIN).[8] The library contains the shapes of all the classes as well as the connectors for the properties of each ontology, while the colour scheme is the one adopted by the Special Interest Group (SIG) of CIDOC CRM.[9]

[5] https://cidoc-crm.org/
[6] https://www.iso.org/standard/85100.html
[7] https://cidoc-crm.org/versions-of-the-cidoc-crm (accessed on June 2, 2025)
[8] https://github.com/chin-rcip/diagrams.net_libraries
[9] More in: https://cidoc-crm.org/Resources/cidoc-crm-graphical-representation-templates

2.3 Related Work

A wide range of tools have been developed for visualizing RDF data [1]. These tools typically focus on graph-based representations of semantic data to support navigation, exploration, and understanding. Examples include LodLive [2], VOWL [10], and WebVOWL [9]. While effective for presenting data that already exists in RDF form, these tools operate in the opposite direction of our approach: they take RDF as input and produce visual representations, whereas Drawio2Triples starts from visual diagrams and produces RDF data as output.

Closer to our work are tools and approaches that support the construction of ontology instances (RDF content) through graphical interfaces or diagrammatic input. For example, Graffoo [5] allows users to create diagrams that represent OWL ontologies and provides mappings to OWL constructs, although it is mainly used for ontology design rather than instantiation. Similarly, tools like WebProtégé [7] and VocBench [11] provide form-based interfaces for entering RDF/OWL content, but they lack support for free-form diagrammatic modeling. Finally, some research prototypes have explored template/form-based and sketch-driven ontology population, but these are often limited in scope or lack semantic validation capabilities. Examples include Populus [8] (providing a spreadsheet-like interface), and OntoCAD [6] (tailored to CAD drawings and buildings-related ontologies).

To our knowledge, Drawio2Triples is the first tool that combines support for free-form ontology instantiation diagrams (via draw.io), semantic validation against an underlying ontology, and transformation into RDF triples. This makes it particularly suitable for documentation, teaching, and rapid prototyping of data using a very familiar (to end users) visual paradigm.

3 The Drawio2Triples Application

3.1 Input and Output

The application takes as input one `.drawio` file, which is an XML file used to save and store the data of a draw.io diagram. The output of the application includes:

- statistics: number of classes and number of properties (in plain text format),
- ontology validation issues along with information on how to fix them (in plain text format),
- the generated RDF data in two formats: Turtle (.ttl) and RDF/XML (.rdf).

For the transformation to RDF, we consider the RDFS generation policies adopted by CIDOC CRM.[10] For example, when an instance of the class `E62 String` is detected, it is not represented as a separate resource; instead, its value is assigned directly as a literal. The same is followed for the classes `E59`

[10] https://gitlab.isl.ics.forth.gr/cidoc-crm/cidoc_crm_rdf/-/tree/master/7.1.3

`Primitive Value`, `E60 Number`, `E61 Time Primitive`, `E94 Space Primitive`, and `E95 Spacetime Primitive`.

The URIs of the class instances in the two RDF files consist of a constant prefix and a unique ID. The ID is the same as that of the shape in the draw.io file. In addition to the RDF triples that describe the instances depicted in the diagram, the generated RDF files include three supplementary triples: one that represents the entire diagram as an instance of `rdfs:Resource`, one that assigns a label (`rdfs:label`) to this resource, and one that provides a comment (`rdfs:comment`). The label follows the format: "CIDOC-CRM v7.1.3 example produced by input file: <filename.drawio>". The comment includes the statistics and any validation errors that were detected (more below). A related future task is to make all the above configurable through the user interface.

3.2 Diagram Design Requirements

For the creation of a class instance, the "List" shape from the general library should be used. The name of the class must be written in the upper section of the shape and the name of the class instance in the lower section, as illustrated in Figure 1a. Optionally, an image can be included at the bottom of the shape (Figure 1b). The image must be embedded within the shape (by dragging and dropping it onto the shape). During data transformation, the image is automatically encoded in base64 and embedded in the resulting RDF file.

For the properties connecting class instances, any of the arrows can be selected. The name of the ontology property should be entered in the textbox that appears after double-clicking the arrow, and the arrow's source and target endpoints must be correctly attached to the class instances (see Figure 1c).

If syntactic errors are detected in the diagram, such as a property whose source or target endpoint is not property attached, the application displays an error message to the user and does not proceed to the validation and transformation processes.

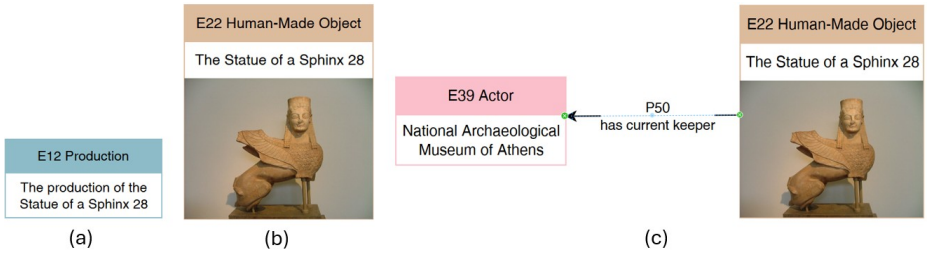

Fig. 1. Example of drawing (a) a class instance, (b) a class instance together with an image, and (c) a property instance connecting two class instances, in draw.io.

