



Ontopanel: A Tool for Domain Experts Facilitating Visual Ontology Development and Mapping for FAIR Data Sharing in Materials Testing

Yue Chen¹ · Markus Schilling¹ · Philipp von Hartrott² · Hossein Beygi Nasrabadi¹ · Birgit Skrotzki¹ · Jürgen Olbricht¹

Received: 22 July 2022 / Accepted: 27 September 2022 / Published online: 17 October 2022
© The Author(s) 2022

Abstract

In recent years, the design and development of materials are strongly interconnected with the development of digital technologies. In this respect, efficient data management is the building block of material digitization and, in the field of materials science and engineering (MSE), effective solutions for data standardization and sharing of different digital resources are needed. Therefore, ontologies are applied that represent a map of MSE concepts and relationships between them. Among different ontology development approaches, graphical editing based on standard conceptual modeling languages is increasingly used due to its intuitiveness and simplicity. This approach is also adopted by the Materials-open-Laboratory project (Mat-o-Lab), which aims to develop domain ontologies and method graphs in accordance with testing standards in the field of MSE. To suit the actual demands of domain experts in the project, *Ontopanel* was created as a plugin for the popular open-source graphical editor diagrams.net to enable graphical ontology editing. It includes a set of pipeline tools to foster ontology development in diagrams.net, comprising imports and reuse of ontologies, converting diagrams to Web Ontology Language (OWL), verifying diagrams using OWL rules, and mapping data. It reduces learning costs by eliminating the need for domain experts to switch between various tools. Brinell hardness testing is chosen in this study as a use case to demonstrate the utilization of *Ontopanel*.

Keywords Materials testing · Ontology · Visual ontology development · Data mapping · FAIR

Introduction

With the widespread use of data-driven technologies in materials research, the sharing and integration of scientific data and knowledge generated in organizations and institutions have become a critical aspect [1]. These data are shared in heterogeneous databases and structured files with different representations, terminology, and formats, making their interpretation and use in different contexts quite difficult. To improve the management of research data, the FAIR data principles were proposed, which imply that research data should be findable, accessible, interoperable, and reusable [2]. Ontologies that map all elements of a research process

in terms of concepts and their relationships are emerging as efficient ways for putting these principles into practice. They use common human-readable vocabularies to represent knowledge while concurrently providing rich machine-processable semantic descriptions so that new information can be derived by semantic reasoning through data-driven techniques. Therefore, ontology-based solutions are well suited for developing models for product, process, or material design in the field of materials science and engineering (MSE) [3–5].

Transferring knowledge on materials testing and characterization processes into method graphs, i.e., graphs representing specific chains of processes, equipment, specifications, and objects relevant of testing procedures [6], and deriving the respective domain ontologies to incorporate semantic meaning can be challenging. Not only collaborative efforts of ontologists and domain experts are required, but methods and tools also play a key role in the modeling process. Among all approaches, graphical editing based on standard conceptual modeling languages is widely used

✉ Markus Schilling
Markus.Schilling@bam.de

¹ Bundesanstalt für Materialforschung und -prüfung (BAM),
Unter den Eichen 87, Berlin, Germany

² Fraunhofer-Institut für Werkstoffmechanik (IWM),
Wöhlerstr. 11, Freiburg, Germany

because of its intuitiveness and simplicity. In particular, the Unified Modeling Language (UML)-based language for graphical editing of Web Ontology Language (OWL) has been proposed in the scientific literature for ontology representation [7, 8]. Some tools allowing the graphical creation or modification of ontologies, such as OWLGrEd [9] (desktop version for editing, online version for visualization only) and Protégé Plugin CoModIDE [10] (desktop version), have already been proposed. Furthermore, other promising options such as a PURO language-based tool [11, 12] are under development.¹ In the current work, the *Ontopanel* tool is introduced, which primarily aims at facilitating ontology development.

Ontopanel was developed within the framework of the Materials-open-Laboratory (Mat-o-Lab) project, which aims to develop domain ontologies for typical characterization methods in the field of MSE, predominantly in accordance with testing standards. In this respect, actual case studies reveal that many needs cannot be met by current solutions, such as reuse of ontologies, ontology conversion and data mapping.

Domain experts must switch inconveniently between different tools to achieve a certain step, which invariably increases learning costs. Furthermore, the lack of validation can lead to incorrect method graphs created by domain experts with limited experience in ontology rules. *Ontopanel* addresses these issues by providing a custom-designed tool in the *diagrams.net* environment,² a popular and open-source web graphical editor.

Modeling Approach

The Mat-o-Lab project is a joint project between the Fraunhofer Group for Materials and Components (MATERIALS) and the Bundesanstalt für Materialforschung und -prüfung (BAM), which focuses on the digitization of materials and components throughout their life cycle [6]. One of the tasks within the Mat-o-Lab project is the semantic modeling of mechanical testing processes and generating respective datasets according to testing standards for the field of MSE. Associated ontologies are published in the open-access platform “Matportal.”³

In this paper, the modeling process of the Brinell hardness testing, a well-known test for the characterization of metallic materials, is selected as a use case to show the application of the *Ontopanel* tool, which is described in “Application of

Ontopanel in Brinell Hardness Testing Modeling” section, along with the modeling results.

Testing Method Modeling

In the project, the considered mechanical testing methods are modeled using the Materials Science and Engineering Ontology (MSEO) which was created for the MSE domain.⁴ MSEO uses the Basic Formal Ontology (BFO)⁵ as the top-level and the Common Core Ontology (CCO)⁶ as the mid-level and will be further extended by, e.g., ontologies from the fields of mechanical testing and material properties created by the Mat-o-Lab workgroups [6].

Mechanical testing standards define the terms, symbols, and processes of a testing method and thereby provide a direction for structuring and classifying the required entities and concepts. Moreover, general guidelines for materials characterization data (CHADA) were proposed in a workshop agreement (CWA 17815) [13].⁷ Accordingly, such documents are considered as guidelines for domain ontology development and method modeling. In addition, manuals, standard operating procedures (SOPs), test protocols and the data structure of typical experimental data files plus the valuable experience of domain experts are essential to successfully model the complete testing procedure with all steps and elements.

Besides the extension of domain ontologies, the storage and sharing of experimental data are another goal of modeling since semantic data are designed for facilitated sharing. To achieve this, datasets from file formats such as comma-separated values (CSV) are mapped to method graphs to reproduce the results in the model. These datasets can then be converted to Resource Description Framework (RDF)⁸ format and saved in a database for querying with SPARQL Protocol and RDF Query Language (SPARQL) [6].⁹

Testing methods usually tend to be modeled graphically due to their complexity. To model graphically and semantically, this project takes a combined UML and ontology approach using *diagrams.net*, which was originally proposed in the Chowlk project [14]. *Diagrams.net* is a visual editing tool that allows users to customize their own library of shapes and draw diagrams in the first place. The Chowlk

¹ PURO tool website: <https://protegeserver.cz/purom5/#>

² Diagrams.net website: <https://www.diagrams.net/>

³ Matportal website: <https://matportal.org/>

⁴ MSEO repositories: <https://github.com/Mat-O-Lab/MSEO> and <https://matportal.org/ontologies/MSEO>

⁵ BFO website: <https://basic-formal-ontology.org/>

⁶ CCO GitHub repository: <https://github.com/CommonCoreOntology/CommonCoreOntologies>

⁷ CWA 17,815:2021 (E) document: <https://www.cenelec.eu/media/CEN-CENELEC/CWAs/ICT/cwa17815.pdf>

⁸ RDF document: <https://www.w3.org/RDF/>

⁹ SPARQL document: <https://www.w3.org/TR/rdf-sparql-query/>

project proposes a library for ontology conceptualization that consists of a set of shapes representing each element of the OWL specification. Therefore, diagrams constructed with this library are semantic and can be converted to OWL.

