Project Acronym:	**BIMERR**
Project Full Title:	**BIM-based holistic tools for Energy-driven Renovation of existing Residences**
Grant Agreement:	**820621**
Project Duration:	**45 months**

DELIVERABLE D4.5
BIMERR Building Semantic Modelling tool 2

Deliverable Status:	**Final**
File Name:	**BIMERR_D4.5-v1.00**
Due Date:	**30/06/2021 (M30)**
Submission Date:	**30/06/2021 (M30)**
Task Leader:	**Suite5 (T4.3)**

Dissemination level	
Public	x
Confidential, only for members of the Consortium (including the Commission Services)	

This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621

The BIMERR project consortium is composed of:		
FIT	Fraunhofer Gesellschaft Zur Foerderung Der Angewandten Forschung E.V.	Germany
CERTH	Ethniko Kentro Erevnas Kai Technologikis Anaptyxis	Greece
UPM	Universidad Politecnica De Madrid	Spain
UBITECH	Ubitech Limited	Cyprus
SUITE5	Suite5 Data Intelligence Solutions Limited	Cyprus
HYPERTECH	Hypertech (Chaipertek) Anonymos Viomichaniki Emporiki Etaireia Pliroforikis Kai Neon Technologion	Greece
MERIT	Merit Consulting House Sprl	Belgium
XYLEM	Xylem Science And Technology Management Gmbh	Austria
CONKAT	Anonymos Etaireia Kataskevon Technikon Ergon, Emporikon Viomichanikonkai Nautiliakon Epicheiriseon Kon'kat	Greece
BOC	Boc Asset Management Gmbh	Austria
BX	Budimex Sa	Poland
UOP	University Of Peloponnese	Greece
UEDIN	University of Edinburgh	United Kingdom
NT	Novitech As	Slovakia
FER	Ferrovial Agroman S.A	Spain
UCL	University College London	United Kingdom

Disclaimer

BIMERR project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°820621. The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Commission (EC). EC is not liable for any use that may be made of the information contained therein.

Authors List

Leading Author (Editor)			
Surname	**First Name**	**Beneficiary**	**Contact email**
Lampathaki	Fenareti	Suite5	fenareti@suite5.eu

Co-authors (in alphabetic order)				
#	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1	Bikas	George	Suite5	gbikas@suite5.eu
2	Bountouni	Nefeli	Suite5	nefeli@suite5.eu
3	Giannakis	Giorgos	HYPERTECH	g.giannakis@hypertech.gr
4	González-Gerpe	Salvador	UPM	salvador.gonzalez.gerpe@upm.es
5	Kousouris	Spiros	Suite5	spiros@suite5.eu
6	Poveda-Villalón	María	UPM	mpoveda@fi.upm.es
7	Vafeiadis	Giorgos	UBITECH	gvafeiadis@ubitech.eu
8	Vergeti	Danai	UBITECH	vergetid@ubitech.eu

Reviewers List

List of Reviewers (in alphabetic order)				
#	**Surname**	**First Name**	**Beneficiary**	**Contact email**
1	Chávez-Feria	Serge	UPM	schavez@delicias.dia.fi.upm.es
2	Giorgos	Vafeiadis	UBITECH	gvafeiadis@ubitech.eu

Revision Control

Version	Author	Date	Status
0.10	Suite5	29/04/2021	Draft ToC
0.20	Suite5	27/05/2021	Draft Section 1,2,4,5
0.30	UPM	09/06/2021	Draft Section 3
0.40	Suite5	09/06/2021	Draft Section 1,2,3,4,5
0.50	Suite5	14/06/2021	Draft version circulated for Internal Quality Check
0.60	Suite5	28/06/2021	Updates to address the comments received during the Internal Quality Check
1.00	Suite5	30/06/2021	Final draft for submission to the EC

Table of Contents

LIST OF FIGURES

List of Tables

ACRONYMS

Acronym	Meaning
API	Application Programming Interface
BIF	BIMERR Interoperability Framework
BIMERR	BIM-based holistic tools for Energy-driven Renovation of existing Residences
BO2DM	BIMERR Ontology to Data Model
GUI	Graphical User Interface
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
GUI	Graphical User Interface
MM	Model Mapper
MLM	Model Lifecycle Manager
OMF	Ontology Manager Framework
OWL	Ontology Web Language
XML	Extensible Markup Language

EXECUTIVE SUMMARY

The purpose of the present documentation for the BIMERR Deliverable D4.5 "BIMERR Building Semantic Modelling tool 2" is to accompany the final release of the BIMERR Semantic Modelling Component and formalise the conclusion of the development activities carried out in task T4.3 "Building Semantic Modelling Tools Creation". Overall, the BIMERR Semantic Modelling Component is an integral part of the BIMERR Interoperability Framework (BIF) that aims to ensure the semantic mapping and reconciliation of the building-related data that are to be collected from various sources, ranging from the BIMERR applications to legacy systems, while effectively addressing a number of semantic and syntactic interoperability challenges at data and model levels.

In alignment with the BIMERR architecture, the BIMERR Semantic Modelling Component relies on the complementary use of ontologies and data models taking the best of breed of both data modelling paradigms in its Ontology Manager Framework (OMF), its Model Mapper (MM) and its Model Lifecycle Manager (MLM). The three subcomponents that constitute the BIMERR Semantic Modelling Component build on over 15 state-of-the art technologies to deliver the intended functionalities for: (a) the users, i.e., BIMERR applications owners/developers that act as building data providers to the BIF, and (b) the data model and ontology managers that are responsible for managing the creation, evolution and alignment between the BIMERR data models and ontologies.

The present documentation of the BIMERR Semantic Modelling Component (along with its subcomponents) is oriented towards the functionalities it broadly delivers, the technology stacks it builds upon, the APIs it exposes, the installation instructions and usage walkthroughs it offers to its users.

D4.5 is built on the outcomes of D4.4 "BIMERR Building Semantic Modelling tool 1" and provides the updated documentation, as well as the final release of the BIMERR Semantic Modelling Component, fully implementing all the envisaged functionalities along with refinements and updates based on the feedback received from the BIMERR applications, during the BIF integration activities performed in the context of WP4.

1. INTRODUCTION

1.1 Scope and Objectives of the Deliverable

The present deliverable D4.5, entitled "BIMERR Building Semantic Modelling tool 2" constitutes a report of the activities undertaken within the context of Task T4.3 "Building Semantic Modelling Tools Creation" of WP4 "BIMERR Interoperability Framework", towards the delivery of the final version of the Semantic Modelling Tools component of the BIMERR Interoperability Framework (BIF). This document is built upon the outcomes of D4.4. "BIMERR Building Semantic Modelling tool 1" and includes refinements and enhancements based on the outcomes of the design, specification and integration activities performed in WP4 and the feedback received from the partners on the first release.

Overall, the BIMERR Semantic Modelling component is responsible for the definition, application and maintenance of the BIMERR ontology and data model and their proper synchronization, towards ensuring semantic consistency and coherency among the building-related data exchange within BIMERR and with any other external systems. The main objective of D4.5 is to provide a comprehensive overview and the documentation of the final release of the Building Semantic Modelling tool.

Since this deliverable is of type "Demonstrator", it provides the updated documentation of the final release of the actual software that has been developed and delivered in accordance with the BIMERR requirements and architecture. In more detail, D4.5 provides an overview of the functionalities of the Building Semantic Modelling component, along with its architecture. Each of the three subcomponents that constitute the Building Semantic Modelling component, namely the Ontology Manager Framework (OMF), the Model Mapper (MM) and the Model Lifecycle Manager (MLM), is described in detail by:

- Elaborating on the functionalities of each subcomponent.
- Defining the technology stack upon which each subcomponent is based.
- Documenting the Application Programming Interfaces (APIs), i.e., endpoints which will enable the required communications and information exchanges between the different subcomponents and / or within the BIF.

- Explaining any assumptions and restrictions considered in the final release of each subcomponent.
- Providing installation instructions, in order to deploy the subcomponents.
- Offering a usage walkthrough through a set of step-by-step screenshots (whenever available) to explain in detail each subcomponent's intended use.
- Identifying the accompanying licensing of each subcomponent.
- Detailing the new features/changes introduced in this final release.

Finally, considering that the technology and the architecture of the Building Semantic Model component and its associated subcomponents, have not undergone any deviations from their initial release, specific parts of their documentation have the same state, as presented in D4.4.

1.2 Relation to other tasks/deliverables

The BIMERR Deliverable D4.5 documents the activities performed in Task T4.3 "Building Semantic Modelling Tools Creation" and its main scope is to report the final stable version of the Building Semantic Modelling Component. Towards this direction, for the design and implementation of the components described in this deliverable, but also for the creation of this deliverable, the current document receives input from the following deliverables of the BIMERR project:

- D3.1 "Stakeholder requirements for the BIMERR system" [2], where the key BIMERR stakeholders and their requirements are documented, along with a thorough description of the business scenarios, use cases and system requirements tailored to the project's goals, setting the skeleton for the BIMERR framework.
- D3.6 "BIMERR system architecture 2nd version" [4], where the second version of the BIMERR architecture is provided, defining the BIMERR tools and components along with their updated functionalities, as well as the specification for information exchange among them.
- D4.3 "BIMERR Ontology & Data Model 2" [6], where the final BIMERR ontology and data model structure is developed, in order to address the various semantic interoperability challenges for BIM-related data in an efficient manner. The resulting ontology and data model are utilised by the Semantic Modelling component in order

to effectively link information coming from any data source to the BIMERR Interoperability Framework.

- D4.4 “BIMERR Building Semantic Modelling tool 1” [7], describing the initial development activities and the actual delivery of the first stable version of the Building Semantic Modelling tool, upon which the present document (D4.5) is based to deliver the component’s final version.
- D4.6 “BIMERR Information Collection & Enrichment Tool 1” [8], documenting the initial release of the BIMERR Information Collection & Enrichment Tool .
- D4.8 “Integrated BIMERR Interoperability Framework 1” [9], providing a comprehensive documentation of the integrated BIF, and presenting the key technical aspects of the Building Information Secure Provisioning component and Building Information Query Builder and their interaction with the other components of BIF.

D4.5 will be also used as input in the following tasks and work packages:

- T4.4 “Building Information Collection and Enrichment Tools Creation” [6], where the Semantic Modelling tool will be utilised for semantic mapping of the data collected in the Building Information Collection and Enrichment component.
- T4.6 “Building Information Query Builder Creation”, where the Semantic Modelling tool will provide the necessary “data model”-related input for the development of the Building Information Query Builder component ensuring consistency between the interfaces among the respective components.

In addition, D4.5 will provide significant input and a better understanding of the semantic interoperability repercussions to all BIMERR applications that are delivered in WP5 “As-is Building Information Extraction & Model Population Tools”, WP6 “Process Management Tools & End-User Apps for On-site Stakeholders” and WP7 “Renovation Decision Support System”. Finally, D4.5 and the Building Semantic Modelling Component is naturally part of the system level software integration and pre-validation activities to be performed in WP8 and thereafter in the validation and evaluation activities of WP9.

1.3 STRUCTURE OF THE DOCUMENT

In order to address all the aspects relevant to the scope of T4.3, the present deliverable has been structured as follows:

- Section 1 introduces the work performed and the scope of this deliverable along with its relevance to other BIMERR tasks and the deliverable's structure.
- Section 2 provides an overview of the BIMERR Building Semantic Modelling Tool 2, and its architecture.
- Sections 3–5 provide a comprehensive documentation of the different subcomponents forming the Building Semantic Modelling component, i.e., the Ontology Manager Framework, the Model Mapper and the Model Lifecycle Manager, respectively and the changes introduced in their final release.
- In Section 6, the conclusions are provided.

2. BIMERR BUILDING SEMANTIC MODELLING COMPONENT

2.1 OVERVIEW

The BIMERR Building Semantic Modelling component constitutes one of the four core components of the BIMERR Interoperability Framework (BIF) and it is instrumental for addressing the semantic interoperability challenges pertaining to the building-relevant data that derive from and are exchanged between the BIMERR applications and external systems. Such a component is responsible for understanding the semantics of the data that are to be handled by the BIF, linking or mapping them with the BIMERR data models and ontologies and effectively handling their lifecycle and alignment.

The Building Semantic Modelling component is composed of three main subcomponents (as depicted in Figure 2-1) which are briefly described below and individually presented in the Sections 3, 4 and 5.

- The ***Model Mapper*** (MM) aims to ensure semantic consistency between the building data that are extracted from legacy systems or from the BIMERR applications, and the BIMERR data model. Such a consistency is ensured through a semi-automated mapping prediction and configuration process that takes place with the help of the user (e.g., BIMERR application developer). In the Model Mapper, the users can review the automated mapping predictions (that are accompanied by a confidence level scale), and set the applicable mapping and transformation rules, as well as propose new concepts to be added to the BIMERR data model that satisfy the scope of their data.
- The ***Model Lifecycle Manager*** (MLM) is responsible for providing a thorough overview of the BIMERR data models and managing their evolution, from their initial creation and storage to their eventual evolution based on a set of predefined evolution rules and the moderation of the data model managers. Any evolution event is communicated to the Ontology Manager Framework to ensure that the BIMERR data models and ontologies are aligned.
- The ***Ontology Manager Framework*** (OMF) aims to support the various ontology management activities, including the documentation, evaluation, versioning and publishing of the BIMERR ontology while ensuring that both the BIMERR data model

and ontology are updated according to the latest state-of-the-art standards and enriched with further concepts required from the various applications.

The final version of the BIMERR Semantic Modelling component is deployed at: https://bimerr.s5labs.eu/

2.2 Architecture

In alignment with the BIMERR deliverable D3.6 [4], Figure 2-1 illustrates the BIMERR Semantic Modelling component's architecture including its main subcomponents, information flows and interactions among them. As already mentioned, the Semantic Modelling component is composed of three core subcomponents, namely the Model Mapper, the Model Lifecycle Manager and the Ontology Manager Framework; each one designed with a distinct role, a distinct scope and a distinct set of core functionalities.

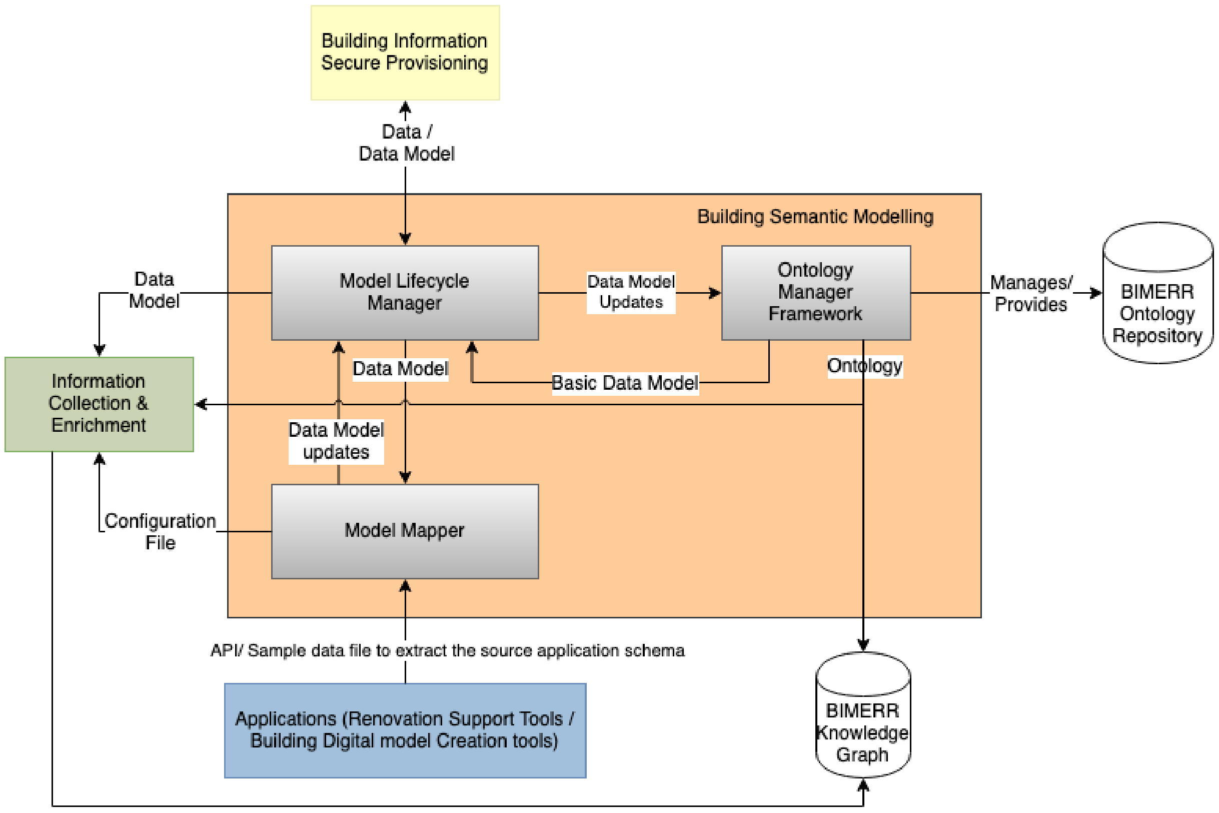

Building Information Secure Provisioning
Data / Data Model
Building Semantic Modelling
Data Model
Model Lifecycle Manager
Data Model Updates
Ontology Manager Framework
Manages/ Provides
BIMERR Ontology Repository
Information Collection & Enrichment
Data Model
Basic Data Model
Ontology
Data Model updates
Configuration File
Model Mapper
API/ Sample data file to extract the source application schema
Applications (Renovation Support Tools / Building Digital model Creation tools)
BIMERR Knowledge Graph

Figure 2- 1: Architecture of the BIMERR Semantic Modelling component

Once the user has initiated a Data Collection job in the BIF Information Collection & Enrichment component, the Model Mapper is involved and undertakes the semi-automatic mapping of the source data that are collected and shall be exchanged through the BIF, to the underlying BIMERR data model. The Model Mapper derives the underlying schema from the available sample, predicts the mappings of the external data concepts to the BIMERR concepts, while enabling users to complement the mapping arrangements and add custom transformation rules. Upon a successfully completed mapping configuration, the Model Mapper populates the "Configuration File" with the data model mapping information, which are made available to the Building Information Collection component.