An alternative (and more convenient) way for adding CIDOC CRM class and property instances into the draw.io diagram is to use a ready-made library[11] which contains all classes and properties of the latest official CIDOC CRM version (7.1.3) and which follows the colour scheme adopted by CIDOC CRM. The library can be quickly imported from the draw.io menu (File → Open Library From: → URL). The examples of Figure 1 make use of this library.

Properties of properties. CIDOC CRM contains properties that have their own properties (called .1 properties). In this case, the domain of the property is a property and the range a class. An example is the property *P14 carried out by: E39 Actor* (of E7 Activity) which has the property *P14.1 in the role of: E55 Type*. This .1 property allows specifying the role that the actor had while carrying out the activity.

Since RDF does not provide a direct way to express properties of properties, there have been different approaches on how to implement them, including standard RDF reification, named graphs, singleton properties, N-ary relations, and RDF-star (see Sect. 3.3 in [4] for more details). CIDOC CRM SIG recommends to implement them using *property classes* and provides a supplementary RDFS file for this.[12] Drawio2Triples follows this approach for validating and transforming .1 properties existing in the drawing. Such properties are defined similarly to simple property arrows by specifying the target endpoint of the arrow to a class instance and grouping together the .1 and the parent property arrows. An example is shown in Figure 2.

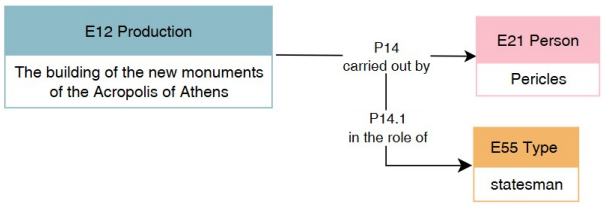

Fig. 2. Example of drawing a property of property (.1 property) in draw.io.

3.3 Validation

The application performs the following ontology consistency checks and provide feedback on how to resolve them:

– *Invalid domain or range class*: Each CIDOC CRM property has a specific domain and range class. However, valid domain or range classes include not only the explicitly defined ones, but also all their subclasses, as determined

[11] https://isl.ics.forth.gr/cidoc_services/files/examples/diagrams_to_triples/CIDOC-CRM_7.1.3_examples_library.xml

[12] See the file `CIDOC_CRM_v7.1.3_PC.rdf` at: https://gitlab.isl.ics.forth.gr/cidoc-crm/cidoc_crm_rdf/-/tree/master/7.1.3

by the transitive closure of subclass relationships in the ontology. When such an error is detected, the application suggests the correct set of domain or range classes associated with the property.
- *Wrong property direction*: Most CIDOC CRM properties have inverse properties. For example, the property *P2 has type: E55 Type* (of E1 CRM Entity) has the inverse property *P2i is type of: E1 CRM Entity*. A common mistake is using the property in the wrong direction when linking an instance of the domain class to an instance of the range class. When such an error is detected, the application suggests using the appropriate inverse property.
- *Unknown class or property name*: This occurs when the name of a class or property does not match any class or property in CIDOC CRM, typically due to a typo or the use of a deprecated term. When such an error is detected, the application suggests renaming the class or property name.

If one or more of the above errors are detected, the application continues generating the RDF files and embeds error descriptions as a comment using the `rdfs:comment` property.

3.4 Web Interface

The application is accessible online at: https://isl.ics.forth.gr/cidoc_services/drawioXMLToTriples/. Figure 3 displays the home page. Users can either upload their own draw.io diagram or choose from twenty preloaded examples.[13]

Figure 4 shows the resulting webpage after validation and RDF transformation have been completed. In this example, the validation process identified two issues and provides guidance on how to correct them. The entire validation and transformation process takes only a few seconds.

3.5 Demonstration Scenarios

Apart from experimenting with the twenty preloaded example diagrams, users can interact with the application through the following scenarios:

- **Scenario 1**: Create a simple diagram in draw.io, save it locally, and upload it to the Drawio2Triples application for validation and transformation into RDF. Then, examine the generated statistics and inspect the RDF triples.
- **Scenario 2**: Introduce a syntactic error in the diagram such as a property whose source or target endpoint is not properly attached to the corresponding class shape. Identify the error through validation, correct it, and retry.
- **Scenario 3**: Introduce a semantic error, such as assigning a property to a class with an invalid domain or range. Use the feedback provided by the application to identify the issue, make the necessary corrections, and retry.

[13] Developed as part of a MSc thesis at the University of West Attica, Greece.