Use Case: Brinell Hardness Testing

The Brinell hardness test of metallic materials described in ISO 6506-1:2014 [15] is one of the standardized mechanical tests that are to be modeled in the Mat-o-Lab project.

To perform the test, a spherical indenter (tungsten carbide ball) is pressed into the surface of a test piece and the diameter of the created indentation, which remains on the test surface after the test force has been removed, is measured. The characteristic value of Brinell hardness is proportional to the quotient calculated from the test force divided by the curved surface of the indentation. The indentation is assumed to take the shape of the unloaded spherical indenter (ball), and its surface area is calculated from the mean indentation diameter and the ball diameter. Hence, the important parameters of the determination of Brinell hardness are (i) the diameter of the indenter D , (ii) the test force F , (iii) the mean indentation diameter d arithmetically averaged of two measurements that have been performed perpendicular to each other (d_1 , d_2). These and additional parameters are to be considered in a method graph that is supposed to semantically describe the measurement process.

In accordance with the Mat-o-Lab approach, the MSEO ontology, the respective testing standard ISO 6506-1:2014 and a full dataset¹⁰ containing measurement information and results are used for the process modeling of the Brinell hardness testing and data mapping. The available data to be included resulted from Brinell hardness tests of thirty-one aluminum test pieces (alloy EN AW-2618A in T61 condition) that were further annealed at different aging temperatures for different aging times. Each test piece was tested five times at room temperature using a hardness testing machine (Model M4C 025 G3 EMCO-TEST Prüfmaschinen GMBH, Kuchl, Austria). Tests were performed using an indenter with a diameter of 2.5 mm and a test force of 612.9 N. The diameters of indentations in the test pieces were measured in two directions, respectively, and the average indentation diameter was determined. The Brinell hardness value of each test piece was then calculated from the test force and the (idealized) indentation surface in accordance with the standard. Finally, the individual results of five indentations were averaged to obtain a representative Brinell hardness value. Furthermore, measurement uncertainties were calculated to assess the final Brinell hardness during the test.

¹⁰ Brinell hardness dataset address: <https://doi.org/10.5281/zenodo.6787085#Yr76oHC6n7Y.mailto>

Ontopanel Features

Ontopanel is a diagrams.net web-based plugin that helps domain experts to build domain ontologies and method graphs in a simple and user-friendly way. It was designed originally to address the inconveniences that domain experts encounter when building domain ontologies using the Mat-o-Lab approach. However, its use is neither limited to the MSE domain nor to a particular project. With this plugin, diagrams.net turns into a visual editing tool suitable for ontology development and data mapping.

It contains three tools: (i) *Library*, (ii) *EntityManager*, and (iii) *Converter*. Users can build ontology graphs using shapes provided in *Library* and search and reuse entities from external ontologies in *EntityManager*. Then, diagrams can be converted into OWL applying *Converter*, that includes validation and data mapping options. The service¹¹ is available online, and the source code for the front-end¹² and the back-end is shared in the GitHub repositories under the Apache-2.0 license.¹³

Library

The *Ontopanel-Library* is an xml library for ontology conceptualization that provides a set of shapes (nodes and arcs) to represent each element of the OWL specification. Any method graph constructed with this library can be converted in OWL using the *Ontopanel-Converter*.

The *Ontopanel-Library* is based on the Chowlk library,¹⁴ which has already been used extensively in the Mat-o-Lab project. For compatibility with previous work, the *Ontopanel-Library* is a simple extension to the Chowlk library. It redesigns the shapes in Chowlk by adding a data object to each shape. The shapes data object stores additional information that can be interpreted during conversion by means of key-value pair attributes and is displayed through hovering the mouse over the shape as shown in Fig. 1.

The reserved attributes are "Type," "IRI_XX" and "Mapping_XX," each of which is especially designed for different purposes. The "Type" attribute shows the OWL meaning of the shape to enhance readability of the constructed diagrams. It is also used for shape identification in the *Ontopanel-Converter*. The "IRI_XX" attribute allows saving the

¹¹ Ontopanel demo on GitHubPage: <https://github.com/yuechenbam/yuechenbam.github.io>

¹² Ontopanel front-end GitHub repository: <https://github.com/yuechenbam/Ontopanel-frontend>

¹³ Ontopanel back-end GitHub repository: <https://github.com/yuechenbam/Ontopanel-back-end>

¹⁴ Chowlk notation website: <https://chowlk.linkeddata.es/notation.html>

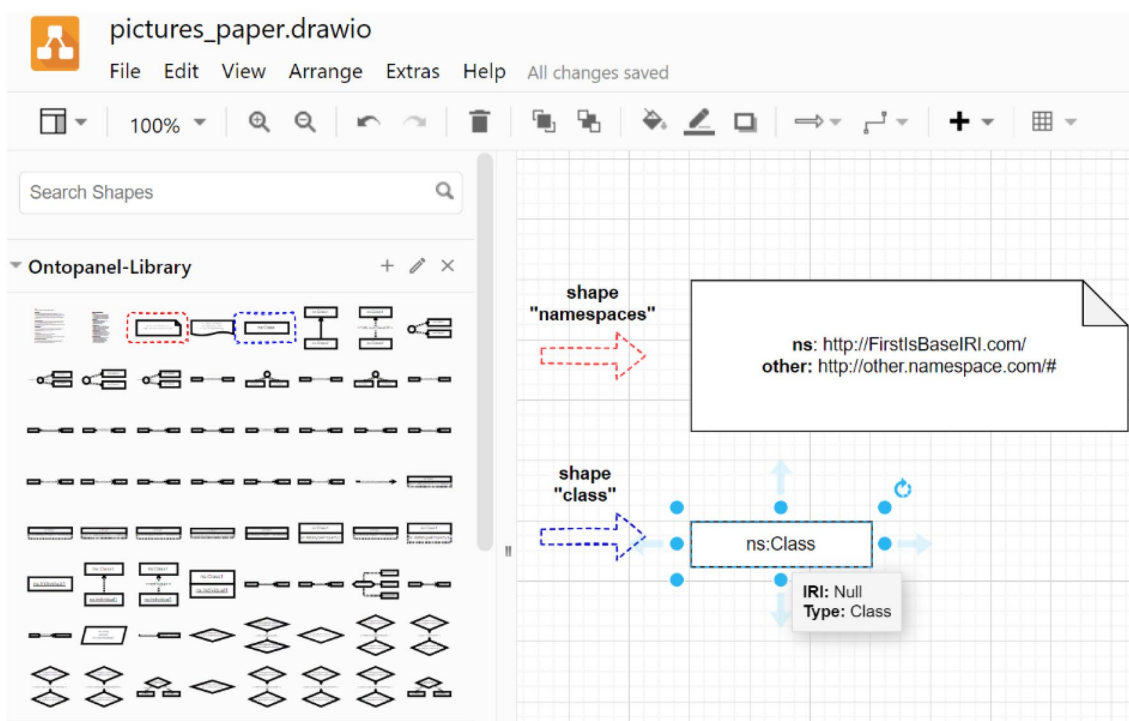


Fig. 1 *Ontopanel-Library* with attributes in shapes. *Ontopanel-Library* contains shapes that represent OWL elements (left side). Each shape contains shape data to store additional information such as its type and IRI (example shown in right side)