The Model Lifecycle Manager's role is to import the basic data models that have been received from the Ontology Manager Framework, to enrich the data model information (in accordance with the data model information described in the BIMERR Deliverable D4.3 [5]), to expose the BIMERR data models to any BIF component that requires them, as well as to oversee the evolution of the BIMERR data model. The Model Lifecycle Manager also receives requests for data model updates from the Model Mapper and effectively manages them with the help of the BIMERR data model administrator. In addition, this component communicates with the OMF to ensure that the stored data models and ontologies are aligned; whenever a new concept is introduced, both the respective BIMERR data model and the ontology are updated from these two components, respectively.

The Ontology Manager Framework offers a documentation and publication tool that extracts the ontology metadata and generates the documentation from the relevant ontology metadata properties which is made available to the BIMERR Knowledge Graph and the BIMERR Ontology Repository. In addition, the Ontology Manager Framework provides a JSON serialization of the ontology to the Model Lifecycle Manager and, in turn, receives the updates from the BIMERR data models that need to be incorporated into the BIMERR ontologies network.

3. ONTOLOGY MANAGER FRAMEWORK

3.1 OVERVIEW

The Ontology Manager Framework is a collaborative environment suite to support several ontology management activities, including the documentation, evaluation, versioning and publishing of the BIMERR ontology network.

The Ontology Manager Framework is composed of by three main modules: 1) OnToology, a web application to support the ontology development process, and 2) Chowlk, a web application and service to transform ontology conceptualizations into OWL code, and 3) BO2DM, a web service that performs the transformation process from an ontology to a data model. Each module is discussed in detail in the following subsections. It should be mentioned that the Chowlk module was not detailed in D4.4.

3.1.1 OnToology

OnToology is a web framework that supports many of the ontological activities carried out during the ontology development process. This tool tracks ontologies stored in GitHub repositories, producing a series of actions any time a new change is made on their implementation. These actions involve: 1) the generation of HTML documentation, 2) generation of an evaluation report that include pitfalls found on the implementation, 3) the publications of the ontologies and its related resources. It should be mentioned that, as the ontologies are published in UPM own servers, the OnToology feature for publishing has been replaced for the development of a system that automates the publication when a new release of an ontology is defined in GitHub.

3.1.2 Chowlk

Chowlk is a web application and service that generates OWL ontology code from a ontology conceptualization serialized in XML. The conceptualization should be done following the Chowlk notation and using the system diagrams.net[1] to generate the XML diagram.

3.1.3 BO2DM

The BIMERR Ontology to Data Model converter is a web service, which given an ontology implemented in OWL performs a series of transformations to finally return a JSON serialized data model. This converter should receive an enriched version of the original ontology in order to extract all the metadata required by the BIF. Some of the fields required by BIF can be directly extracted from the ontology, however there are other fields that require the annotation of the ontology elements with additional metadata. A detailed list of the fields required by BIF is shown in Table 3-1 in alignment with the BIMERR deliverable D4.2 [5]. This table also includes a short description of each term and if they need to be populated during the implementation of the data model in the BIF.

Table 3-1: Metadata List

Metadata on Data Model	Equivalent Annotation Property or Ontology Resource used	Extracted From	Description	Populated By
Definition	rdfs: comment	Original Ontology	A brief overview that acts as an account of a concept's contents	Automatically extracted from the ontology
Type	Extracted from restrictions on: owl:ObjectProperty owl:DatatypeProperty	Original Ontology	The data type to which the concept's data comply, e.g. string, integer, boolean, datetime, etc	Automatically extracted from the ontology and applied to children nodes.
Related_terms	rdfs: labelskos:altLabel	Original Ontology	A set of related terms (e.g. synonyms) that can be alternatively used to represent the concept	Manually annotated by the data model admin in the Model Lifecycle Manager

[1] https://www.diagrams.net/

Metadata on Data Model	Equivalent Annotation Property or Ontology Resource used	Extracted From	Description	Populated By
standards	bm:isDefinedByStandard	Enriched Ontology	A list of standards in which the specific concept is modeled, along with their complementary information, i.e. the exact concept used in such a standard, its type (e.g. element, attribute) and its use (required/optional) that applies for attributes	Partly extracted from the ontology complemented by the data model admin in the Model Lifecycle Manager
Date_added	dc: created	Enriched Ontology	The date when the specific concept was added in the data model	Used for provenance and alignment between the ontology and the respective data model
Date_deprecated	bm: deprecated	Enriched Ontology	The date when the specific concept became obsolete in the data model	Used for provenance and alignment between the ontology and the respective data model
Version	owl: versionInfo	Enriched Ontology	The version of the model when the concept was last modified	Used for provenance and alignment between the ontology and the respective data model
Children	Properties or attributes of type: owl: ObjectProperty owl: DatatypeProperty	Original Ontology	Grouped information for concepts that are conceptually classified under a concept	Automatically extracted from the ontology

Metadata on Data Model	Equivalent Annotation Property or Ontology Resource used	Extracted From	Description	Populated By
Facet	---	---	Complementary information for a concept, essentially grouping the following metadata: cardinalityMax, ordered, sensitive, transformation, measurementType, measurementUnit, timezone	Partly extracted from the ontology and complemented by the data model admin in the Model Lifecycle Manager
cardinalityMax	Max cardinality extracted from: owl: FunctionalProperty owl:maxQualifiedCardinality	Original Ontology	The maximum number of expected appearances of a concept in the data	Automatically extracted from the ontology
Ordered	bm: ordered	Enriched Ontology	An indication of whether ordering is needed for multiple appearances of the same concept.	Automatically extracted from the ontology
Sensitive	bm: sensitive	Enriched Ontology	An indication whether the specific concept models personal or sensitive data	Automatically extracted from the ontology
transformation	bm: transformation	Enriched Ontology	Information for the transformation rules that are related to a specific applicable standard. It practically contains the function that is required for the transformation and the parameters / concepts that are involved.	Manually annotated by the data model admin in the Model Lifecycle Manager
measurementType	bm: measurementType	Enriched Ontology	An indication of the measurement type that is applicable to a concept, e.g. referring to distance,	Manually annotated by the data model admin in the Model

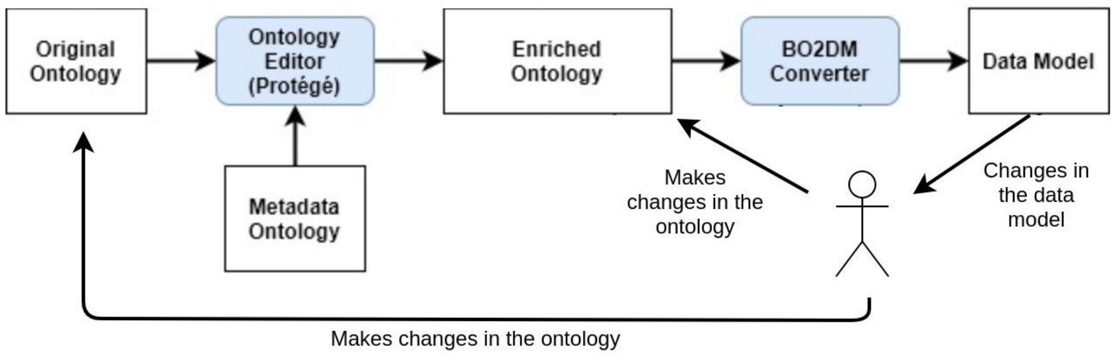
Metadata on Data Model	Equivalent Annotation Property or Ontology Resource used	Extracted From	Description	Populated By
			temperature, etc.	Lifecycle Manager
measurementUnit	bm: measurementUnit	Enriched Ontology	The baseline measurement unit for the specific concept and measurement type	Manually annotated by the data model admin in the Model Lifecycle Manager
timeZone	bm: timeZone	Enriched Ontology	The timezone to which the data refer by default.	Manually annotated by the data model admin in the Model Lifecycle Manager

Table 3-2 shows complementary information regarding the namespaces and the corresponding prefixes.

Table 3-2: Ontology Namespaces and Prefixes

Prefix	Ontology namespace
owl	http://www.w3.org/2002/07/owl#
dc	http://purl.org/dc/elements/1.1/
bm	https://bimerr.iot.linkeddata.es/def/bimerr-metadata#
skos	http://www.w3.org/2004/02/skos/core#
rdfs	http://www.w3.org/2000/01/rdf-schema#

The complete transformation pipeline is depicted on Figure 3-1. The process starts with the enrichment step using an ontology editor, where the ontology engineer instantiates, the extra fields required to generate the enriched version of the ontology. The original ontology is not edited in this process, but the changes are stored in a separated file. The BO2DM service takes as input this enriched ontology and triggers a series of conversion steps to finally generate the data model. Any changes produced to the original ontology will propagate to the enriched one that at the same time will materialize those changes in the data model through the BO2DM converter. Finally, the data model is produced to be inserted and published in the BIF. In case some modifications are needed from the BIF user side in the model, the changes are communicated to the ontology development team members, who update the ontology or the enriched version depending on the type of change.



Original Ontology
Ontology Editor (Protégé)
Enriched Ontology
BO2DM Converter
Data Model
Metadata Ontology
Makes changes in the ontology
Changes in the data model
Makes changes in the ontology

Figure 3-1: Ontology to Data Model Conversion Pipeline

3.2 Technology Stack and Implementation Tools

The OnToology framework takes advantage of existing technologies in order to perform the versioning and hosting activities. Specifically, GitHub is used as the main storing infrastructure, hosting the ontology implementation and all its complementary resources. Additionally, GitHub provides the means to handle multiples versions of the ontology, generated during the implementation sprints. Finally, it provides the opportunities to manage issues and improvement requirements from users or ontology developers through the GitHub issues section.

OnToology is composed of four layers, 1) the presentation layer, which is a Graphical User Interface (GUI) web interface that allows the user the registering of their repositories, 2) the logic layer, which monitors changes in the ontology repository, and selects and activates the appropriate external and internal services that make up the framework suite, and 3) the persistence layer, implemented as an internal database containing a list of the users information, their tracked repositories, and their state in the processing pipeline.

The Ontology Manager Framework has been developed using Python 3.6[2] as the main programming language. Table 3-3 indicates all the libraries and software packages used in the OnToology framework.

Table 3-3: List of Libraries used in OnToology

Name of the Library / Software	Version	License
Django	1.11.28	BSD 3-Clause
Pymongo	2.7.2	Apache 2.0
RDFLib	4.2.2	BSD 3-Clause
Requests	2.22.0	Apache 2.0
PyGithub	1.35	MIT
Selenium	2.52.0	Apache 2.0
Mongoengine	0.14.3	MIT
Widoco	1.4.13	Apache 2.0
OOPS!	---	GPL-3

On the other hand, the Chowlk converter has been developed as a single web page application where the user can upload its ontology conceptualization and the converter returns the corresponding OWL ontology. The service is composed of two layers: 1) The presentation layer where the user can upload the diagram of the ontology in an XML format, and the 2) Logic layer that performs the transformation required to implement the ontology in Web Ontology Language (OWL).

Table 3-4 shows all the libraries and software packages used in the Chowlk component.

Table 3-4: List of Libraries used in Chowlk

Name of the Library / Software	Version	License
Flask	1.1.2	BSD 3-Clause
Flask Bootstrap	3.25.0	MIT
RDFLib	4.2.2	BSD 3-Clause

[2] https://www.python.org/

The BO2DM converter is implemented as a python script, which performs a series of transformations to generate the JSON data model required by BIF, based on an ontology in Turtle format.

Table 3-5: List of Libraries used in BO2DM

Name of the Library / Software	Version	License
Flask	1.1.2	BSD 3-Clause
Flask Swagger UI	3.25.0	MIT
RDFLib	4.2.2	BSD 3-Clause
RDFLib JSON-LD	0.4.0	BSD 3-Clause

3.3 API Documentation

OnToology provides all the appropriate documentation on its web site,[3] including detailed instructions of usage, and explanations of the internal functionality.

Chowlk documentation is provided on its web site[4]. The service is provided both as a web application, accessible from Chowlk website, and a web service[5].

Finally, the converter's documentation is provided in Swagger.[6] This web interface gives examples of the type of requests available and the parameters needed.

3.4 Changes Introduced in Final Release

The Ontology Manager Framework has been improved in two main directions: First, the Chowlk system has been developed to ease the ontology implementation activity taking as input the diagrams generated during the ontology conceptualization. In addition, the publication of the ontologies has been automated by publishing a new version every time there is a new release in the corresponding GitHub repository. This feature is not

[3] http://ontoology.linkeddata.es/

[4] https://chowlk.linkeddata.es/

[5] Web service available at https://chowlk.linkeddata.es/api

[6] https://converter.bimerr.iot.linkeddata.es/swagger/

accessible outside the BIMERR project as it affects a publication server that should allow the application to modify and add files.

3.5 Assumptions and Restrictions

Each of the aforementioned OMF modules has its own set of limitations. These limitations are being handled as restrictions over the input data or operational modes in which the tools can be executed; or assumptions the tools made in order to simplify their functionality.

3.5.1 OnToology

One of the main limitations of OnToology is the missing support of repositories under a GitHub organization and private repositories. If the ontological (OWL) implementation is under an organization repository, it needs to be forked and then fed into OnToology with the new URL (your-username/the-repo-name).

Additionally, it only works with the master branch, which means that modifications result of development activities carried out in another branch need to be merged into the master branch first, before OnToology can track the changes.

The size of ontologies is another aspect to take into consideration when it comes to the evaluation activity. The evaluation of very large ontological models may not work due to OOPS! Web service timing out. However, the HTML documentation and the diagrams generation will not be affected.

The publication in a given organization server involves a manual process to deploy the files from the GitHub repository to the organization web server which includes authentication. The publication under w3id permanent identifiers can’t be set for paths, only the identifier of the ontology can be chosen.

The ontology conceptualization and enrichment are manual steps. The updates of the enriched ontologies are semi-automatic, depending on the changes reported by the data model.

3.5.2 Chowlk

Chowlk assumes the diagram to be converted into an ontology is valid and follows the Chowlk notation[7].

An additional limitation is the use of stereotype lines to connect arrows, which are not detected by Chowlk. This might be used to indicate subproperties, equivalent properties or inverse properties. In this case, the alternative notation using diamonds should be used in addition[8].

3.5.3 BO2DM

The lack of clear transformation rules to convert the BIMERR ontology into a JSON serialized data model makes it necessary to introduce some assumptions to generate the data structure expected by BIF.

First, the datatype properties attached to a specific concept in the ontology are children fields of this concept on the JSON data model.

Second, the object properties between classes in the ontology are also incorporated as children fields of concepts into the JSON data model. However, the target datatype of those fields are references to other concepts in the data structure.

3.6 INSTALLATION INSTRUCTIONS

Each module of the OMF platform is deployed as a web service, meaning that they do not require the installation or downloading of any component in order to use them. OnToology requires an initial setup to register the ontological repositories, however, besides that no other step is required.

[7] https://chowlk.linkeddata.es/chowlk_spec

[8] See https://chowlk.linkeddata.es/chowlk_spec#relations-between-properties for details about the notation of the relations between properties.

3.7 USAGE WALKTHROUGH

The initial step consists of registering the ontological repository to be tracked by OnToology through the web user interface, as shown in Figure 3-2. In case it is the first time the user accesses the service, OnToology will redirect the user to GitHub to confirm tracking permissions. Once the authorization is confirmed, OnToology will start tracking any changes produced on the ontology.

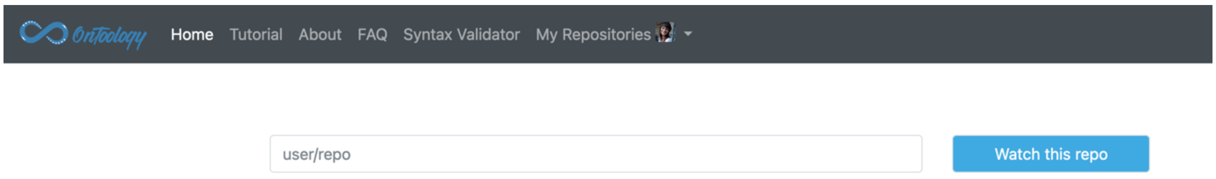


Figure 3-2: Ontology registering

Every time ontology updates are pushed to the remote tracked repository, OnToology will automatically generate or re-regenerate the resources and will create a pull request, as shown in Figure 3-3, with those resources, i.e., the HTML documentation and the validation report.