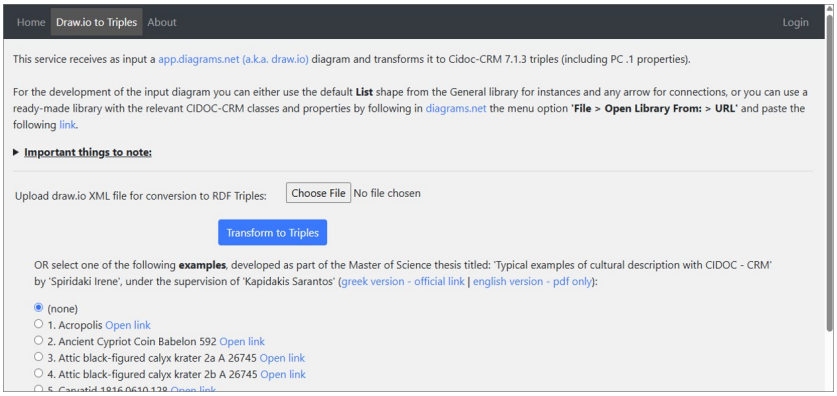

Fig. 3. The web interface of the Drawio2triples application.

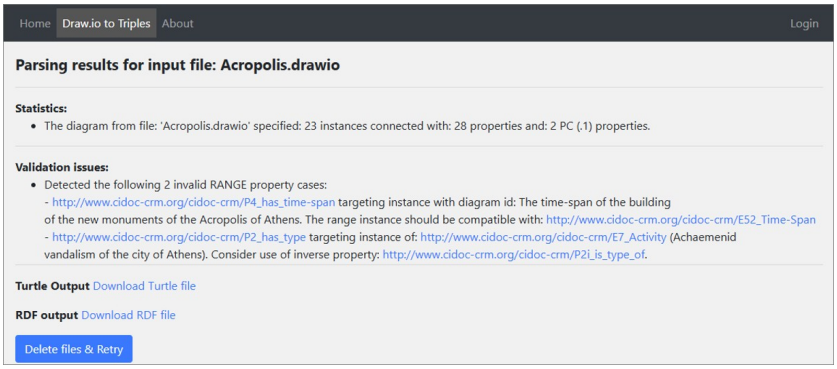

Fig. 4. The webpage after validation and transformation have been completed.

4 Conclusion and Future Work

We have presented Drawio2Triples, an application that validates and transforms ontology instantiation diagrams created with draw.io. The validation concerns the detection of semantic errors, such as the use of invalid domain or range classes for properties. The output includes statistics, description of the validation issues along with information on how to correct them, and RDF files of the ontology instances in Turtle and RDF/XML formats.

We are currently working on generalizing the application to support any input ontology. We also plan to extend it to enable the validation and transformation of diagrams representing ontologies or ontology extensions. This will facilitate the graphical construction of new ontologies and their rapid serialization.

Acknowledgments. The authors sincerely thank Martin Doerr for motivating the development of the application, as well as Irene Spiridaki and Sarantos Kapidakis (Univ. of West Attica, Greece) for their extensive use of the application, their feedback, and for contributing the examples included in the web application.

This work has received funding from the European Union under Grant Agreement No. 101157364 – ECHOES.

References

1. Antoniazzi, F., Viola, F.: Rdf graph visualization tools: A survey. In: 2018 23rd Conference of Open Innovations Association (FRUCT). pp. 25–36. IEEE (2018)
2. Camarda, D.V., Mazzini, S., Antonuccio, A.: Lodlive, exploring the web of data. In: Proceedings of the 8th International Conference on Semantic Systems. pp. 197–200 (2012)
3. Doerr, M.: The cidoc conceptual reference module: an ontological approach to semantic interoperability of metadata. AI magazine **24**(3), 75–75 (2003)
4. Fakih, G., Serrano-Alvarado, P.: A survey on sparql query relaxation under the lens of rdf reification. Semantic Web **15**(6), 2507–2554 (2024)
5. Falco, R., Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: Modelling owl ontologies with graffoo. In: The Semantic Web: ESWC 2014 Satellite Events: ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers 11. pp. 320–325. Springer (2014)
6. Häfner, P., Häfner, V., Wicaksono, H., Ovtcharova, J.: Semi-automated ontology population from building construction drawings. In: KEOD. pp. 379–386 (2013)
7. Horridge, M., Gonçalves, R.S., Nyulas, C.I., Tudorache, T., Musen, M.A.: Webprotégé: A cloud-based ontology editor. In: Companion Proceedings of The 2019 World Wide Web Conference. pp. 686–689 (2019)
8. Jupp, S., Horridge, M., Iannone, L., Klein, J., Owen, S., Schanstra, J., Wolstencroft, K., Stevens, R.: Populous: a tool for building owl ontologies from templates. BMC bioinformatics **13**, 1–12 (2012)
9. Lohmann, S., Link, V., Marbach, E., Negru, S.: Webvowl: Web-based visualization of ontologies. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 154–158. Springer (2014)
10. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with vowl. Semantic Web **7**(4), 399–419 (2016)
11. Stellato, A., Fiorelli, M., Turbati, A., Lorenzetti, T., Van Gemert, W., Dechandon, D., Laaboudi-Spoiden, C., Gerencsér, A., Waniart, A., Costetchi, E., et al.: Vocbench 3: A collaborative semantic web editor for ontologies, thesauri and lexicons. Semantic Web **11**(5), 855–881 (2020)