Internationalized Resource Identifier (IRI) of an entity in the shape data while displaying the entity with other labels (e.g., `rdfs:label`), which is helpful in cases where the entity is named numerically in the ontology. If the value of the "IRI_XX" attribute is not null, the *Ontopanel-Convertor* prefers to use it as the IRI of the entity; otherwise, it takes the IRI short name in the shape label instead. The "Mapping_XX" attribute marks entities that require data mapping and store mapping information such as the mapping column, which will be explained in the "Convertor" section.

EntityManager

In most practical cases, users need to reuse entities from existing ontologies such as MSEO for modeling rather than just creating new ones. Therefore, users must read and search entities of different ontologies in another tool (e.g., Protégé) and copy their IRI to diagram.net for modeling, which is prone to adding errors and inconvenient. To solve these issues, the *Ontopanel-EntityManager* was designed. It allows users to import ontologies via local files or remote Uniform Resource Locators (URLs) and displays ontologies with IRI short names or `rdfs:label` in the hierarchy, as shown in Fig. 2. Users can click on an entity name to show its details, including annotations and constraints, and can also search for entities. In addition, *EntityManager* provides a user registration system and a database for authentication

and ontology storage. When users log in, their stored ontologies are automatically loaded.

The *Ontopanel-EntityManager* can also export entities directly to the method graph. Each entity displayed in the *EntityManager* is implemented with an insert button "Δ" and a replace button "R." By clicking the insert button, the entity is added to the method graph in the corresponding shape from the *Ontopanel-Library*. For example, a class entity is exported in the rectangular shape representing the OWL entity Class. Clicking the replace button replaces labels of the selected shapes and their IRI attributes with the entity. In this way, users do not need to copy the entity from another tool into the graph, thus avoiding typos.

Convertor

The *Ontopanel-Convertor* is used to convert the method graph into the OWL. The *Convertor* tool analyzes shapes and their relationships in the method graph, converts them to OWL triples and displays the results in RDF/XML¹⁵ or Turtle (TTL) format.¹⁶ Users can also search for text and download the corresponding file. In this way, users can view

¹⁵ RDF/XML document: <https://www.w3.org/TR/rdf-syntax-grammar/>

¹⁶ Turtle document: <https://www.w3.org/TR/turtle/>

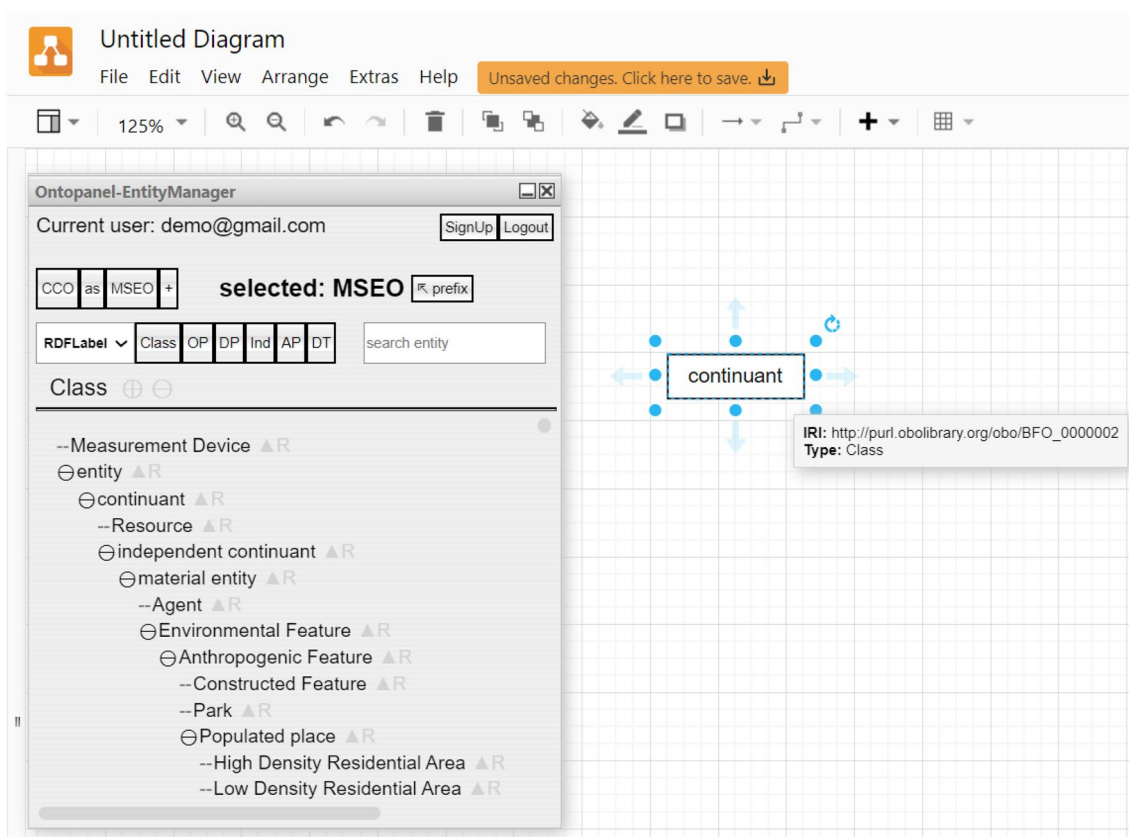


Fig. 2 Functionality of the *Ontopanel-EntityManager*. The *EntityManager* window with entities in `rdfs:label` is displayed on the left. The exported example entity in the diagram is displayed using its `rdfs:label` and has its IRI saved in the shape data


the conversion results while modifying the graph without switching between different tools. Moreover, two additional functions can be performed during the conversion process: (i) validation and (ii) data mapping.

Validation

During the modeling process, mishandling such as disconnections between shapes, naming errors, and the unfamiliarity of users with OWL rules might lead to errors. To avoid such, a validation algorithm for error detection is executed during the conversion process. The *Converter* returns and displays errors as clickable messages in the "Show Errors" window, as shown in Fig. 3. These errors are divided into four groups: general errors, node errors, edge errors, and relation errors. In general errors, unknown shapes in the method graph, i.e., such that are not taken from *Ontopanel-Library* are listed and general information such as whether the dataset for data mapping was uploaded is provided. Node errors include missing namespaces and improper IRIs. Similarly, errors referring to edges are shown as edge errors. In addition, missing connections are highlighted, as disconnections are common during modeling. Relation errors display

incorrect combinations that violate the OWL rules. When the user clicks on an error message, the shape containing the error is highlighted and pushed to the center of the view for quick retrieval.