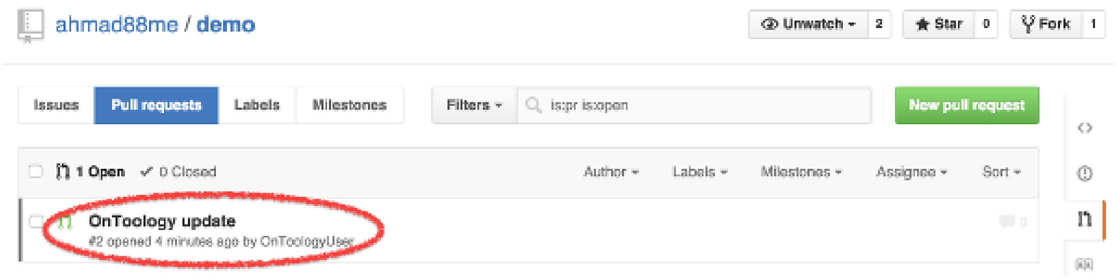
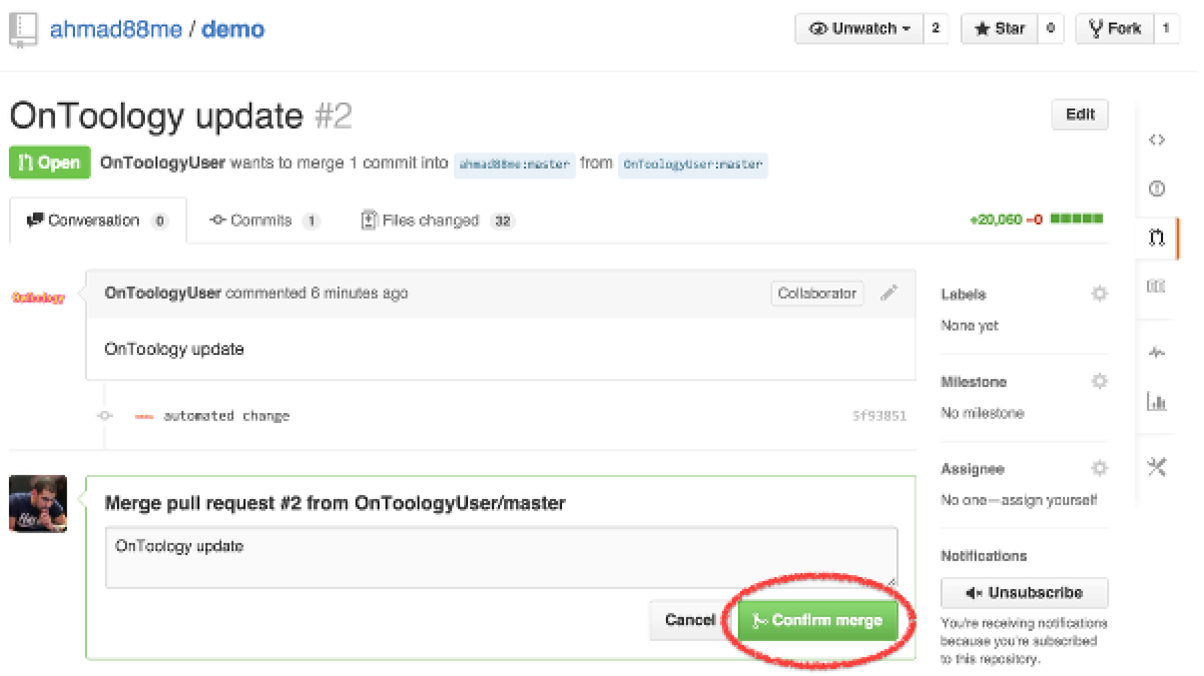


Figure 3-3: Pull request generation

Once the pull request appears on the ontology repository, the ontology owner has to review the changes and merge the appropriate pull request to accept the generated resources, as depicted in Figure 3-4. By confirming the merge, a new folder called OnToology with all the resources will be located inside the ontology repository, as shown in Figure 3-5.



OnToology update #2

Open OnToologyUser wants to merge 1 commit into ahmad88me:master from OnToologyUser:master

OnToologyUser commented 6 minutes ago

OnToology update

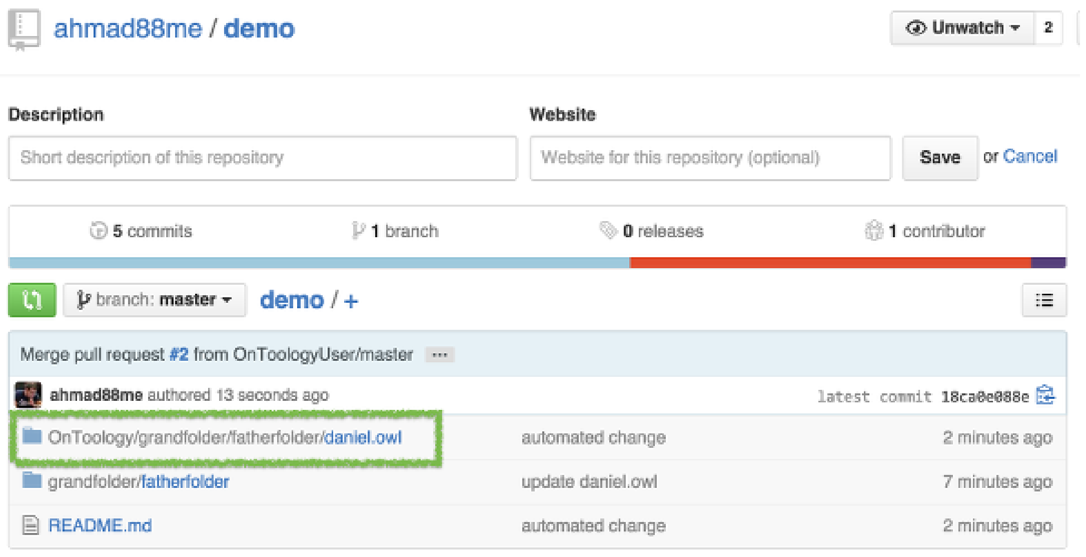
automated change 5f93851

Merge pull request #2 from OnToologyUser/master

OnToology update

You're receiving notifications because you're subscribed to this repository.

Figure 3-4: Pull request merge confirmation



Short description of this repository

Website for this repository (optional)

5 commits | 1 branch | 0 releases | 1 contributor

Merge pull request #2 from OnToologyUser/master

ahmad88me authored 13 seconds ago — latest commit 18ca0e088e

OnToology/grandfolder/fatherfolder/daniel.owl	automated change	2 minutes ago
grandfolder/fatherfolder	update daniel.owl	7 minutes ago
README.md	automated change	2 minutes ago

Figure 3-5 Creation of OnToology resource folder

Even though OnToology provides users with ready-to-publish core documentation extracted from the analyzed ontologies, most of the times users need to further customize parts of this initial HTML documentation to include examples or additional description about the functionality of the ontology. Once all the HTML sections have been customized, users may want to publish the ontology online. Whether it is the first time the users publish their ontology or it is just updating information, OnToology can handle the publication process by the permanent URI mentioned in Section 3.2, or the users can download all the HTML documentation to publish the ontological model in their own server. In the case of BIMERR, the latter option is selected, where the process of publishing is automated by means of GitHub releases and webhooks. More precisely, the Ontology Manager Framework constantly monitors event-driven messages sent by GitHub every time a new release is published in the ontology repository. When a new message is detected, the OMF pulls the latest version of the HTML documentation updating the content of the BIMERR sites automatically.

The BO2DM tool is also tracking the repository that stores the enriched version of the ontological models. The converter generates the data model on demand. When a request comes from any of the semantic modelling tools or a specific user, the module retrieves the respective ontology and generates its JSON-serialized counterpart.

The Chowlk web application would be used during the ontology implementation activity. The result of Chowlk would be stored in the ontology GitHub repository to be later be tracked by OnToology. For using Chowlk a user needs to create a conceptualization diagram in diagrams.net following the Chowlk notation, as shown in Figure 3-6.

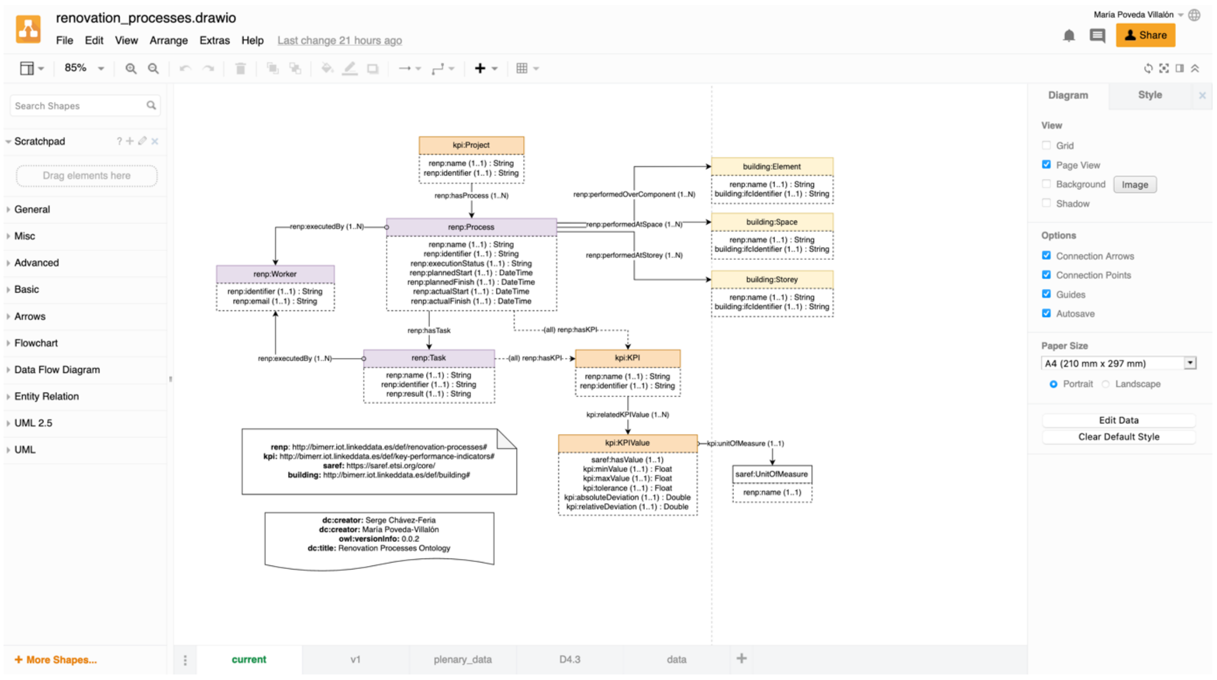
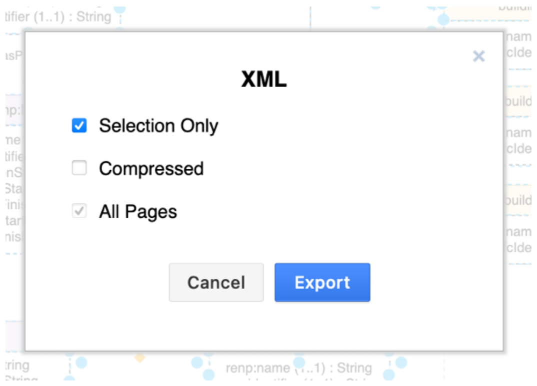

renovation_processes.drawio
File Edit View Arrange Extras Help Last change 21 hours ago
María Poveda Villalón
Share
85%
Search Shapes
Scratchpad
Drag elements here
General
Misc
Advanced
Basic
Arrows
Flowchart
Data Flow Diagram
Entity Relation
UML 2.5
UML
+ More Shapes...
kpi:Project
renp:name (1..1) : String
renp:identifier (1..1) : String
renp:hasProcess (1..N)
renp:Process
renp:name (1..1) : String
renp:identifier (1..1) : String
renp:executionStatus (1..1) : String
renp:plannedStart (1..1) : DateTime
renp:plannedFinish (1..1) : DateTime
renp:actualStart (1..1) : DateTime
renp:actualFinish (1..1) : DateTime
renp:executedBy (1..N)
renp:Worker
renp:identifier (1..1) : String
renp:email (1..1) : String
renp:performedOverComponent (1..N)
renp:performedAtSpace (1..N)
renp:performedAtStorey (1..N)
building:Element
renp:name (1..1) : String
building:ifcIdentifier (1..1) : String
building:Space
renp:name (1..1) : String
building:ifcIdentifier (1..1) : String
building:Storey
renp:name (1..1) : String
building:ifcIdentifier (1..1) : String
renp:hasTask
(all) renp:hasKPI
renp:executedBy (1..N)
renp:Task
renp:name (1..1) : String
renp:identifier (1..1) : String
renp:result (1..1) : String
(all) renp:hasKPI
kpi:KPI
renp:name (1..1) : String
renp:identifier (1..1) : String
kpi:relatedKPIValue (1..N)
kpi:KPIValue
saref:hasValue (1..1)
kpi:minValue (1..1) : Float
kpi:maxValue (1..1): Float
kpi:tolerance (1..1) : Float
kpi:absoluteDeviation (1..1) : Double
kpi:relativeDeviation (1..1) : Double
kpi:unitOfMeasure (1..1)
saref:UnitOfMeasure
renp:name (1..1)
renp: http://bimerr.iot.linkeddata.es/def/renovation-processes#
kpi: http://bimerr.iot.linkeddata.es/def/key-performance-indicators#
saref: https://saref.etsi.org/core/
building: http://bimerr.iot.linkeddata.es/def/building#
dc:creator: Serge Chávez-Feria
dc:creator: María Poveda-Villalón
owl:versionInfo: 0.0.2
dc:title: Renovation Processes Ontology
current
v1
plenary_data
D4.3
data
Diagram
Style
View
Grid
Page View
Background
Image
Shadow
Options
Connection Arrows
Connection Points
Guides
Autosave
Paper Size
A4 (210 mm x 297 mm)
Portrait
Landscape
Edit Data
Clear Default Style

Figure 3-6: Diagram creation in diagrams.net following Chowlk notation

The user should then export the diagram in XML as shown in Figure 3-7. Finally, the user uploads the XML file in the Chowlk web user interface (Figure 3-8) and obtains the OWL code both in ttl and rdf/xml formats (Figure 3-9).



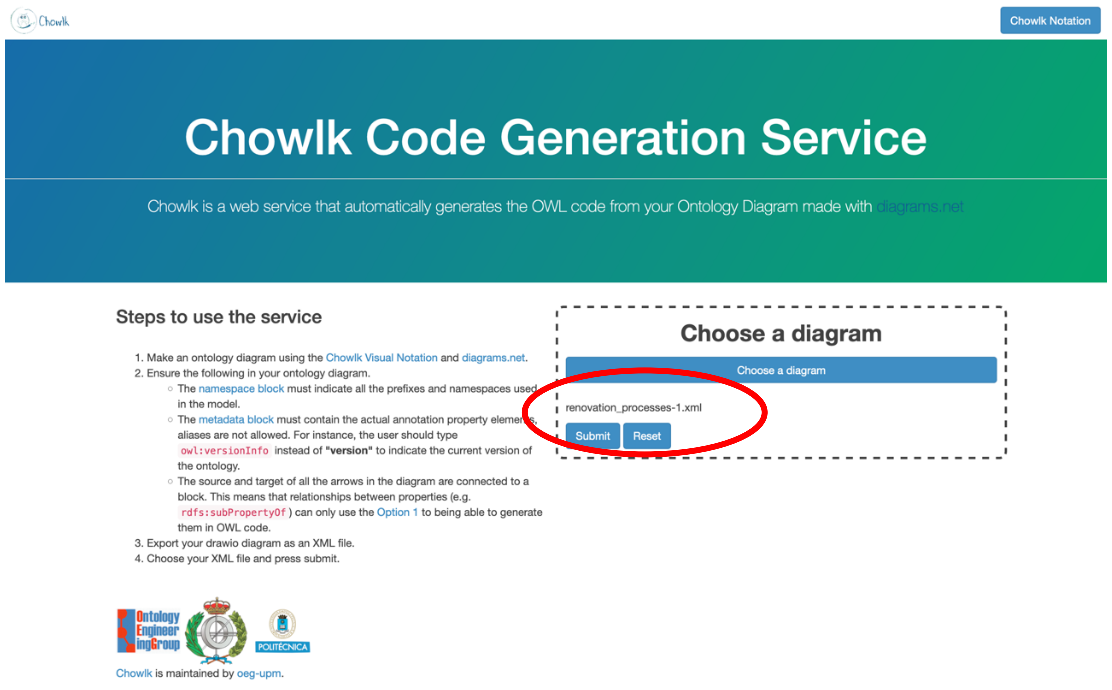
XML

☑ Selection Only

☐ Compressed

☑ All Pages

Figure 3-7: Export diagram in XML


Chowlk Notation

Chowlk Code Generation Service

Chowlk is a web service that automatically generates the OWL code from your Ontology Diagram made with diagrams.net

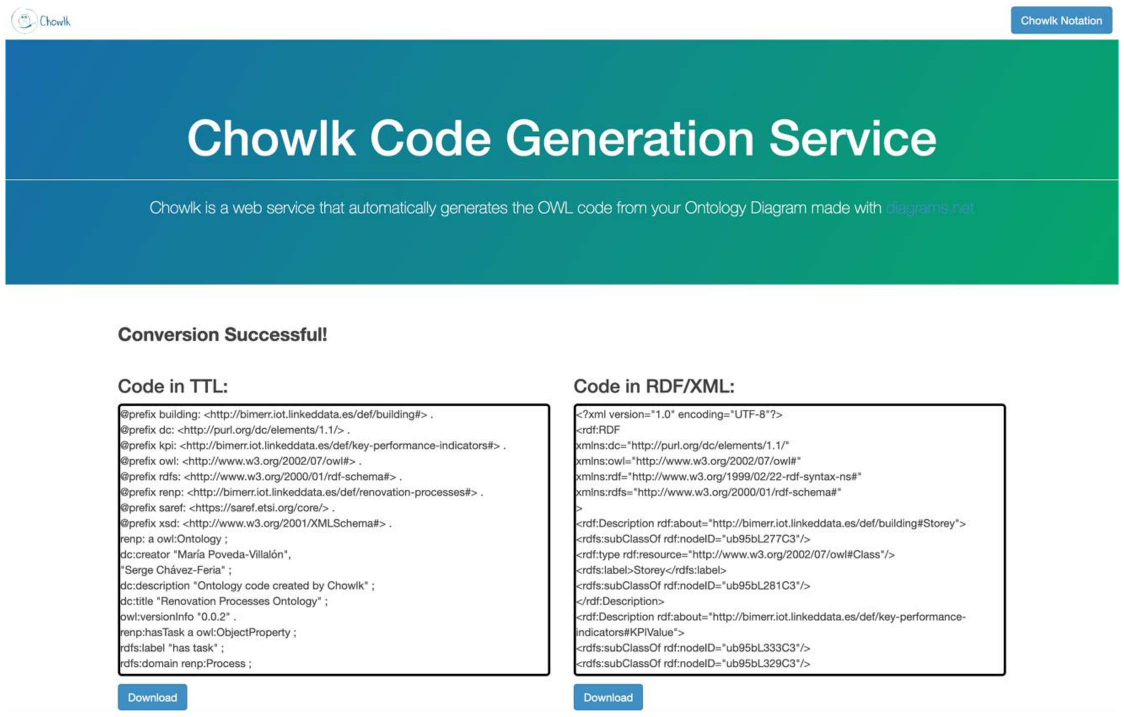
Steps to use the service

1. Make an ontology diagram using the Chowlk Visual Notation and diagrams.net.
2. Ensure the following in your ontology diagram.
   - The namespace block must indicate all the prefixes and namespaces used in the model.
   - The metadata block must contain the actual annotation property elements, aliases are not allowed. For instance, the user should type `owl:versionInfo` instead of **"version"** to indicate the current version of the ontology.
   - The source and target of all the arrows in the diagram are connected to a block. This means that relationships between properties (e.g. `rdfs:subPropertyOf`) can only use the Option 1 to being able to generate them in OWL code.
3. Export your drawio diagram as an XML file.
4. Choose your XML file and press submit.