Data Mapping

The data mapping feature of the *Ontopanel-Converter* enables the combination of an experimental dataset and a method graph in a simple and intuitive way. Users simply upload a tabular dataset in CSV or XLS(X) format in the mapping window and then link an individual or literal in the method graph to one of the columns in the dataset by clicking the  button in the column header. The selected individual or literal is then automatically marked in red in the graph to indicate that it was transformed to an individual- or literal-template which will be mapped during conversion.

The working principle is shown in Fig. 4. After linking a column and an entity, the "MappingCol" keyword and the column name are stored as a key-value pair attribute in the shape data of this entity. Once *Converter* detects this keyword, it clones this entity multiple times depending on the

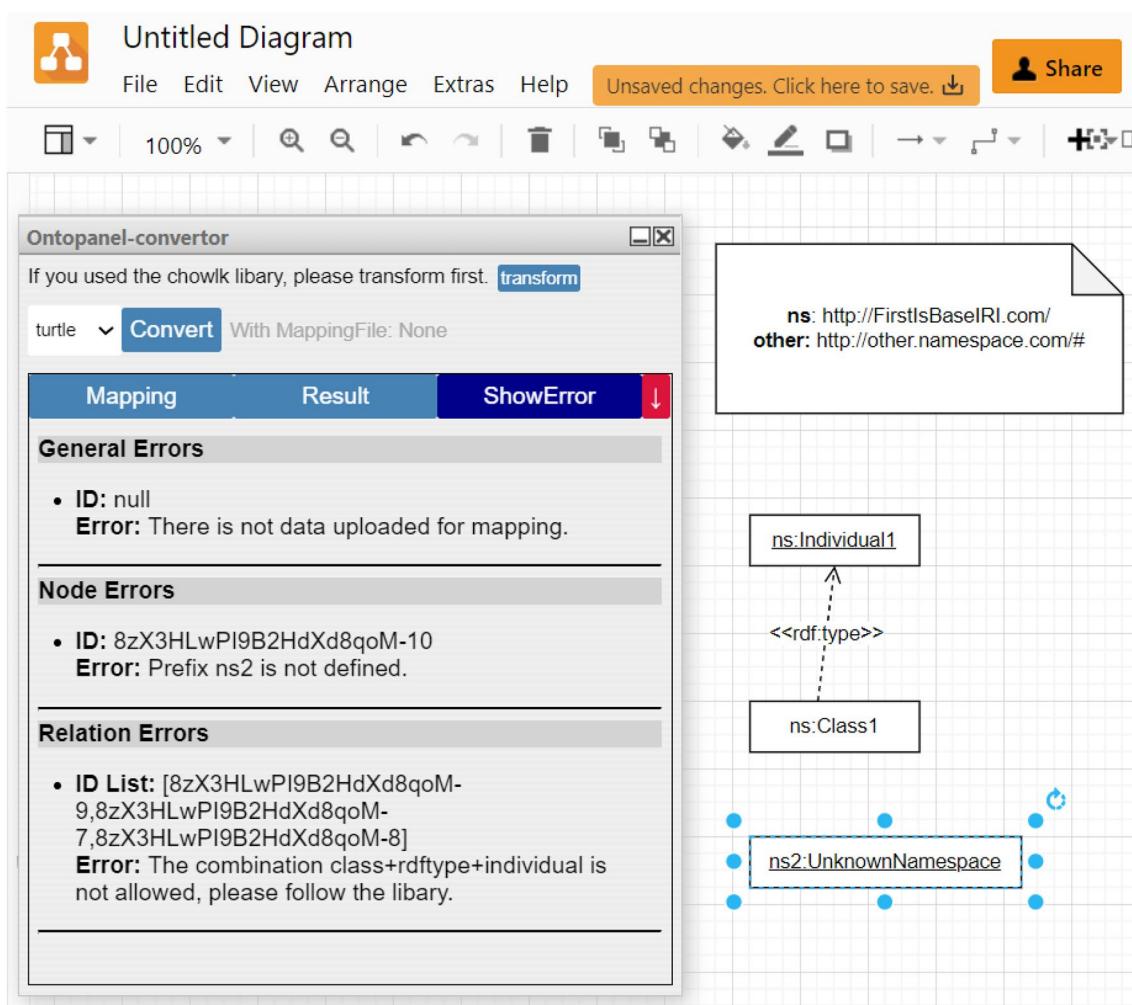


Fig. 3 *Ontopanel-Convertor* detects and locates the errors in the diagram. The errors in the diagram are detected and displayed in *Convertor*. The namespace "ns2" is not defined, so the shape using this namespace is recognized as a node error

length of the mapped column, cf. Case A in Fig. 4. These new entities are automatically named according to their type. If this original entity is an OWL individual element, as shown in the blue color in the figure, the IRI of each new entity is a string concatenation of the IRI of the original entity and the value of the corresponding cell of the mapped column. If this original entity is an OWL literal element, as shown in the orange color, the value of each new entity is replaced with the value of the corresponding cell in the mapped column.

The mapping relations such as one-to-many are defined in accordance with the dataset. As shown in Case B in Fig. 4, the "name" column has the same value and all the cloned entities are named "ns:individual_A" after mapping. Since entities created with the same IRIs are the same entity, a one-to-three mapping relationship is achieved in this way.

Ontopanel Architecture

The architecture of the *Ontopanel* plugin is shown in Fig. 5. When the user adds the plugin into *diagrams.net*, *Entity-Manager* and *Convertor* are loaded as floating windows, while *Library* is added to the sidebar. The plugin interacts with a back-end through REST APIs, which is mainly used for data processing, data storage, and authentication.

The plugin (front-end) is implemented in JavaScript, bundled, and minified into a single file using bundling tools such as Webpack.¹⁷ The main library used to interact with *diagrams.net* is *mxGraph* library.¹⁸

¹⁷ Webpack website: <https://webpack.js.org/>

¹⁸ MxGraph document: <https://jgraph.github.io/mxgraph/>

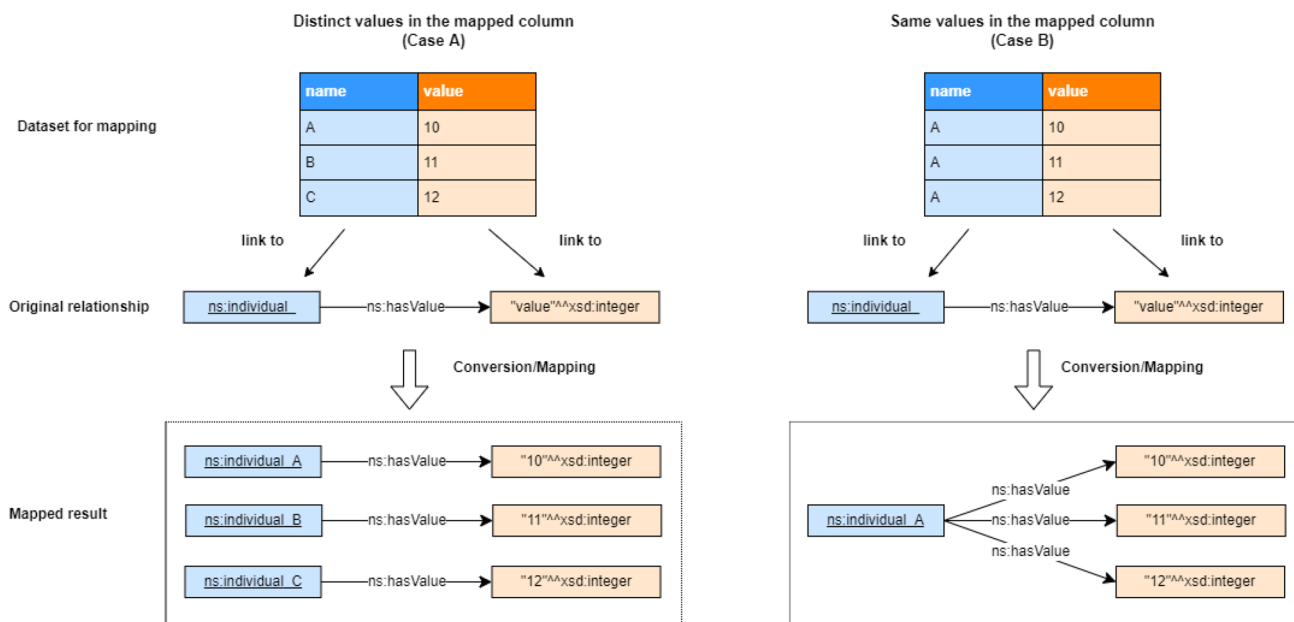
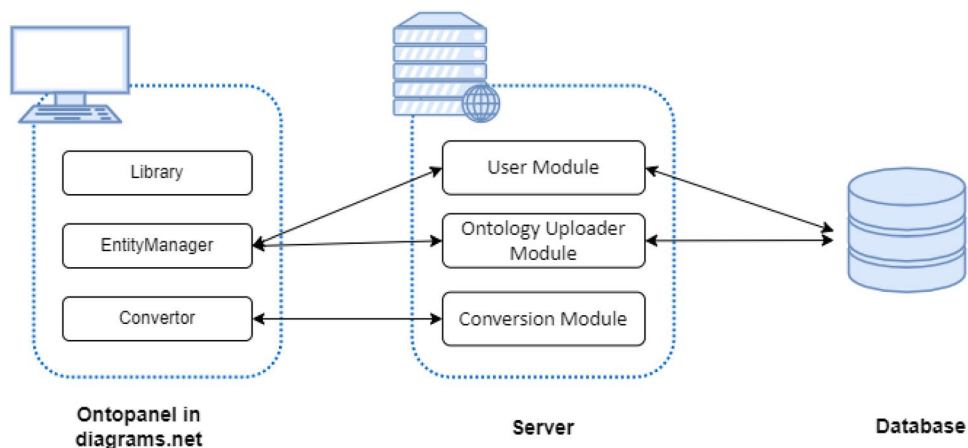


Fig. 4 Illustration of data mapping principle. Case A shows a one-to-one mapping because the values of the "name" columns are different, while Case B shows a one-to-three mapping because the values of the "name" columns are the same

Fig. 5 *Ontopanel* architecture. *Ontopanel* as front-end with its three tools (shown on the left) communicates with the server (center) via REST APIs. User information and uploaded ontologies are stored in a database (right)



The back-end is built using the Django REST framework.¹⁹ The main library for handling ontologies is RDFLib.²⁰ The back-end contains a database (Fig. 5 right) and three main modules (Fig. 5 center): the user module, the ontology uploader module, and the conversion module. The user module and the ontology uploader module communicate with *EntityManager* (Fig. 5 left). The ontology uploader module performs ontology detail extraction. It first executes a loop to parse the file or URL sent from *EntityManager* and its imported ontologies and

then extracts and saves the details of each entity in the Manchester syntax. Finally, all entities are sorted into a hierarchy and returned in JSON format to be displayed in *EntityManager*. The user module is for authentication. Registered users can save ontologies in the database and load them automatically to *EntityManager* when they log in.

The conversion module works for *Convertor* (Fig. 5 left). *Convertor* extracts the method graphs built with the *Ontopanel-Library* into JSON format as input to the conversion module. The conversion module then analyzes the entities and their relationships and performs error detection and OWL rule validation. If the dataset for data mapping is not empty, data mapping is performed during the conversion process. Finally, the entities and their relationships are

¹⁹ Django REST framework website: <https://www.django-rest-framework.org/>

²⁰ RDFLib website: <https://github.com/RDFLib/rdfliib>

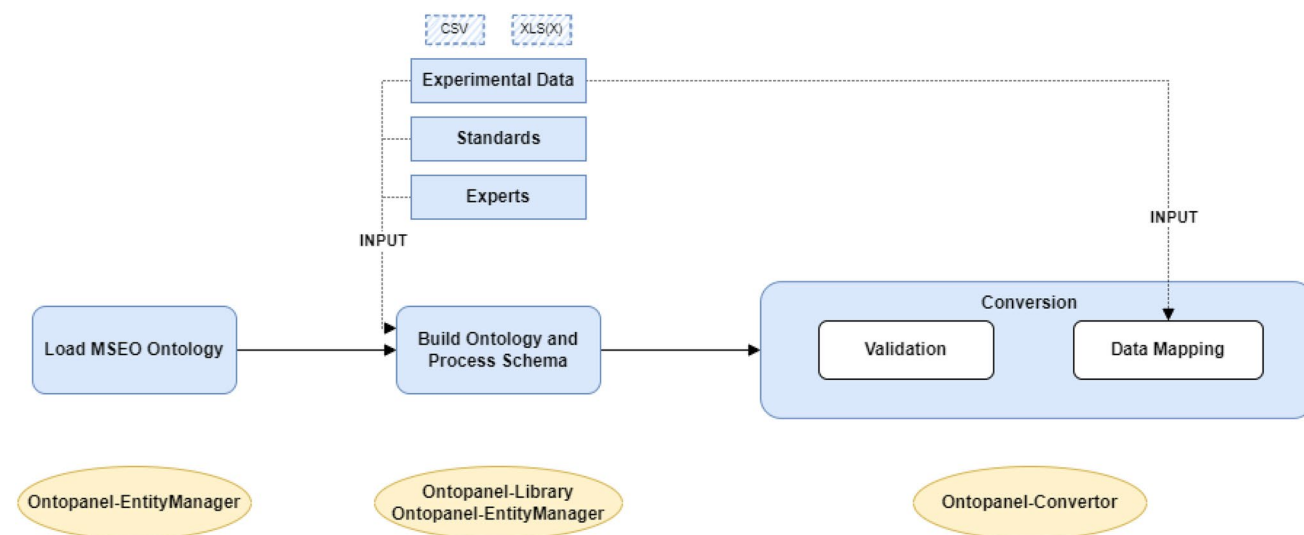


Fig. 6 Procedure for ontology development with *Ontopanel*

serialized into OWL and returned in JSON format along with the detected errors to be displayed in different windows of *Convertor*.