Chowlk is maintained by oeg-upm.

Choose a diagram

Figure 3-8: Upload diagram in XML


Chowlk Notation

Chowlk Code Generation Service

Chowlk is a web service that automatically generates the OWL code from your Ontology Diagram made with diagrams.net

Conversion Successful!

Code in TTL:

```
@prefix building: <http://bimerr.iot.linkeddata.es/def/building#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix kpi: <http://bimerr.iot.linkeddata.es/def/key-performance-indicators#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix renp: <http://bimerr.iot.linkeddata.es/def/renovation-processes#> .
@prefix saref: <https://saref.etsi.org/core/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
renp: a owl:Ontology ;
dc:creator "María Poveda-Villalón",
"Serge Chávez-Feria" ;
dc:description "Ontology code created by Chowlk" ;
dc:title "Renovation Processes Ontology" ;
owl:versionInfo "0.0.2" .
renp:hasTask a owl:ObjectProperty ;
rdfs:label "has task" ;
rdfs:domain renp:Process ;
```

Code in RDF/XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
<rdf:Description rdf:about="http://bimerr.iot.linkeddata.es/def/building#Storey">
<rdfs:subClassOf rdf:nodeID="ub95bL277C3"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
<rdfs:label>Storey</rdfs:label>
<rdfs:subClassOf rdf:nodeID="ub95bL281C3"/>
</rdf:Description>
<rdf:Description rdf:about="http://bimerr.iot.linkeddata.es/def/key-performance-
indicators#KPIValue">
<rdfs:subClassOf rdf:nodeID="ub95bL333C3"/>
<rdfs:subClassOf rdf:nodeID="ub95bL329C3"/>
```

Figure 3-9: Download ontology code

3.8 LICENSING

OnToology, Chowlk and BO2DM are licensed under the Apache 2.0[9] license.

[9] https://www.apache.org/licenses/LICENSE-2.0

4. MODEL MAPPER

4.1 OVERVIEW

The functionalities of the Model Mapper subcomponent are of great significance for the interoperability between the BIMERR applications, as data previously modeled following different standards and models will be harmonized under a common data model, which will facilitate their sharing among the BIMERR applications. The appropriately configured data model mapping processes ensure the smooth and well-structured "data collection" of application data (i.e., uploading an application's data to the BIF) and sensor data through the Middleware, as well as the compatibility of the interoperating applications.

If the user has declared that the data to be retrieved by a data collection job shall undergo the mapping stage, instead of being handled as an object, the Model Mapper becomes involved in the "data collection" preparation and configuration step. Its core functionalities entail the mapping prediction and then the semi-automatic mapping configuration through the population of a "Configuration file" with data model mapping information, following a semi-automated process. The mapping information, which is typically included in the "Configuration file", defines how the underlying data model of the data to be exchanged through the BIF should be mapped to a selected BIMERR data model.

More specifically, the functionalities provided by the Model Mapper are the following:

- **Automatic calculation of mapping predictions based on a set of matching techniques**: The Model Mapper shall combine the information provided by the users and shall extract the underlying data model from the data sample that has been uploaded. With the help of various techniques ranging from fuzzy matching (i.e. matching the user input field to model fields based on the names and the related terms of the leaf nodes using Levenshtein distance) and elastic matching (i.e. matching the user input field to model fields based on various text fields of the model's leaf nodes using advanced queries on the indexed data models) to sample-based matching (applying different algorithms for learning from the sample contents) and standards-based matching (that considers the predefined mapping of the data model to existing data models for the concept at hand), the Model Mapper provides possible mappings from the external data model concepts to the BIMERR concepts. These predictions are accompanied by calibrated confidence scores.

- **Manual mapping configuration confirmation and updates by the users**: The Model Mapper shall provide an intuitive user interface to the users, where they shall navigate through the mapping predictions, providing the necessary mapping and transformation details or updating them to the correct concepts. Through the Model Mapper, the users shall identify the mappings or select alternative, related concepts for any unidentified concepts (for which the mapping prediction confidence score was not above a certain threshold). In this context, the Model Mapper shall also offer the option to define data types, measurement units and define any other required transformations (e.g., for datetime fields). The final mapping configurations are stored in the "Configuration File".
- **Easy navigation to the BIMERR data models**: The Model Mapper shall facilitate the users in understanding the different concepts that appear in the BIMERR data models in order to make prediction reconciliations and manual mappings in a more effortless manner.
- **User-driven proposition of new concepts**: In case the concepts in the underlying BIMERR data models do not effectively address the needs of a BIMERR application, the Model Mapper shall allow for suggestions for new concepts in an easy manner. Once created, the suggestions are forwarded to the Model Lifecycle Manager subcomponent, in order to be semi-automatically handled and further processed by the model administrator, whenever needed.

4.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Model Mapper builds on state-of-the art technologies across 3 layers: the Presentation Layer, containing the Model Mapper User Interface that is developed in VueJS[10] and TailwindCSS[11]; the Business Logic Layer, containing the different packages of the Model Mapper Backend that are based in the Flask micro web framework[12]; and the Data Access Layer that essentially refers to the BIF Storage and Indexing that has been

[10] https://vuejs.org/

[11] https://tailwindcss.com/

[12] https://flask.palletsprojects.com/en/1.1.x/

set up in the context of the Building Information Collection and Enrichment component and utilizes ElasticSearch[13] and PostgreSQL[14], for the Model Mapper needs.

Such layers along with the different technologies are depicted in Figure 4- 1.

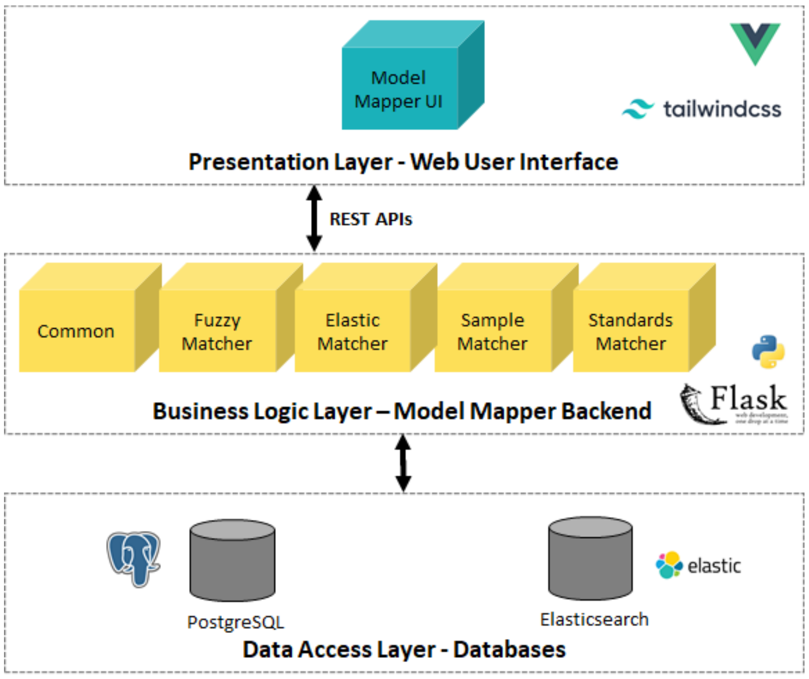


Figure 4- 1: Architecture of the Model Mapper under the BIMERR Semantic Modelling component

The Model Mapper is written in Python 3.8.2[15] and utilizes the following open-source technologies as defined in Table 4-1 below.

Table 4-1: Technologies and libraries used in the Model Mapper, along their licenses

Name of the Library	Version	License
Flask	1.1.1	BSD 3-Clause
Flask RESTful extension	0.3.8	BSD 3-Clause
Flask CORS support	3.0.8	MIT

[13] https://www.elastic.co/

[14] https://www.postgresql.org/

[15] https://www.python.org/

Name of the Library	Version	License
PostgreSQL	12.2	PostgreSQL License (similar to BSD/MIT)
Elasticsearch	7.6.0	Elastic License
Elasticsearch DSL	7.6.0	Apache License 2.0
Vue.js	2.6.11	MIT
TailwindCSS	-	MIT
NumPy	1.18.1	BSD
Pandas	1.0.3	BSD 3-Clause
scikit-learn	0.22.1	BSD 3-Clause

4.3 API Documentation

The APIs that accompany the Model Mapper have been documented in Swagger, but they are intended only for internal use by the BIF, thus they are not presented in this section.

4.4 Changes Introduced in Final Release

In respect to the first release of the Model Mapper component, a number of improvements and enhancements has been introduced in this final release. In particular, the following features and extensions have been developed upon discussions with the partners and the feedback received on the first release:

- Improvements of the matching techniques upon experimentation with sample data collected in BIF from the BIMERR applications during the integration activities performed in WP4.
- Management of “multiple” predictions per concept (from the same data model and additional data models)
- Better notifications to the user for the inclusion of new concepts to the BIMERR data models of interest for a specific data collection job.
- Integration with the Model Lifecycle Manager to properly handle (from the side of the affected data collection jobs) any changes that are introduced in the respective data models.
- Better user experience in the drag’n’drop functionalities.
- Bug fixing and performance improvements.

4.5 Assumptions and Restrictions

In the final release of the Model Mapper, a number of assumptions (that in certain cases, represent restrictions for the Model Mapper) were taken:

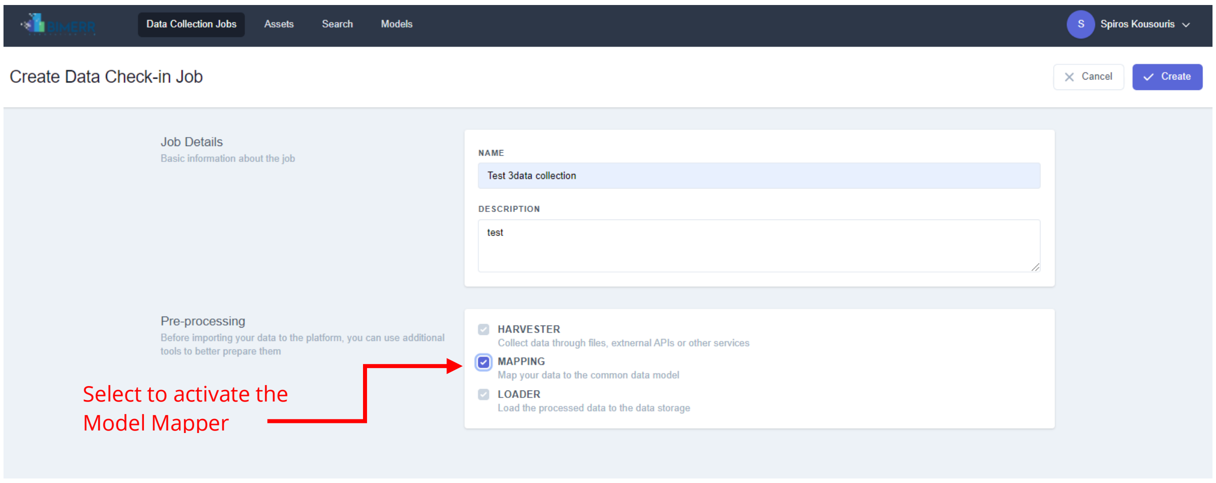
- The Model Mapper will receive the data sample and data structure in a JSON format that may be flattened or non-flattened (in a nested structure), depending on the building data collection parameters.
- The Model Mapper will allow users to create connections between the different models.
- The Model Mapper will only return "single" results per concept that appears in the source data, without offering any alternatives in the BIMERR data model to which a source concept could be also potentially mapped. Such an assumption was taken in order to offer a better user experience without overloading or confusing the users.
- The Model Mapper may predict mapping of the same concept of the applicable BIMERR data model to multiple source concepts. Such a decision was taken not only since there are concepts that may appear multiple times in the data, but also because the users should review the "best-predicted" mappings and select the most appropriate concept to their needs.
- The combination of underlying matching techniques have been extended and fine-tuned with actual data that were exchanged through the BIF and mapped with the help of the first version of the Model Manager.
- The sample-based, machine learning techniques were trained on a rather limited scale with the data provided by the actual BIMERR applications and collected in the first release of BIF in order to make reliable mapping predictions. Such data practically refer to limited samples that typically accompany the different concepts, rather than large volumes of data, which is expected to take place during the validation activities.

4.6 Installation Instructions

The Model Mapper is served as a web application and does not require the installation of any component by the user. Detailed instructions for the Model Mapper deployment are provided in the related private code repo and all subcomponents are already available as Docker containers to speed up the process.

4.7 Usage Walkthrough

The configuration of the Model Mapper consists of an integral part of a Data Collection Job (that is described in detail in the BIMERR Deliverable D4.7 [9]), where the users (e.g., the BIMERR Application owner/developer) need to select the Mapping step as depicted in Figure 4-2, in order for the Model Mapper to be activated.


Data Collection Jobs
Assets
Search
Models
S Spiros Kousouris
Create Data Check-in Job
Cancel
Create
Job Details
Basic information about the job
Pre-processing
Before importing your data to the platform, you can use additional tools to better prepare them
Select to activate the Model Mapper

NAME: Test 3data collection

DESCRIPTION: test

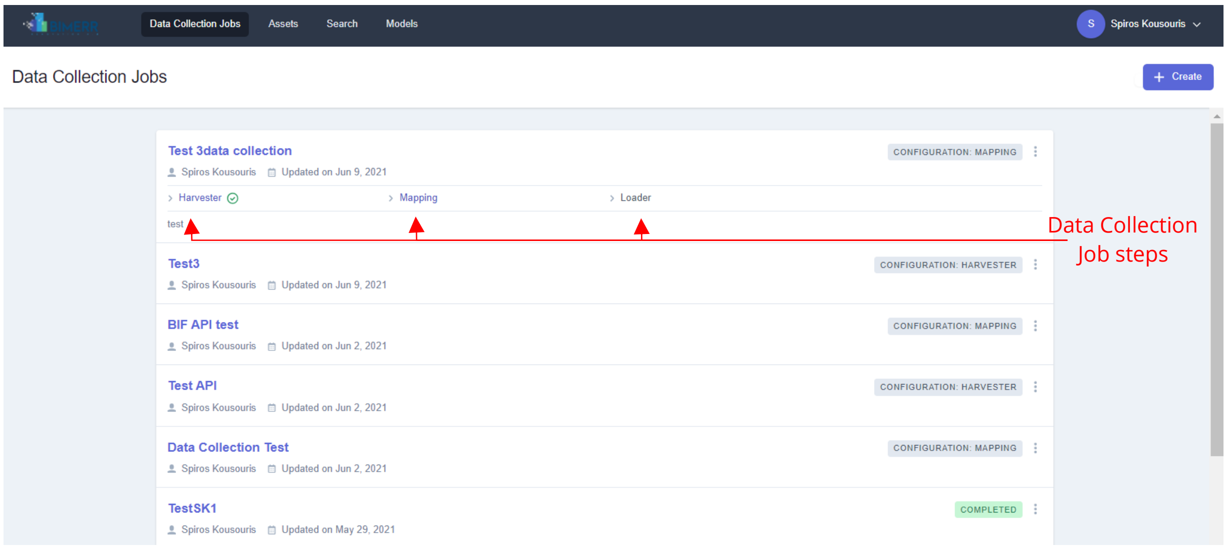
[ ] HARVESTER — Collect data through files, extnernal APIs or other services

[✓] MAPPING — Map your data to the common data model

[ ] LOADER — Load the processed data to the data storage

Figure 4-2 Activation of the Model Mapper, during the "Data Collection" job definition

Note: If the Mapping step is skipped, any data uploaded though this data collection job will be treated as a single object, meaning that in later stages it will not be possible to perform other tasks requiring mapping, such as queries on these data.

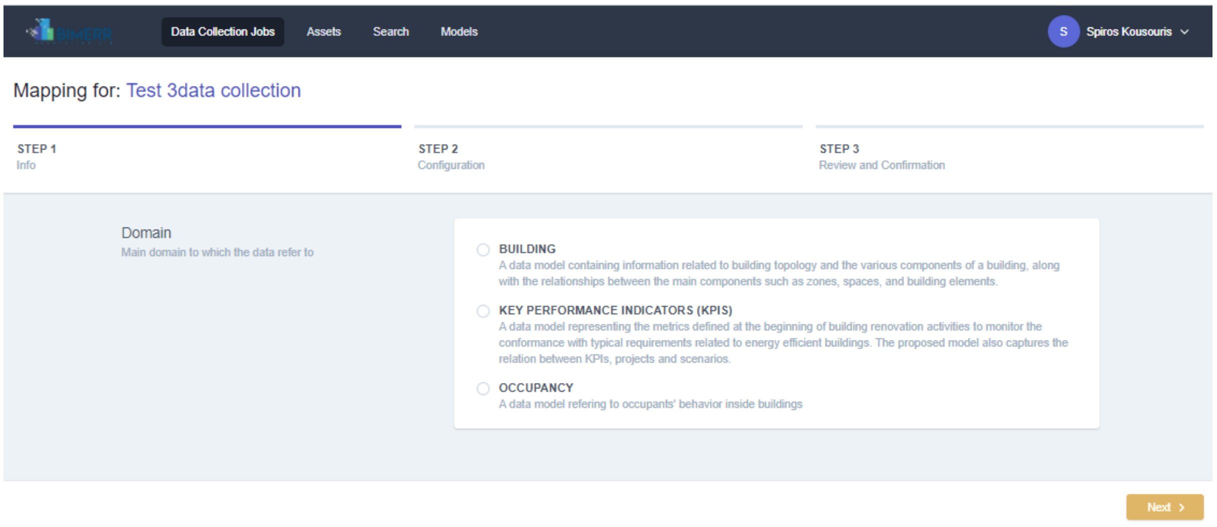

Data Collection Jobs

Test 3data collection	CONFIGURATION: MAPPING
Spiros Kousouris · Updated on Jun 9, 2021	
Harvester · Mapping · Loader	
test	
Test3	CONFIGURATION: HARVESTER
Spiros Kousouris · Updated on Jun 9, 2021	
BIF API test	CONFIGURATION: MAPPING
Spiros Kousouris · Updated on Jun 2, 2021	
Test API	CONFIGURATION: HARVESTER
Spiros Kousouris · Updated on Jun 2, 2021	
Data Collection Test	CONFIGURATION: MAPPING
Spiros Kousouris · Updated on Jun 2, 2021	
TestSK1	COMPLETED
Spiros Kousouris · Updated on May 29, 2021	

Figure 4-3 View the collection and mapping steps associated with each "Data Collection" job

As shown in Figure 4-3, by selecting a specific data collection job from the users' list, the steps (i.e., Harvester, Mapping and Loader) that have been activated are displayed and the users can create or update their configuration. Having configured the "Harvester" step, (described in D4.7), the users can select the "Mapping" step to access the Model Mapper interface.

In the Model Mapper interface (see Figure 4-4), initially the users shall select the main domain their data refer to (e.g., Occupancy)


Mapping for: Test 3data collection

STEP 1 Info · STEP 2 Configuration · STEP 3 Review and Confirmation

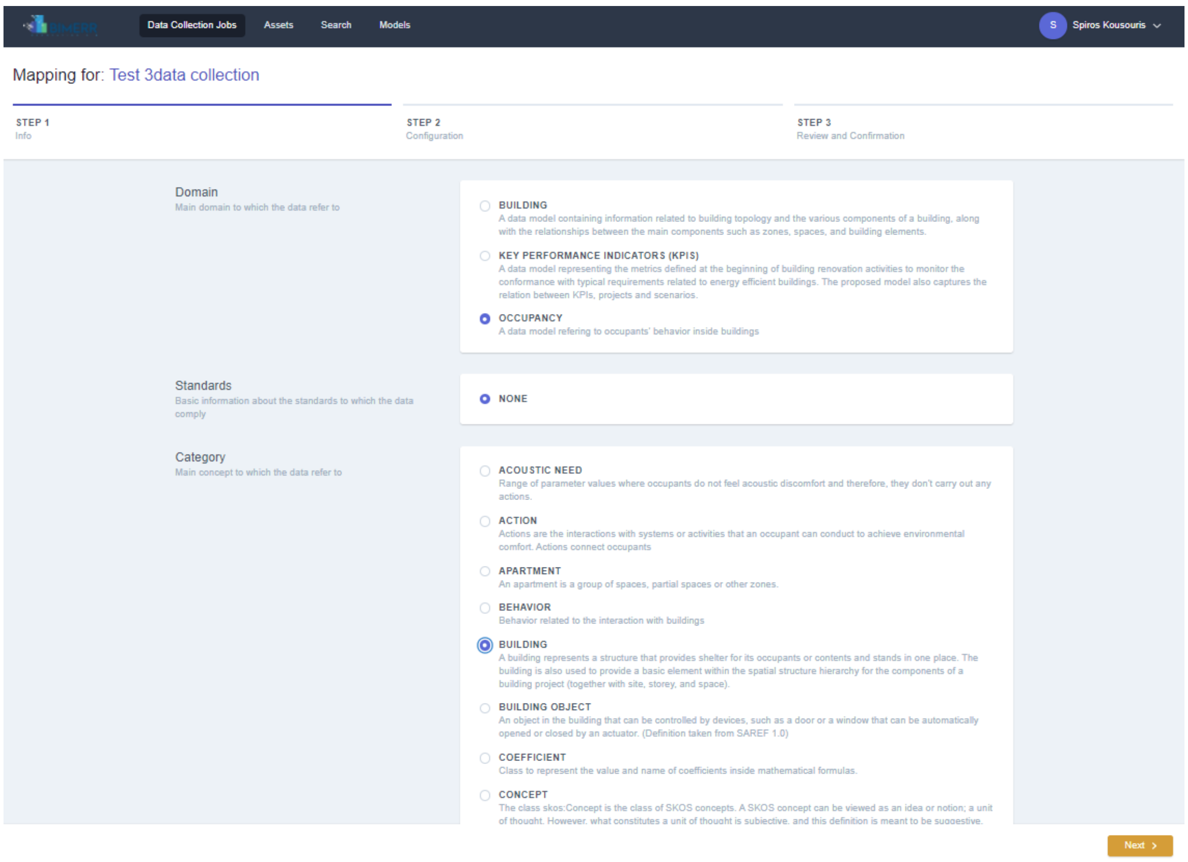
Domain
Main domain to which the data refer to

- BUILDING
  A data model containing information related to building topology and the various components of a building, along with the relationships between the main components such as zones, spaces, and building elements.
- KEY PERFORMANCE INDICATORS (KPIS)
  A data model representing the metrics defined at the beginning of building renovation activities to monitor the conformance with typical requirements related to energy efficient buildings. The proposed model also captures the relation between KPIs, projects and scenarios.
- OCCUPANCY
  A data model refering to occupants' behavior inside buildings

Figure 4- 4: Model Mapper interface Step 1a – Selection of relevant domain

Upon selection, the users are prompted to fill in the standards associated with the specific data model, (if applicable-otherwise the specific selection is not visible), and the core concept (e.g., Building) to which their data refer to (see Figure 4-5).

The listed "Domains" directly refer to the different BIMERR ontologies and data models; while the "Categories" mirror the main "parent" concepts in the respective BIMERR data model. The Model Mapper correlates the provided mapping information with the underlying source data model as extracted by the sample uploaded during the respective Data Collection Job.



Mapping for: Test 3data collection

STEP 1
Info

STEP 2
Configuration

STEP 3
Review and Confirmation

Domain
Main domain to which the data refer to

- BUILDING
  A data model containing information related to building topology and the various components of a building, along with the relationships between the main components such as zones, spaces, and building elements.
- KEY PERFORMANCE INDICATORS (KPIS)
  A data model representing the metrics defined at the beginning of building renovation activities to monitor the conformance with typical requirements related to energy efficient buildings. The proposed model also captures the relation between KPIs, projects and scenarios.
- [✓ OCCUPANCY]
  A data model refering to occupants' behavior inside buildings

Standards
Basic information about the standards to which the data comply

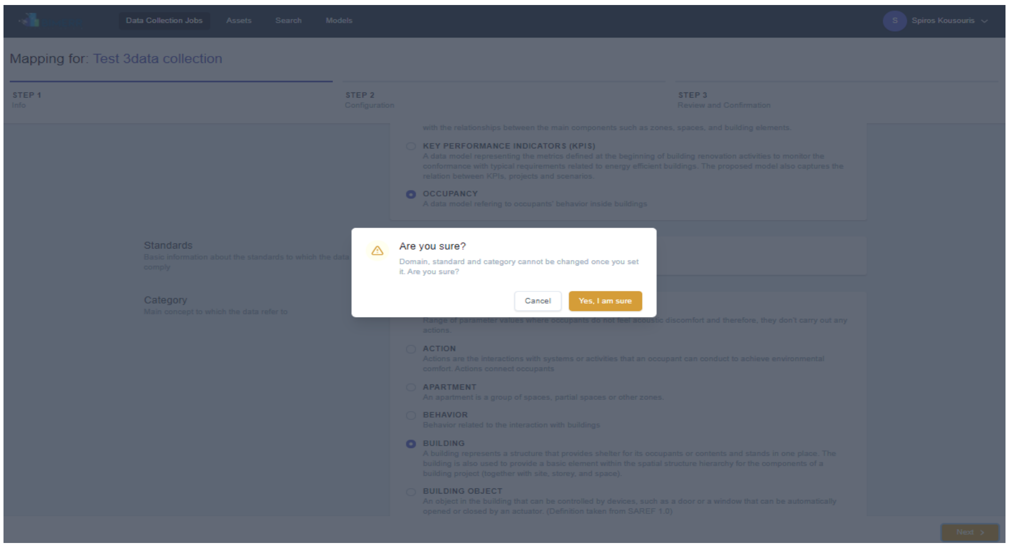
- [✓ NONE]

Category
Main concept to which the data refer to

- ACOUSTIC NEED
  Range of parameter values where occupants do not feel acoustic discomfort and therefore, they don't carry out any actions.
- ACTION
  Actions are the interactions with systems or activities that an occupant can conduct to achieve environmental comfort. Actions connect occupants
- APARTMENT
  An apartment is a group of spaces, partial spaces or other zones.
- BEHAVIOR
  Behavior related to the interaction with buildings
- [✓ BUILDING]
  A building represents a structure that provides shelter for its occupants or contents and stands in one place. The building is also used to provide a basic element within the spatial structure hierarchy for the components of a building project (together with site, storey, and space).
- BUILDING OBJECT
  An object in the building that can be controlled by devices, such as a door or a window that can be automatically opened or closed by an actuator. (Definition taken from SAREF 1.0)
- COEFFICIENT
  Class to represent the value and name of coefficients inside mathematical formulas.
- CONCEPT
  The class skos:Concept is the class of SKOS concepts. A SKOS concept can be viewed as an idea or notion; a unit of thought. However, what constitutes a unit of thought is subjective, and this definition is meant to be suggestive.

Figure 4-5 Model Mapper Step 1b – Provision of Mapping Information (domain, standard, category)

Note: As shown in Figure 4-6 the Domain, Standard and Category cannot be changed once set by the users; if changes are required at a later stage, the users will need to create a new data collection job.



Mapping for: Test 3data collection

STEP 1 Info | STEP 2 Configuration | STEP 3 Review and Confirmation

with the relationships between the main components such as zones, spaces, and building elements.

KEY PERFORMANCE INDICATORS (KPIS)
A data model representing the metrics defined at the beginning of building renovation activities to monitor the conformance with typical requirements related to energy efficient buildings. The proposed model also captures the relation between KPIs, projects and scenarios.

[✓ OCCUPANCY]
A data model refering to occupants' behavior inside buildings

Standards
Basic information about the standards to which the data comply

Category
Main concept to which the data refer to

Are you sure?
Domain, standard and category cannot be changed once you set it. Are you sure?

Cancel | Yes, I am sure

Range of parameter values where occupants do not feel acoustic discomfort and therefore, they don't carry out any actions.

ACTION
Actions are the interactions with systems or activities that an occupant can conduct to achieve environmental comfort. Actions connect occupants

APARTMENT
An apartment is a group of spaces, partial spaces or other zones.

BEHAVIOR
Behavior related to the interaction with buildings

[✓ BUILDING]