Application of Ontopanel in Brinell Hardness Testing Modeling

The modeling steps of the Brinell hardness testing can be summarized and illustrated in Fig. 6. It consists of the following three primary steps: loading MSEO ontology, creating the Brinell hardness-related ontology entities and process schema in the method graph using MSEO, and converting the method graph into an output OWL file. *Ontopanel's* toolbox in diagrams.net can assist the entire modeling process.

Based on the Mat-o-Lab modeling approach, MSEO is used as an upper-level domain ontology to express general concepts such as material composition and test piece. Therefore, the first step is to load MSEO using *EntityManager* for quick search and export (Fig. 6 left). The second step is to create the process and the associated ontology for the test (Fig. 6 center). Entities that represent Brinell hardness testing terms are created in the method graph using shapes from *Ontopanel-Library* and are connected to MSEO via OWL relationship shapes to extend MSEO. All the entities used are based on the testing standard ISO 6506–1:2014, as well as on the experimental dataset of Brinell hardness and the experience of domain experts.

Figure 7 shows the modeling of material state: The test pieces used in the Brinell test have different IDs (1) and aging conditions (2) including aging temperature (3) and aging time (4), but the same original material composition

(5). The entities with the namespace “base” are created locally as individuals to describe the material states of the test pieces, while entities with the namespaces “cco” and “mid” are added from *EntityManager*. The orange boxes represent literals from the experimental dataset.

After completing the method graph, *Convertor* comes into play (Fig. 6 right). Conversion is usually applied first for validation to get a clean method graph. The method graph as depicted only describes one test piece and its processing during the performance of the Brinell testing method. Hence, in order to apply this to the whole experimental dataset containing numerous test pieces, data mapping must be performed. The experimental dataset is first transformed into a suitable table format and then uploaded to the mapping window of *Convertor* as shown in Fig. 8a. Mapping relationships are established by selecting the entities in the method graph and clicking the  button in the column header in *Convertor*. Here, the “Test piece ID” column is selected to link all individuals representing the test piece and the material condition, i.e., the individual-templates next to the green marked box. After conversion, these individual-templates are cloned and renamed according to each test piece ID. Since all test pieces have different IDs, each test piece can be retrieved by a simple property path. The literals in orange are mapped to the corresponding values in the tabular dataset, while the remaining entities representing material compositions are not mapped, as they are the same for all test pieces. Therefore, the method graph is extended to include all test pieces with different aging conditions and the same material compositions after data mapping as shown in Fig. 8b.

In the method graph, the measurement process and the results are also modeled. A challenge is to model the

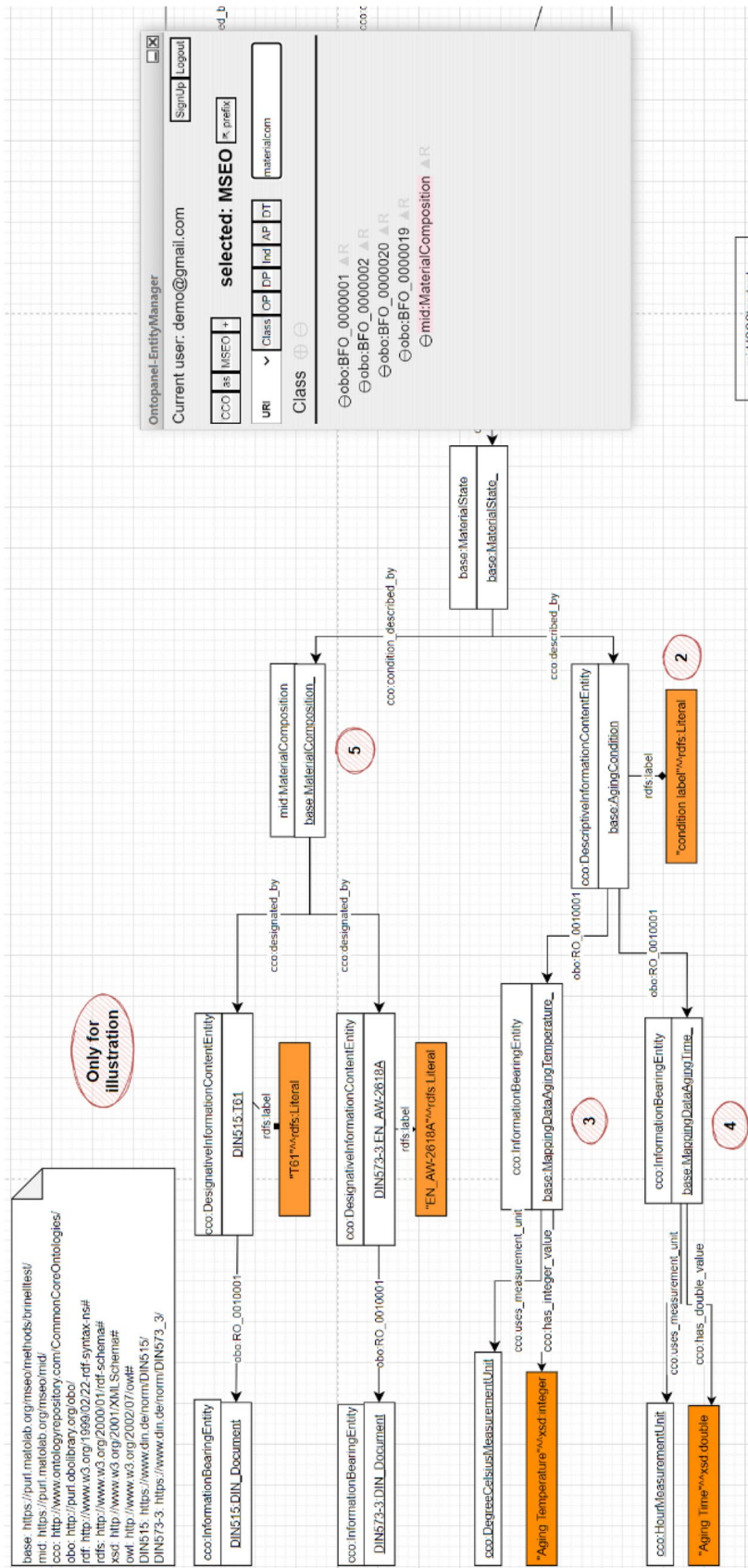


Fig. 7 Modeling of material state with Ontopanel-EntityManager. Entities with namespace “base” are created with Ontopanel-Library, while others are exported from EntityManager