A building represents a structure that provides shelter for its occupants or contents and stands in one place. The building is also used to provide a basic element within the spatial structure hierarchy for the components of a building project (together with site, storey, and space).

BUILDING OBJECT
An object in the building that can be controlled by devices, such as a door or a window that can be automatically opened or closed by an actuator. (Definition taken from SAREF 1.0)

Next >

Figure 4-6 Model Mapper: Confirmation of selected domain, standard and category

As shown in Figure 4-7, through the “Mapping Playground” users can view the initial predictions that have been provided by the Model Mapper from the source concepts of the data that the BIF shall collect through this data collection job, to the target concepts of the respective BIMERR data model (that was selected in Step 1). The automated predictions/mappings are complemented with varying confidence levels along with color coding (i.e., green for high confidence, yellow for medium confidence and red for low confidence).

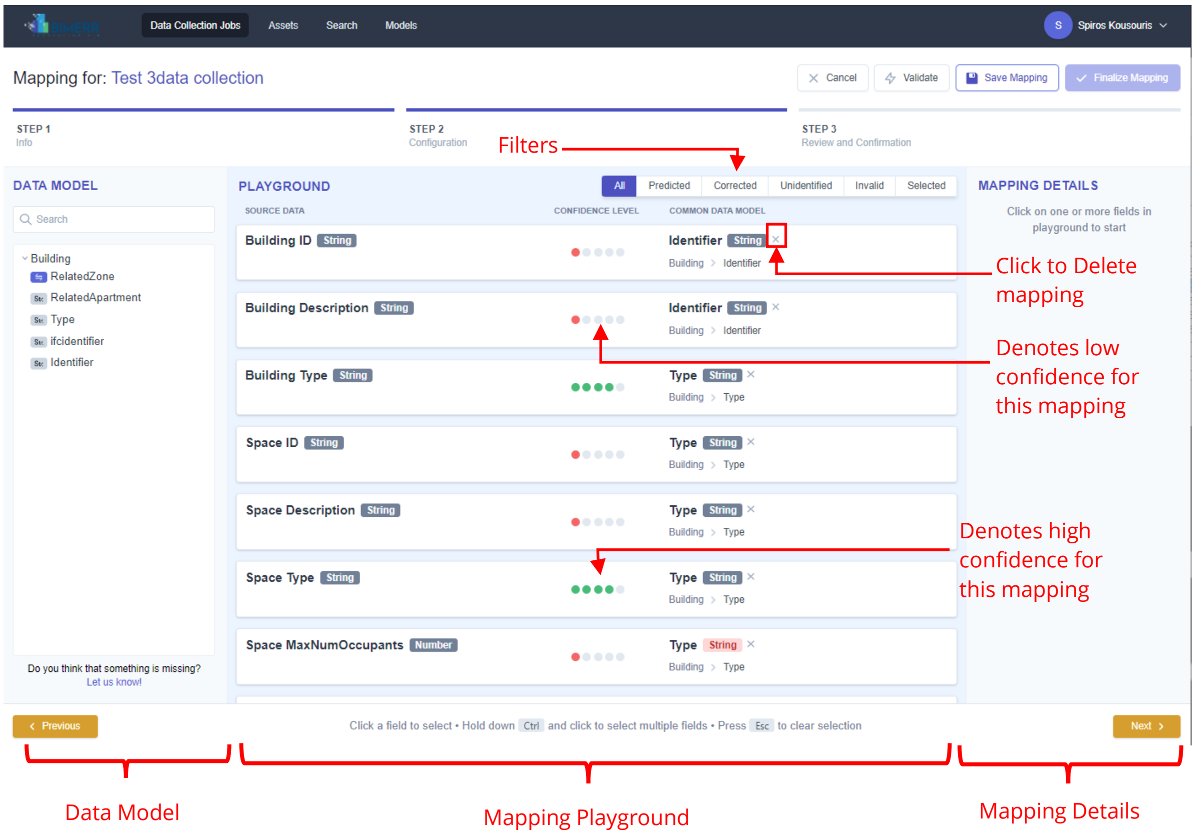

Data Collection Jobs
Assets
Search
Models
Spiros Kousouris
Mapping for: Test 3data collection
Cancel
Validate
Save Mapping
Finalize Mapping
STEP 1 Info
STEP 2 Configuration
STEP 3 Review and Confirmation
Filters
DATA MODEL
Search
Building
RelatedZone
RelatedApartment
Type
ifcidentifier
Identifier
Do you think that something is missing? Let us know!
PLAYGROUND
All
Predicted
Corrected
Unidentified
Invalid
Selected
SOURCE DATA
CONFIDENCE LEVEL
COMMON DATA MODEL
Building ID String — Identifier String — Building > Identifier
Building Description String — Identifier String — Building > Identifier
Building Type String — Type String — Building > Type
Space ID String — Type String — Building > Type
Space Description String — Type String — Building > Type
Space Type String — Type String — Building > Type
Space MaxNumOccupants Number — Type String — Building > Type
MAPPING DETAILS
Click on one or more fields in playground to start
Click to Delete mapping
Denotes low confidence for this mapping
Denotes high confidence for this mapping
Previous
Click a field to select • Hold down Ctrl and click to select multiple fields • Press Esc to clear selection
Next
Data Model
Mapping Playground
Mapping Details

Figure 4-7 Model Mapper Step 2 – Mapping Playground Overview

The users can view the data type of each concept and identify any data type mismatches, which are highlighted in red. In the Playground, users can delete any wrong mappings by selecting the "x" button on the top-right of each concept. Filtering of the mappings is also provided, so that users can quickly navigate to the Predicted / Corrected / Unidentified / Invalid / Selected mappings. In the Data Model area (left side of the screen in Figure 4-7), the users also can view the respective data model and navigate to the details of each concept by selecting it, in order to verify whether the mapping is correct.

By deleting all the wrong mappings, the users shall now manually map the concepts of their source model or update the automatically generated mappings. Users can select a source concept (this is automatically highlighted with a blue bar) from their source data in the Mapping Playground (e.g., Building ID) and drag and drop the corresponding concept (i.e., Identifier) from the Data Model as depicted in Figure 4-8.

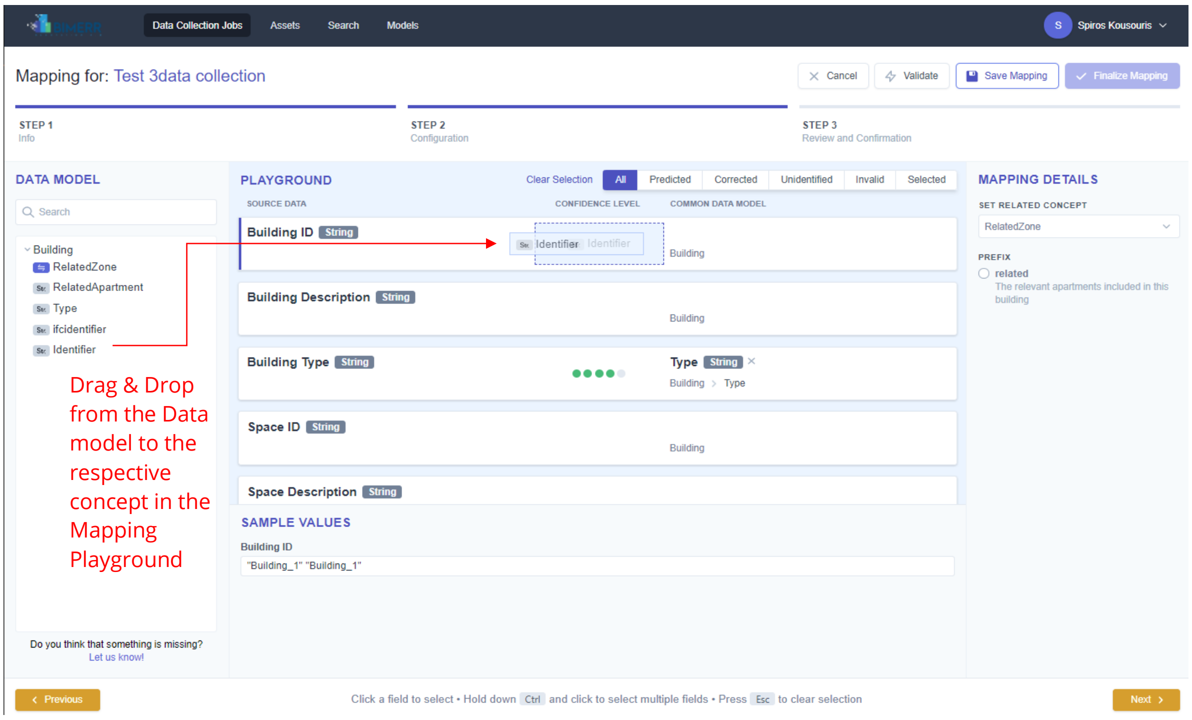


Figure 4-8: Model Mapper Step 2 – Manual mappings of concepts

The users can also manually map (i.e., set the related concept) their source data concept(s) through the Mapping Details area. As shown in Figure 4-9, in the Mapping Details, by selecting two or more fields belonging to the same concept, (e.g., Meeting ID and Meeting Duration), the users shall also identify the related concept to which these fields refer.

Upon selecting the respective concept, they need to define the appropriate prefix or add their own (if this allowed by the respective BIMERR data model). The users shall then, either set the specific related concept ("SET CONCEPT" button) or request for an updated prediction for the selected concepts ("SET & PREDICT" button). As such users can delve deeper to the provided concepts and provide the correct nesting of the source concepts selected (e.g., Building > relatedApartment > RelatedSpacesSpace > relatedMeeting.)

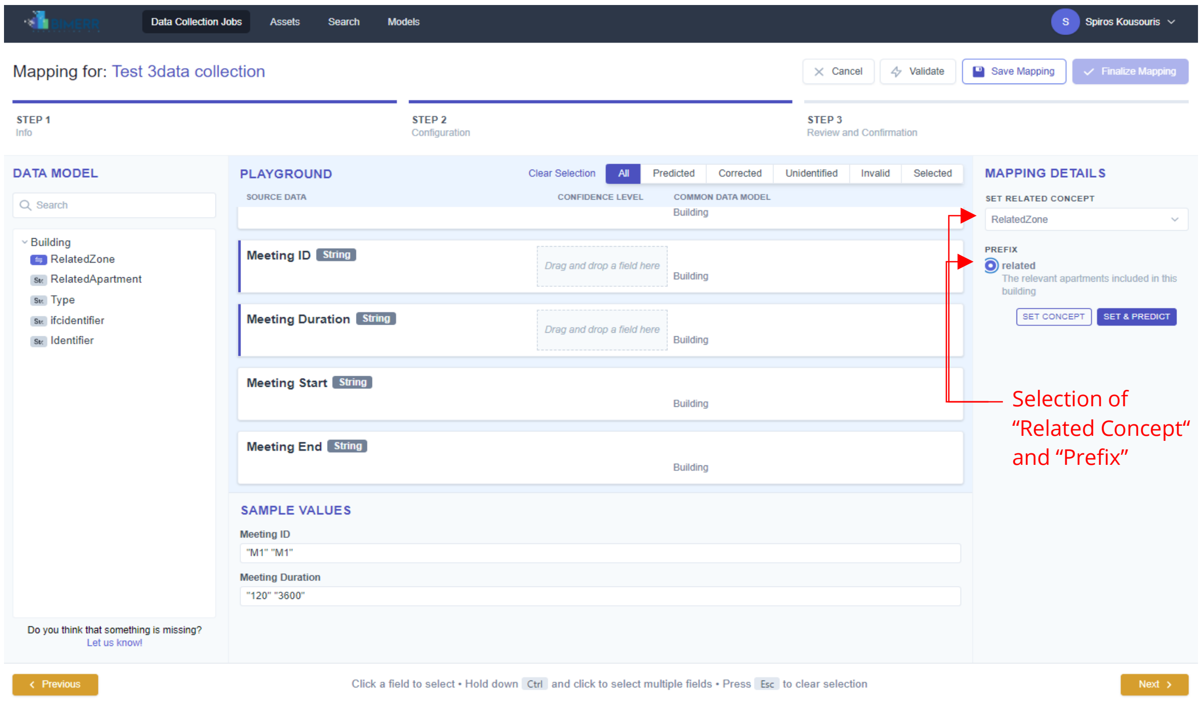
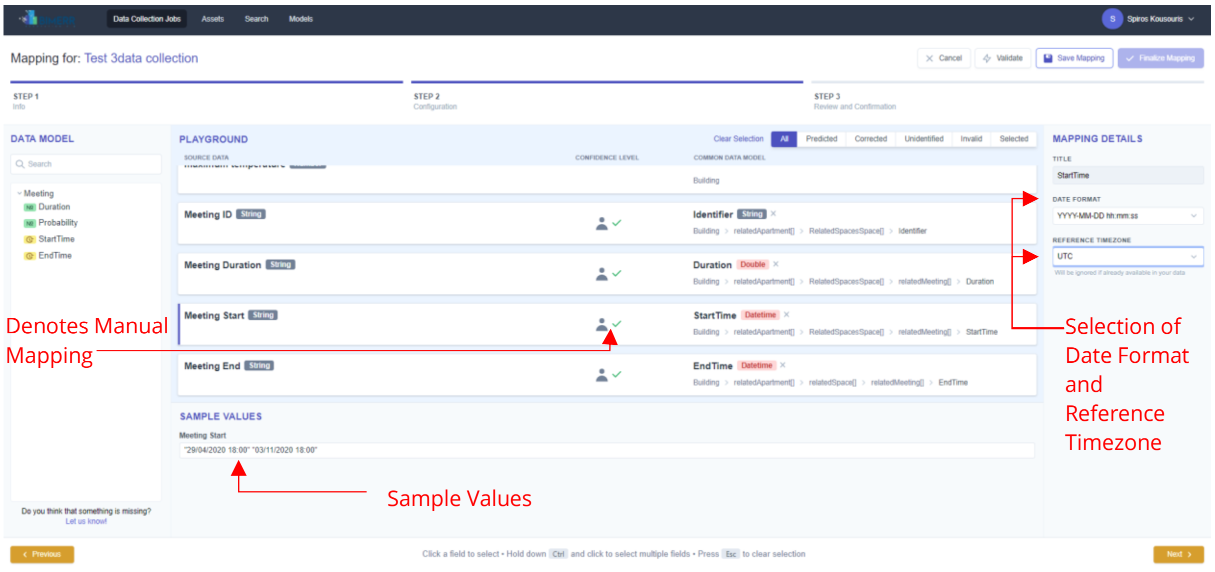


Figure 4-9 Model Mapper Step 2 – Define concept

When selecting a specific concept, the users can view the Mapping Details (right side of Figure 4-10) that are associated with the specific concept, and the sample values. For example, in case the concept refers to a datetime data type (e.g., Meeting Start), the users shall define the format from a large selection of supported datetime formats (e.g., YYYY.MM.DD hh:mm:ss, DD/MM/YYYY hh:mm AM/PM, YYYYMMDDhhmmss, etc.) and the timezone to which the data refer (e.g., UTC, Europe/Berlin, etc.). The provided mapping details will enable the mapping transformation functions to transform them to the baseline datetime format followed in the respective BIMERR data model.



Mapping for: Test 3data collection

SOURCE DATA	CONFIDENCE LEVEL	COMMON DATA MODEL
Meeting ID String		Identifier String — Building > relatedApartment[] > RelatedSpacesSpace[] > Identifier
Meeting Duration String		Duration Double — Building > relatedApartment[] > RelatedSpacesSpace[] > relatedMeeting[] > Duration
Meeting Start String		StartTime Datetime — Building > relatedApartment[] > RelatedSpacesSpace[] > relatedMeeting[] > StartTime
Meeting End String		EndTime Datetime — Building > relatedApartment[] > relatedSpace[] > relatedMeeting[] > EndTime

Data model: Meeting — Duration, Probability, StartTime, EndTime

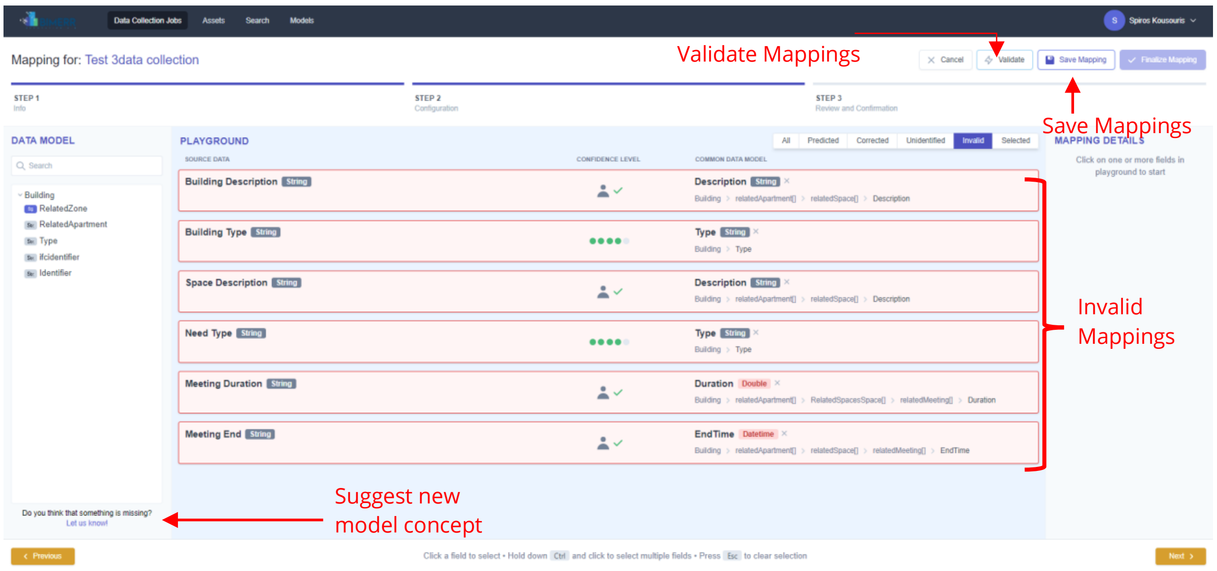
SAMPLE VALUES
Meeting Start
"29/04/2020 18:00" "03/11/2020 18:00"

MAPPING DETAILS
TITLE: StartTime
DATE FORMAT: YYYY-MM-DD hh:mm:ss
REFERENCE TIMEZONE: UTC

Click a field to select • Hold down Ctrl and click to select multiple fields • Press Esc to clear selection

Figure 4-10 Model Mapper Step 2 – View/Edit Mapping Details for a selected concept

At any time, the users can save their progress and validate the mappings provided. Any errors in the mapping validations are highlighted in red, to the users (see Figure 4-11).



Mapping for: Test 3data collection

SOURCE DATA	CONFIDENCE LEVEL	COMMON DATA MODEL
Building Description String		Description String — Building > relatedApartment[] > relatedSpace[] > Description
Building Type String	●●●●	Type String — Building > Type
Space Description String		Description String — Building > relatedApartment[] > relatedSpace[] > Description
Need Type String	●●●●	Type String — Building > Type
Meeting Duration String		Duration Double — Building > relatedApartment[] > RelatedSpacesSpace[] > relatedMeeting[] > Duration
Meeting End String		EndTime Datetime — Building > relatedApartment[] > relatedSpace[] > relatedMeeting[] > EndTime

Data model: Building — RelatedZone, RelatedApartment, Type, ifcidentifier, Identifier

Click a field to select • Hold down Ctrl and click to select multiple fields • Press Esc to clear selection

Figure 4-11 Model Mapper Step 2 – Validation of mappings and invalid mappings detection

Any remaining concepts that were not possible to be mapped to an existing BIMERR concept are indicated as "Unidentified". In this case, the users shall manually map the unidentified concepts to the BIMERR concepts or select alternative related concepts to which the mapping prediction should be executed.

Note: Any remaining unmapped concepts will be excluded from the final mapping configuration file and will be discarded from the actual mapped data that will be eventually uploaded in the BIF storage (with the help of the Building Information Collection & Enrichment Component).

In the event, the users want to propose a new model concept or update an existing one in the BIMERR data model, this can be done by selecting the "Let us know!" button as shown in Figure 4-11. Once selected in the appearing pop-up window (see Figure 4-12), users are urged to provide the name of the new model concept, its parent concept (if applicable), its description, any related term and mappings to existing models. Once the request for a new/updated model concept is submitted, it is forwarded to the Model Lifecycle Manager, where it will be handled by the data model administrator (see section 5.6, Figure 5-3).

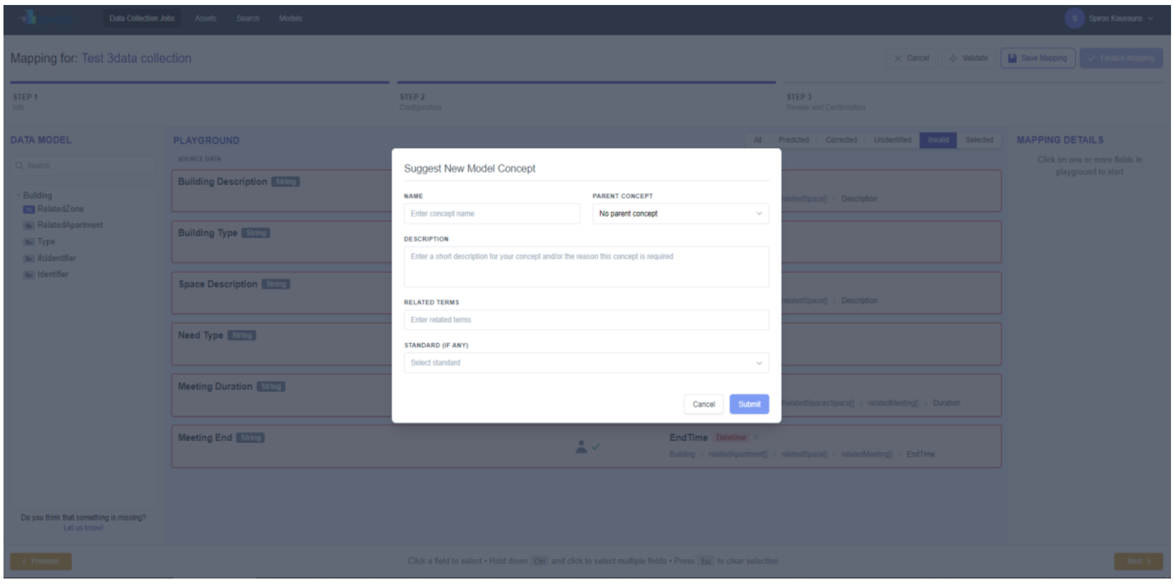
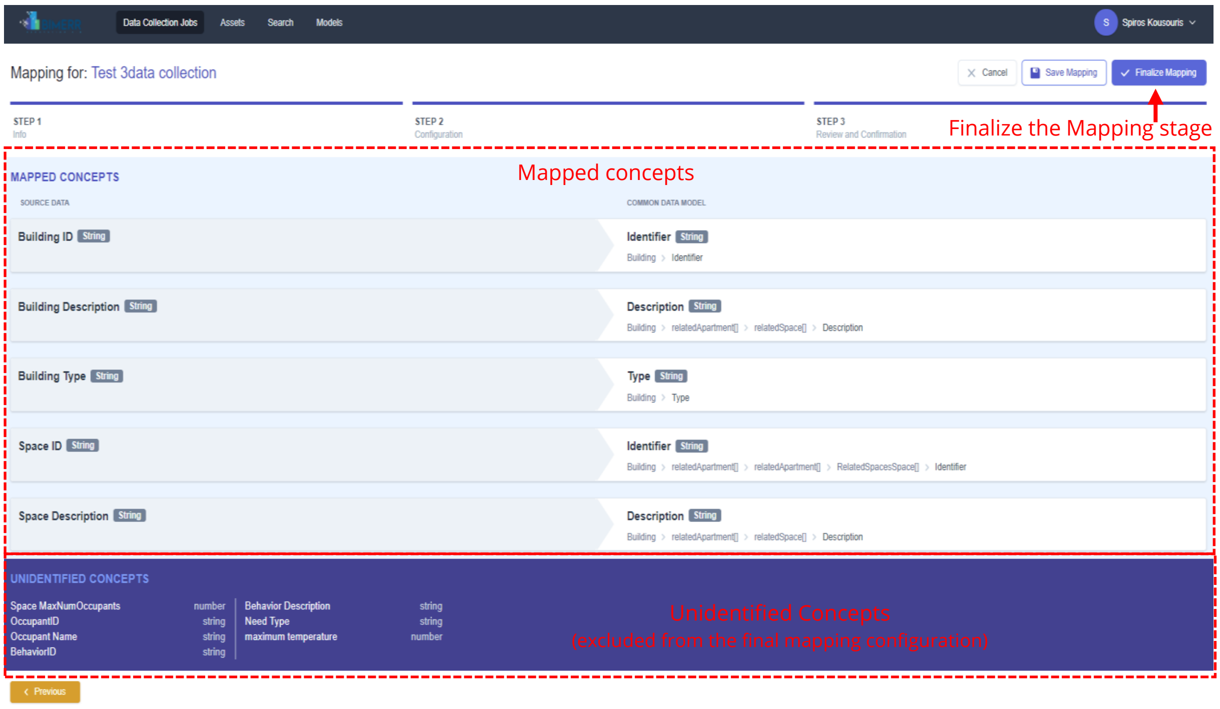


Figure 4-12 Model Mapper Step 2 - Suggest New Model Concept

Once all the mappings have been completed, by clicking "Next" the users are forwarded to the 3rd step of the Model Mapper. As shown in Figure 4-13, the users can view all the executed mappings between the BIMERR Data model (i.e. common data model) and the fields/concepts of the source data. Any "Unidentified" concepts are presented in the bottom area of the screen. Users are able to save the mapping configuration in the "Configuration File" and revisit it at a later stage or finalize it by selecting the "Finalize Mapping" button.

Note: When the users "finalize" the Mapping stage it cannot be edited any more, as the specific data collection job proceeds to the mapping execution stage.



Mapping for: Test 3data collection

STEP 1 Info — STEP 2 Configuration — STEP 3 Review and Confirmation

MAPPED CONCEPTS

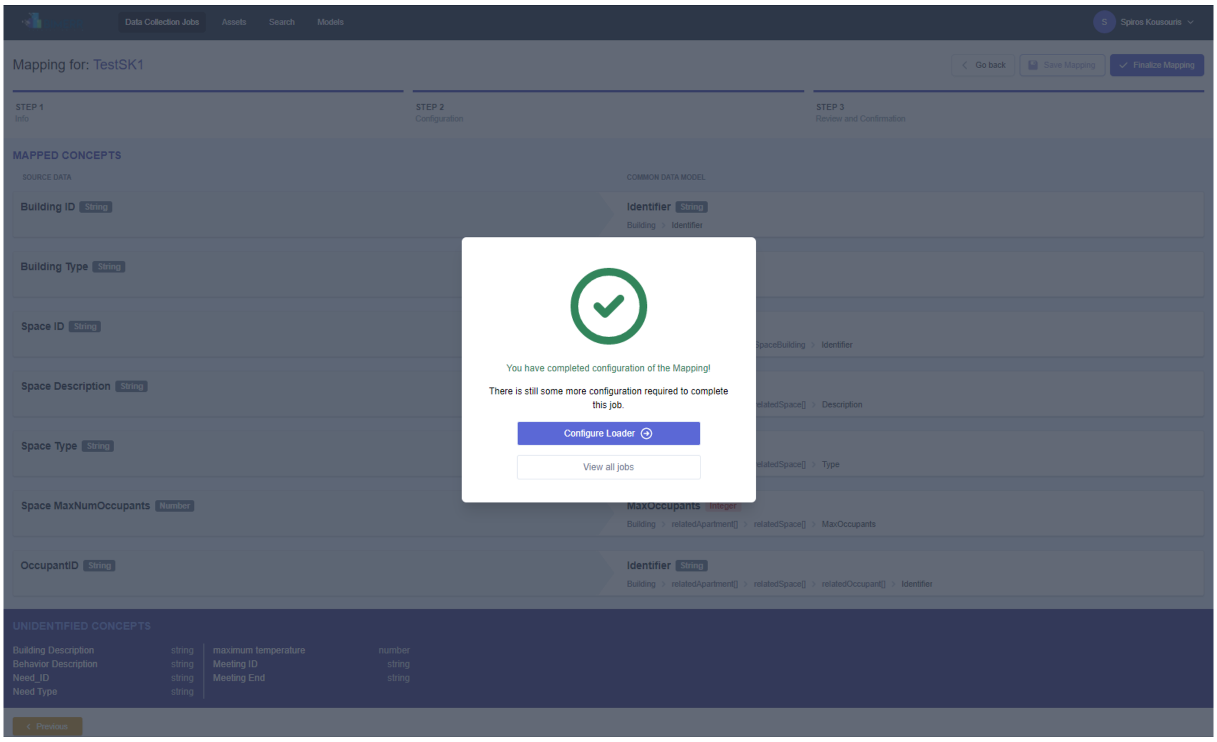
SOURCE DATA	COMMON DATA MODEL
Building ID String	Identifier String Building › Identifier
Building Description String	Description String Building › relatedApartment[] › relatedSpace[] › Description
Building Type String	Type String Building › Type
Space ID String	Identifier String Building › relatedApartment[] › relatedApartment[] › RelatedSpacesSpace[] › Identifier
Space Description String	Description String Building › relatedApartment[] › relatedSpace[] › Description

UNIDENTIFIED CONCEPTS

Space MaxNumOccupants	number	Behavior Description	string
OccupantID	string	Need Type	string
Occupant Name	string	maximum temperature	number
BehaviorID	string		

Figure 4-13 Model Mapper Step 3 – Overview of mappings between source data and BIMERR data model

Lastly, as shown in Figure 4-14



Mapping for: TestSK1

MAPPED CONCEPTS

SOURCE DATA	COMMON DATA MODEL
Building ID String	Identifier String Building > Identifier
Building Type String	
Space ID String	…paceBuilding > Identifier
Space Description String	…elatedSpace[] > Description
Space Type String	…elatedSpace[] > Type
Space MaxNumOccupants Number	MaxOccupants Integer Building > relatedApartment[] > relatedSpace[] > MaxOccupants
OccupantID String	Identifier String Building > relatedApartment[] > relatedSpace[] > relatedOccupant[] > Identifier

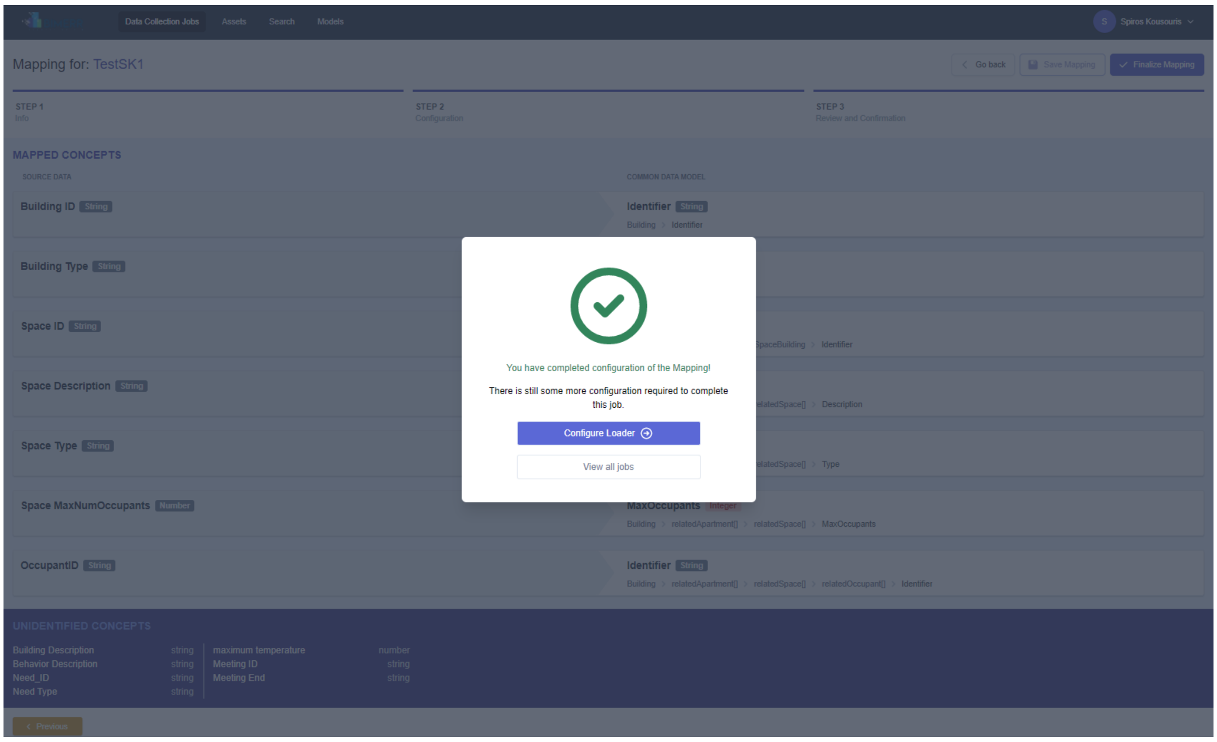
You have completed configuration of the Mapping!

There is still some more configuration required to complete this job.

UNIDENTIFIED CONCEPTS

Building Description	string	maximum temperature	number
Behavior Description	string	Meeting ID	string
Need_ID	string	Meeting End	string
Need Type	string		

Figure 4-14, upon finalization of the Mapping configuration, users are notified for its completion and asked to proceed with the configuration of the Loader or view all their generated data collection jobs (see D4.7, section 8.5)



Mapping for: TestSK1

STEP 1 Info — STEP 2 Configuration — STEP 3 Review and Confirmation

MAPPED CONCEPTS

SOURCE DATA	COMMON DATA MODEL
Building ID String	Identifier String Building > Identifier
Building Type String	
Space ID String	paceBuilding > Identifier
Space Description String	latedSpace[] > Description
Space Type String	latedSpace[] > Type
Space MaxNumOccupants Number	MaxOccupants Integer Building > relatedApartment[] > relatedSpace[] > MaxOccupants
OccupantID String	Identifier String Building > relatedApartment[] > relatedSpace[] > relatedOccupant[] > Identifier

UNIDENTIFIED CONCEPTS

Building Description	string	maximum temperature	number
Behavior Description	string	Meeting ID	string
Need_ID	string	Meeting End	string
Need Type	string		

You have completed configuration of the Mapping!

There is still some more configuration required to complete this job.

Figure 4-14 Model Mapper Step 3 – Notification for completion of the Mapping configuration

4.8 LICENSING

The BIMERR Model Mapper is a closed source component.

5. MODEL LIFECYCLE MANAGER

5.1 OVERVIEW

The Model Lifecycle Manager of the BIF Semantic Modelling Component offers a set of features for the effective management, storage and evolution of the BIMERR data model by the BIMERR data model administrator. Such a component is also responsible for ensuring that the BIMERR data model is aligned with the BIMERR ontology in cases when the data model or the ontology are updated or enriched with concepts as required by the BIMERR Applications that exchange data through the BIF, in collaboration with the Ontology Manager Framework.

More specifically, the functionalities provided by the final version of the Model Lifecycle Manager are the following:

- **Navigation to the data model's concepts hierarchy and details**: The component provides to the data model administrator a user-friendly model navigation interface, where the administrator can manage the BIMERR data model concepts' hierarchy, as well as the individual concepts and their details.
- **"Spontaneous" and "upon-request" data model evolution events (create-update-deprecate) moderated by the data model administrator**: The Model Lifecycle Manager allows the data model administrator to update the data models' concept on their own preference (e.g., considering a new data model that emerged or a change introduced in the respective ontology) or respond to the suggestions for new concepts provided by users in the Model Mapper. The data model administrator should perform any updates to existing concepts or additions of new concepts, through the user interface provided by the Model Lifecycle Manager. When the administrator has proceeded with a decision for a specific suggestion, the Model Lifecycle Manager sends an update to the Model Mapper regarding the progress and outcome of the suggestion at hand.
- **Validation of updated concepts to ensure the model's consistency**: Upon updating the BIMERR data model, the Model Lifecycle Manager shall validate that the changes introduced by the data model administrator have not generated any

unforeseen disruptions, conflicts or inconsistencies with existing concepts (e.g. by introducing a new concept with a synonym name with an existing concept).

- **Persistence of the updated data model based on appropriate versioning conventions for consistency and traceability purposes**: Depending on the type of changes that have been introduced and the extent to which they are considered as "breaking", i.e., non-backwards compatible, the Model Lifecycle Manager automatically applies appropriate versioning conventions (in terms of major and minor versions) for consistency and traceability purposes. The updated concepts are stored in the Data Storage and Indexing component (that is described in the Building Information Collection and Indexing component in D4.7 [7] but is applicable to the whole BIF).
- **Alignment between the BIMERR ontologies and data models**: The Model Lifecycle Manager allows importing an automatically extracted data model from the respective ontology through the BO2DM subcomponent of the Ontology Manager Framework (as described in Section 3). In addition, the Model Lifecycle Manager on its behalf will forward the new or updated concepts to the Ontology Manager Framework, in order for the component to update accordingly the respective ontology concepts and attributes and ensure the ontology-data model alignment.

5.2 TECHNOLOGY STACK AND IMPLEMENTATION TOOLS

The Model Lifecycle Manager builds on state-of-the art technologies across three layers:

- the Presentation Layer, containing the Model Lifecycle Manager User Interface that is developed in VueJS10 and TailwindCSS11;
- the Business Logic Layer, containing the different packages of the Model Lifecycle Manager Backend that are based in the Nest NodeJS web framework[16];
- the Data Access Layer that essentially refers to the BIF Storage and Indexing that has been set up in the context of the Building Information Collection and Enrichment

[16] https://nestjs.com/

component and utilizes ElasticSearch13 and PostgresSQL14, for the model lifecycle management needs.

Such layers along with the different technologies are depicted in the following figure.

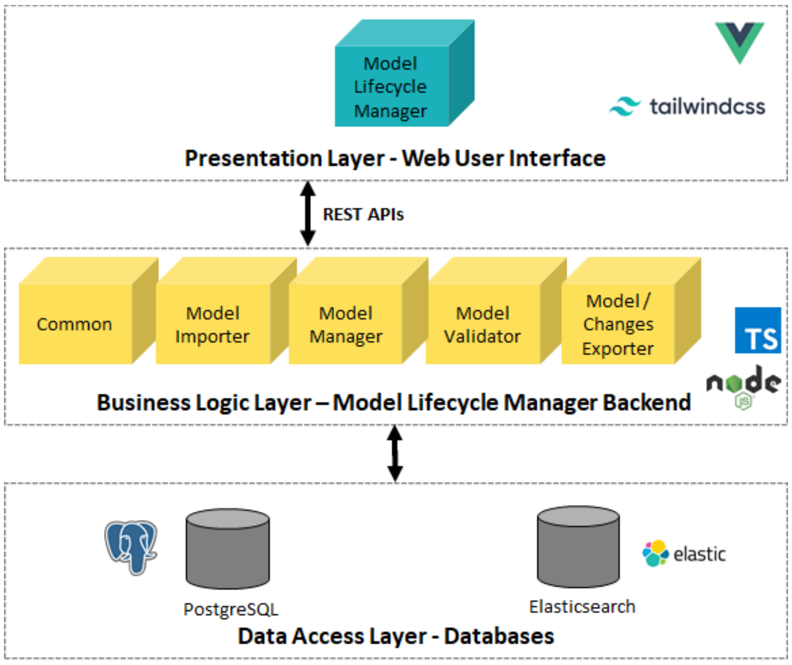


Figure 5-1: Architecture of the Model Lifecycle Manager under the BIMERR Semantic Modelling component

The Model Lifecycle Manager is written in Typescript[17] and utilizes the following open-source technologies as depicted in Table 5-1.

Table 5-1: Technologies and libraries used in the Model Lifecycle Manager, along their licenses

Name of the Library	Version	License
Nest NodeJS Web Framework	7	MIT
TypeORM	-	MIT
PostgreSQL	12.2	PostgreSQL License (similar to BSD/MIT)
Elasticsearch	7.6.0	Elastic License
Vue.js	2.6.11	MIT

[17] https://www.typescriptlang.org/

Name of the Library	Version	License
TailwindCSS	-	MIT

5.3 API Documentation

The APIs that accompany the Model Lifecycle Manager have been documented in Swagger, but they are intended only for internal use by the BIF, thus they are not presented in this section.

5.4 Changes Introduced in Final Release

In respect to the 1st release of the Model Lifecycle Manager component, a number of improvements and enhancements has been introduced in this final release. In particular, the following features and extensions have been developed upon discussions with the partners and the feedback received on the first release:

- Intuitive user interfaces for the different back-end functionalities.
- Semi-automatic evolution actions are enforced once a new concept is proposed.
- Effective management of relations between the different BIMERR data models.

5.5 Assumptions and Restrictions

In the final release of the Model Lifecycle Manager, a number of assumptions (that in certain cases, represent restrictions were taken:

- The data models are not self-standing (as in the first release) but are related to the other BIMERR data models.