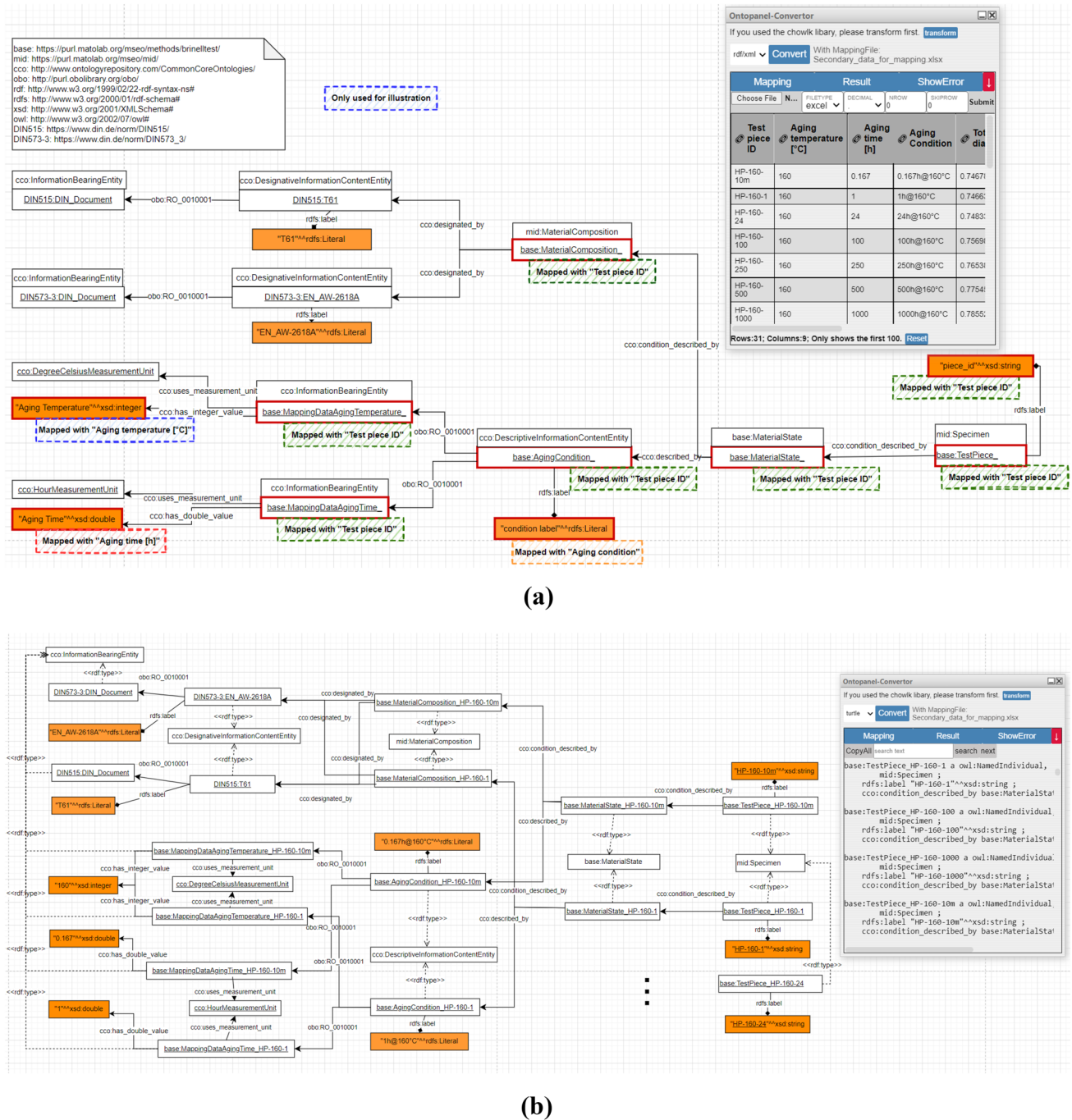


Fig. 8 Data mapping **a** and illustration of conversion result **b**. The method graph is extended to include all test pieces with different aging conditions and same material compositions through data mapping

relationship between the test piece and its measurement results. Since each test piece is measured and Brinell hardness calculated five times to obtain the final average Brinell hardness, there is a one-to-five relationship between each test piece and its relative measurements. Instead of establishing such a relationship in the method graph, it can be solved by modifying the dataset used for the data mapping.

As shown in Fig. 9, the test piece-related data (blue) in the dataset is replicated five times to match the length of the measurement-related data (green). Any test piece-related entities (blue) in the method graph are mapped with “Test piece ID” and any measurement-related entities (green) are mapped with “Measurement ID.” Since the test piece-related data (blue) are the same for every five rows and entities with

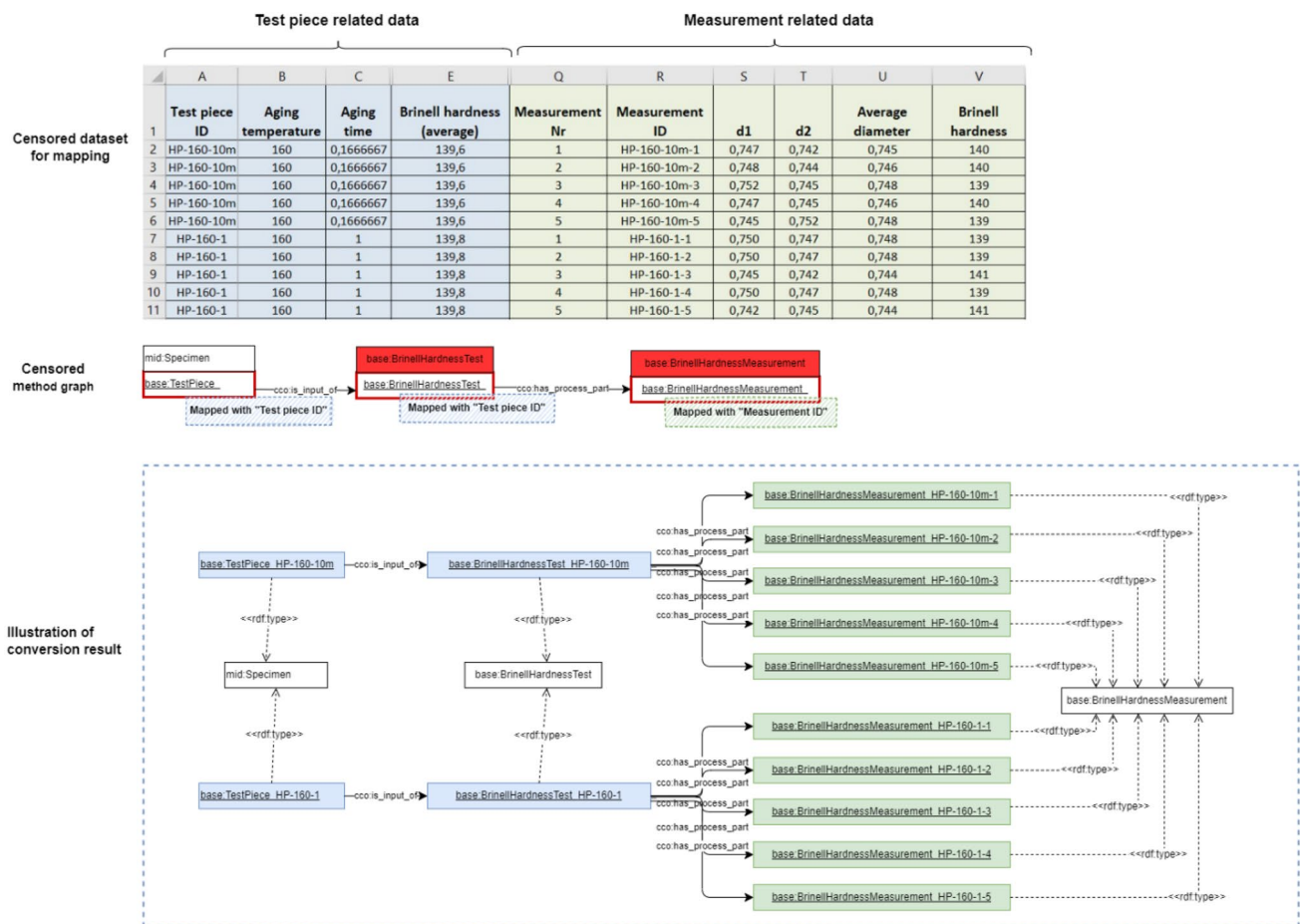


Fig. 9 Data mapping for one-to-many relationship. In the data mapping, a one-to-five relationship between a test piece and its measurements is established by copying the test piece-related data five times to meet the length of the measurement-related data

the same IRIs are identical, so after conversion, the test piece-related entities (blue) mapped with each five rows are one entity. In contrast, no row in the "Measurement ID" is the same, so the measurement-related entities (green) are all different. Thus, a one-to-five relationship arises as a result.

Finally, the method graph including the mapped dataset is converted and downloaded in TTL or RDF/XML file format in *Converter*. The results can be uploaded to a database for querying such as the triple store of the Mat-o-Lab project.

The full dataset is available online.²¹

Conclusions and Future Work

This paper presents the structure, operation, and application of the newly developed *Ontopanel* plugin for ontology development and method modeling in the MSE domain.

It enables

- The creation of new ontologies and construction of method graphs from existing ontologies. Its module *Ontopanel-Library* provides a set of shapes representing OWL elements for construction, while *Ontopanel-EntityManager* allows to upload and display ontologies in hierarchy and add entities directly to the method graph so that domain experts can search and reuse these ontologies in a simple way.
- Graph validation and conversion to OWL. *Ontopanel-Converter* performs the conversion at a fast speed and identifies graph errors, such as disconnections and relationships that contradict OWL rules, considerably assisting domain experts unfamiliar with OWL rules in getting the proper results in less time.
- Data mapping. The data mapping feature integrated in *Ontopanel-Converter* enables the combination of the experimental dataset and the method graph. Following mapping, the method graph is expanded to reflect big real datasets, allowing for scientific data exchange.

²¹ Data archive in GitHub repository: <https://github.com/BAMresearch/Ontopanel-BrinellHardnessData>

Overall, *Ontopanel* enables graphical ontology creation in diagram.net and provides evident benefits such as avoiding tool switching and lowering domain expert learning costs.

In its current state, the *Ontopanel* tool already promotes the access of MSE domain experts to semantics-based data structuring—and, in turn, it helps to increase the acceptance of these concepts in the MSE community. To further enhance its performance, the next development steps will include an integration of SPARQL queries and the establishment of communication with graph databases so that users can update their data stores more conveniently. It is also planned to develop an *Ontopanel*-based system to automate the storage of experimental data. Once the method graph is established, new experimental datasets could then be stored continuously in a semantic manner (RDF format) by automatic data mapping.

Acknowledgments Funding provided by Bundesanstalt für Materialforschung und -prüfung (BAM) and Fraunhofer Group Materials and Components (MATERIALS) is gratefully acknowledged in the framework of the project ‘Materials-open-Lab (Mat-o-Lab)’. The authors wish to thank Thomas Hanke (Fraunhofer IWM) and Alexandru Todor (Fraunhofer MATERIALS) for technical support and fruitful discussions. The financial support of the ‘Innovation platform MaterialDigital (PMD)’ project by the Federal Ministry of Education and Research [grant number 13XP5094E] enabling the participation of M.S. is gratefully acknowledged.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Ghiringhelli LM, Carbogno C, Levchenko S et al (2017) Towards efficient data exchange and sharing for big-data driven materials science: metadata and data formats. *Npj Comput Mater* 3:46. <https://doi.org/10.1038/s41524-017-0048-5>
- Wilkinson MD, Dumontier M, Aalbersberg IJ et al (2016) The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* 3:160018. <https://doi.org/10.1038/sdata.2016.18>
- Cheung K (2008) Towards an ontology for data-driven discovery of new materials. In: Semantic scientific knowledge integration AAAI/SSS workshop. The Association for the Advancement of Artificial Intelligence (AAAI), Stanford University, Palo Alto, CA, p 6
- Ashino T (2010) Materials ontology: an infrastructure for exchanging materials information and knowledge. *Data Sci J* 9:54–61. <https://doi.org/10.2481/dsj.008-041>
- Glick J (2013) Ontologies and databases—knowledge engineering for materials informatics. *Informatics for materials science and engineering*. Elsevier, Amsterdam, pp 147–187
- Bayerlein B, Hanke T, Muth T et al (2022) A perspective on digital knowledge representation in materials science and engineering. *Adv Eng Mater*. <https://doi.org/10.1002/adem.202101176>
- Na HS, Choi OH, Lim JE (2006) A method for building domain ontologies based on the transformation of UML models. In: Fourth international conference on software engineering research, management and applications (SERA'06). IEEE, Seattle, WA, USA, pp 332–338
- Mejhed Mkhini M, Labbani-Narsis O, Nicolle C (2020) Combining UML and ontology: an exploratory survey. *Comput Sci Rev* 35:100223. <https://doi.org/10.1016/j.cosrev.2019.100223>
- Liepinš R, Grasmanis M, Bojars U (2014) OWLGrEd ontology visualizer. In: Proceedings of the 2014 international conference on developers, pp 37–42
- Shimizu C, Hammar K (2019) CoModIDE—the comprehensive modular ontology engineering IDE. In: ISWC 2019 satellite tracks. CEUR-WS, p 5
- Dudas M, Hanzal T, Svatek V, Zamazal O (2015) OBOWLMorph: starting ontology development from PURO background models. In: 12th international experiences and directions workshop on OWL (OWLED), pp 14–20. Springer International Publishing Ag, Bethlehem
- Hanzal T, Svatek V, Vacura M (2016) Event categories on the semantic web and their relationship/object distinction. In: 9th international conference on formal ontology in information systems (FOIS), pp 183–196. Ios Press, Annecy, France
- CWA 17815:2021 E (2021) Materials characterization—terminology, metadata and classification, European Committee for Standardization (CEN-CENELEC), Brussels, Belgium
- Chávez-Feria S, García-Castro R, Poveda-Villalón M (2021) Converting UML-based ontology conceptualizations to OWL with chowlk. In: Verborgh R, Dimou A, Hogan A et al (eds) The semantic web: ESWC 2021 satellite events. Springer International Publishing, Cham, pp 44–48
- ISO 6506-1:2014-10 (2014) Metallic materials—Brinell hardness test—Part 1: Test method, International Organization for Standardization (ISO), Geneva, Switzerland

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.