- The alignment between the BIMERR ontologies and data models cannot be automatically ensured due to the inherent differences between the formats utilized (i.e. OWL and JSON). In order to ensure that the BIF will run according to the functionalities planned for semantic data mapping and transformation, the BIMERR data models require significant complementary information to be provided by the data model administrator. In parallel, when a change is introduced in the BIMERR data models, it is propagated to the ontology, but the ontology manager needs to also approve it as described in section 3.

- Over time, the BIMERR applications need change and certain evolution events, i.e. *addition* (of a new concept, a new sub-concept or a new standard), *update* (of metadata or sub-concepts) or *deletion* (of a concept, a sub-concept or their associated metadata), are anticipated as defined in the BIMERR Deliverable D4.2 [5] and D4.3 [6]. It needs to be noted that different action is taken depending on the significance and the backwards compatibility of the foreseen event. Such actions range from direct adoption (through the *Propagate* action) to compulsory validation by the data model administrator (in the *Prompt* action) and to prohibition of the specific evolution event (in the Block action).
- Deletion of certain concepts is not literally performed in the BIMERR data models as the concepts shall always be only deprecated in order to ensure backwards compatibility for data that have been already collected in the BIF.

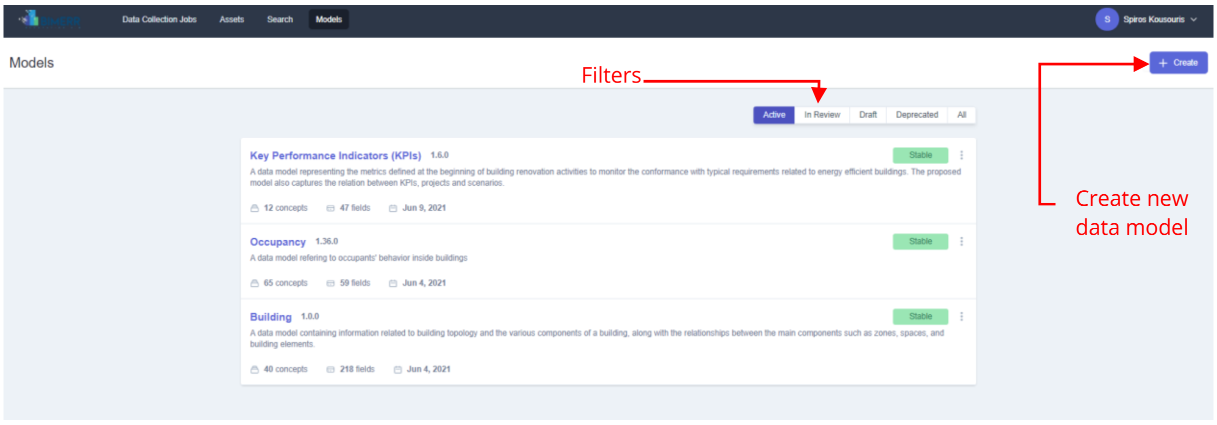
5.6 INSTALLATION INSTRUCTIONS

The Model Lifecycle Manager is intended to be served as a web application and does not require the installation of any component by the user. Detailed instructions for the deployment are provided in the related private code repo and all subcomponents are already provided as Docker containers to speed up the process.

5.7 USAGE WALKTHROUGH

The main scope of the BIMERR Model Lifecycle Manager is to enable the data model administrators to update the BIMERR Data Model. At some point, updates on the data model will be required to align with the evolution of the BIMERR ontologies or popular external data models and standards, which are often updated with new concepts, while making some other concepts obsolete. Another possible scenario where the update of the BIMERR data model may be required, is when the BIMERR application users have submitted their suggestions for new concepts or updates through the Model Mapper subcomponent (see Figure 4-11, Figure 4-12).

As shown in Figure 5-2, when the users select from BIF's main screen the "Models" tab, they are directed to the Model Lifecycle Manager's interface. Here, the users (i.e., data model administrators) can view the available data models and select the appropriate one or create a new data model. Users can also apply filters to quickly navigate to the Active/In Review/Draft/Deprecated data models.



Key Performance Indicators (KPIs) 1.6.0

A data model representing the metrics defined at the beginning of building renovation activities to monitor the conformance with typical requirements related to energy efficient buildings. The proposed model also captures the relation between KPIs, projects and scenarios.

12 concepts 47 fields Jun 9, 2021

Occupancy 1.36.0

A data model refering to occupants' behavior inside buildings

65 concepts 59 fields Jun 4, 2021

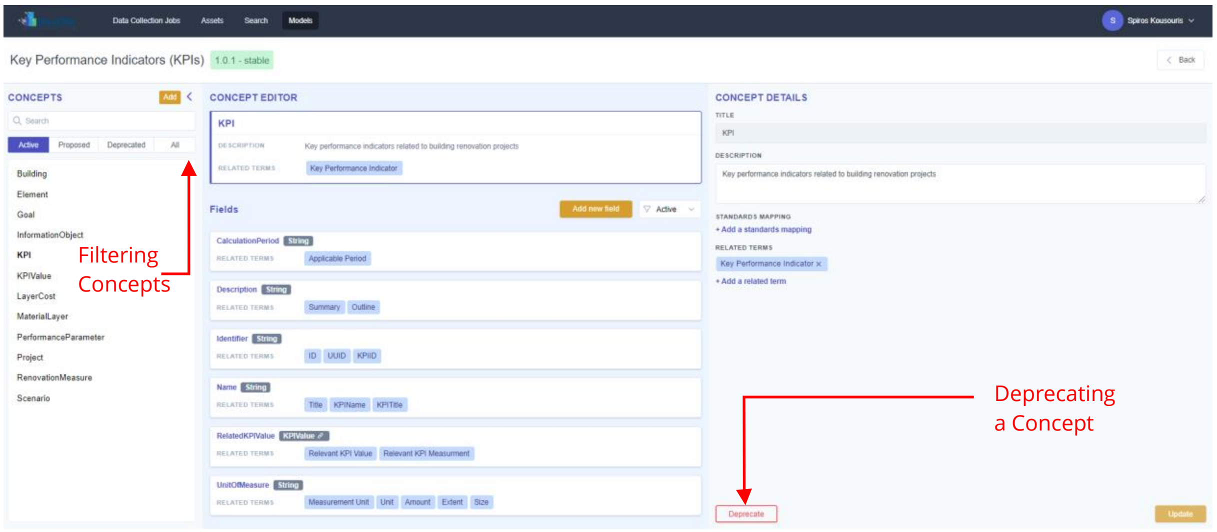
Building 1.0.0

A data model containing information related to building topology and the various components of a building, along with the relationships between the main components such as zones, spaces, and building elements.

40 concepts 218 fields Jun 4, 2021

Figure 5-2 Model Lifecycle Manager - View/Select available data models

Upon selection of a data model, e.g., Key Performance Indicators (KPIs) the users can navigate through its concepts (left side of Figure 5-3) and by selecting one specific concept (e.g., KPI) users can view its fields and their relations, along with their details (right side of Figure 5-3). Users can also apply filters to quickly navigate to all Active, Proposed and Deprecated concepts of the selected data model. The “Proposed” concepts are those suggested by the users during the process shown in Figure 4-11 (see section 4.6).



Key Performance Indicators (KPIs) 1.0.1 - stable

Concepts: Building, Element, Goal, InformationObject, KPI, KPIValue, LayerCost, MaterialLayer, PerformanceParameter, Project, RenovationMeasure, Scenario

KPI
DESCRIPTION: Key performance indicators related to building renovation projects
RELATED TERMS: Key Performance Indicator

Field	Type	Related terms
CalculationPeriod	String	Applicable Period
Description	String	Summary, Outline
Identifier	String	ID, UUID, KPIID
Name	String	Title, KPIName, KPITitle
RelatedKPIValue	KPIValue	Relevant KPI Value, Relevant KPI Measurment
UnitOfMeasure	String	Measurement Unit, Unit, Amount, Extent, Size

TITLE: KPI
DESCRIPTION: Key performance indicators related to building renovation projects
STANDARDS MAPPING
RELATED TERMS: Key Performance Indicator

Figure 5-3 Model Lifecycle Manager - View/Edit Concepts and fields of the selected data model

Note: If a concept is no longer required it can be deprecated by selecting the appropriate button; this means that this concept cannot be re-introduced in the model and users shall provide a different title in order to insert it back into the model.

In the event, the user decides to introduce a new data model need, this can be done by clicking on “+Create” button shown in Figure 5-2. The users shall then provide the title and the description of the new model, as well as define (if any) related standards.

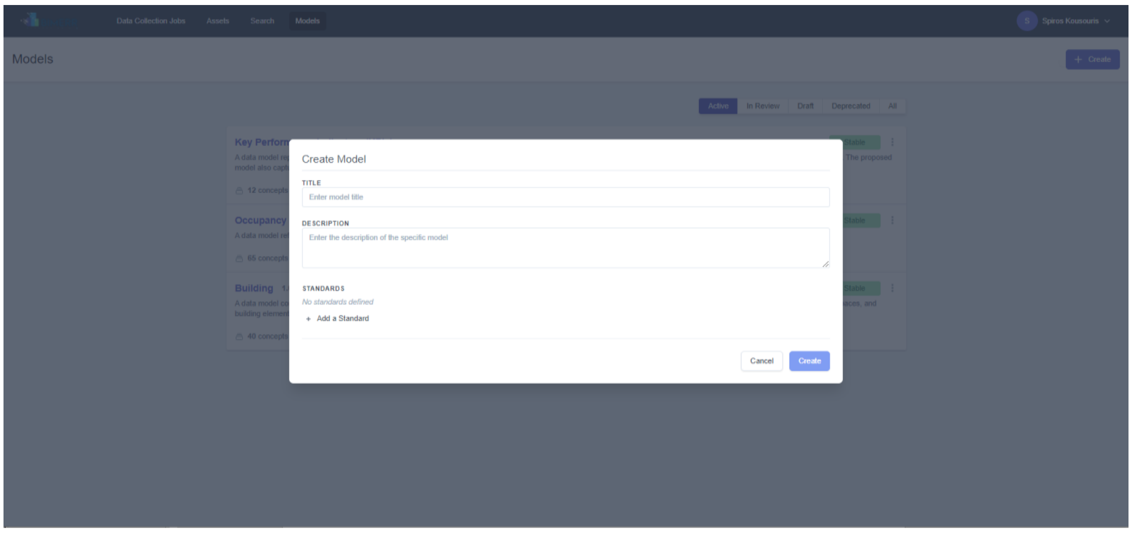


Figure 5-4 Model Lifecycle Manager – Create new data model

Upon the creation of the new data model, users can now start adding its high-level concepts through the “Concept creator” and, upon creation of the concept, can start populating this concept with its required fields. In its final release, the Model Lifecycle Manager enables users to: a) identify the introduced concept as “New Concept” (see Figure 5-5) or b) import a concept from another existing data model in BIF, along with the concept’s fields required to be copied (see Figure 5-6). The users can delete any copied fields that are not required.

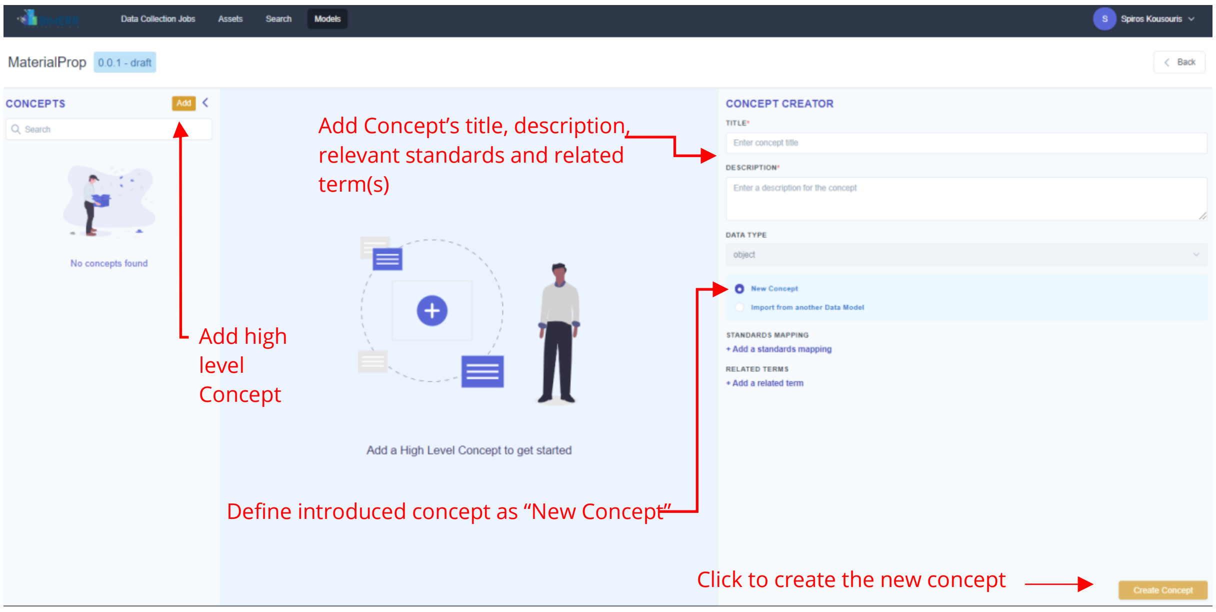


Figure 5-5: Add new concept in the data model

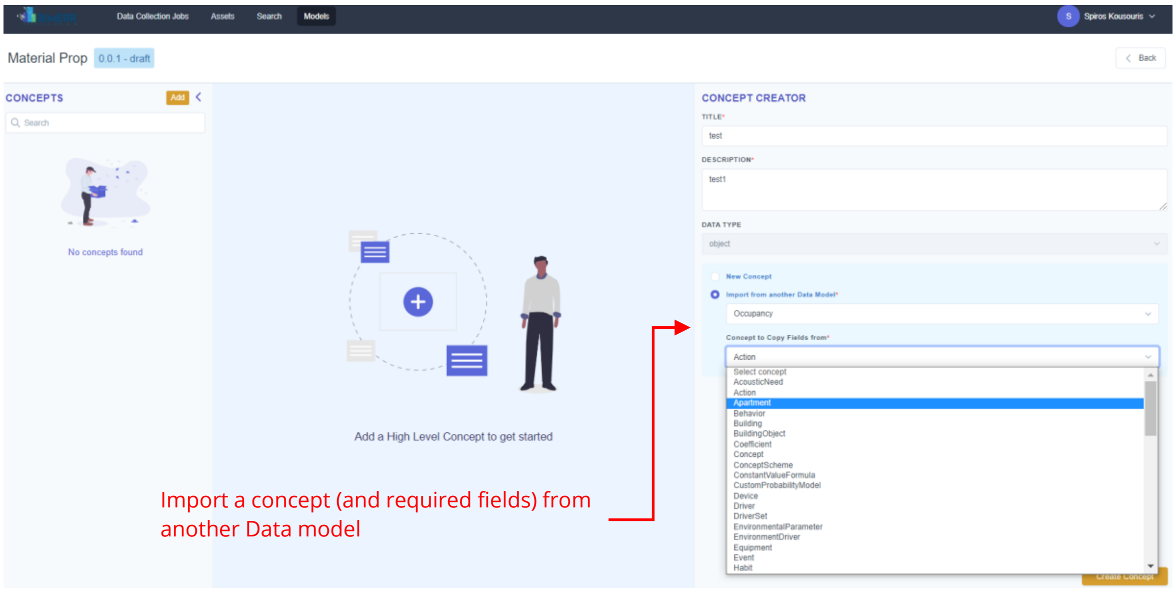


Figure 5-6 Model Lifecycle Manager – Import concept (and required fields) from other data models

If the users introduce a new concept, as shown in Figure 5-7, through the "Field creator", they shall define the title and description of the new field, its data type (e.g., object, string, double, binary, integer, Boolean, date, datetime, time) any relevant standards (if any), related term(s), as well as configure the metadata parameters (e.g., cardinality, ordering, sensitivity, indexing settings, etc.). According to the defined data type of a field (i.e. object, boolean, string, double, integer, date, datetime, time, binary), users are prompted to provide the necessary parameters (e.g., if the users define a field as an integer, the measurement type shall be also selected from the predefined list).

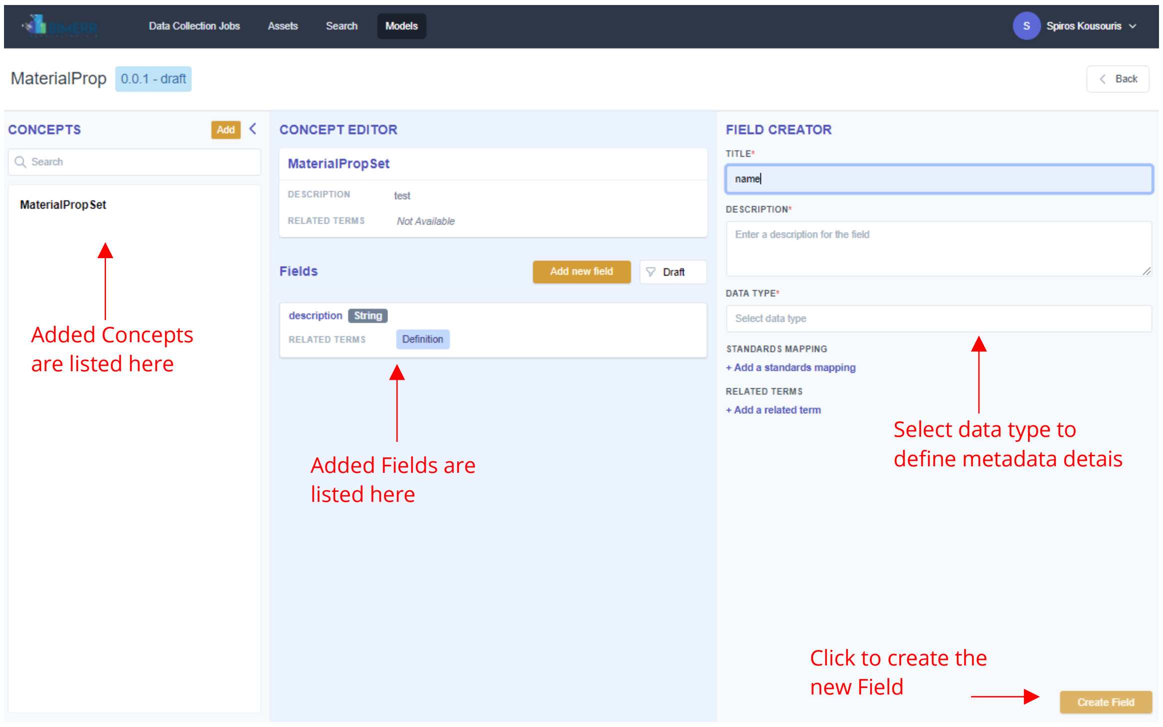


Figure 5-7 Model Lifecycle Manager – Populating a concept with new fields

Upon creation of the required Field, this will be now visible to the users (Figure 5-7) and can be further edited/deleted as described before.

5.8 LICENSING

The BIMERR Model Lifecycle Manager is a closed source component.

6. CONCLUSIONS

The BIMERR Semantic Modelling Component, along with the BIMERR Information Collection and Enrichment Component (that is documented in the BIMERR Deliverable D4.7 [7]), constitute the core of the BIMERR Interoperability Framework when it comes to building data collection, semantic mapping and reconciliation, harmonization and storage from the data providers' perspective.

As documented in the current deliverable (D4.5), the core functionalities of the final version of the BIMERR Semantic Modelling Component along its three subcomponents, namely the Ontology Manager Framework, the Model Mapper and the Model Lifecycle Manager, have been developed as planned in terms of back-end processing requirements and front-end user needs.

This final version of the BIMERR Semantic Modelling Component has been built on the outcomes of D4.4 [7], while the actual software has been refined and updated based on the feedback received during the testing of its first release, carried out in the context of the BIF integration activities of WP4 and in collaboration with the BIMERR applications developed in WP5, WP6 and WP7.

ANNEX I: BIBLIOGRAPHY

[1] BIMERR (2018) Description of Action (DoA)

[2] BIMERR (2019a) D3.1 - Stakeholder requirements for the BIMERR system

[3] BIMERR (2019b) D4.1 - Report on Semantic Alignment & Linking of EEB-related Ontologies

[4] BIMERR (2020b) D3.6 - BIMERR system architecture 2nd version

[5] BIMERR (2020b) D4.2 - BIMERR Ontology & Data Model 1

[6] BIMERR (2021a) D4.3 - BIMERR Ontology & Data Model 2

[7] BIMERR (2020b) D4.4 - BIMERR Building Semantic Modelling tool 1

[8] BIMERR (2020a) D4.6 -BIMERR Information Collection & Enrichment Tool 1

[9] BIMERR (2021a) D4.7 - BIMERR Information Collection & Enrichment Tool 2

[10] BIMERR (2020a) D4.8 -Integrated BIMERR Interoperability Framework 1