

HORIZON EUROPE PROGRAMME - HORIZON-CL5-2023-D3-01-01

Renewable Energy Valleys to increase energy security while accelerating the green transition in Europe - Innovation action (IA)



REFORMERS
RENEWABLE ENERGY VALLEYS

REFORMERS

Regional Ecosystems FOR Multiple-Energy Resilient Systems

Grant Agreement No. 101136211

Duration: 60 months | 1st November 2023 - 31st October 2028

D5.2: ENERGY DATA SPACE FOR DIGITAL TWINS



Funded by
the European Union

DOCUMENT INFO

Deliverable number D5.2

Deliverable title Energy data space for digital twins

Work Package WP5

Deliverable type Report

Dissemination level Public

Due date M24
31st October 2025

Pages 30

Document version 3.0

Lead author(s) James Allan (EMPA)

Contributors Jort Groen (TU DELFT), Edmund Widl (AIT)



DOCUMENT CHANGE HISTORY

Version	Date	Author	Description
DRAFT			
0.1	28.08.2025	James Allan, Empa	Structure, initial contents
0.2	09.10.2025	James Allan, Empa	Consolidation of input from contributors
0.3	10.10.2025	James Allan, Empa	Additional references with updated structure to reflect ontology revisions. Executive summary added.
FIRST REVIEWS			
1.0	21.10.2025	Joshua Dauda & Ida Yap, NewEnergyCoalition	Proof reading and peer review
COORDINATOR APPROVAL			
2.0	30.10.2025	Stella Arapoglou (VUB)	Coordinator review
2.1	30.10.2025	Stella Arapoglou (VUB)	Consolidated version for coordinator approval
FINAL VERSION			
3.0	30.10.2025	Stella Arapoglou (VUB)	Format review, version ready for submission

EXECUTIVE SUMMARY

This deliverable presents an ontological framework designed to drive the REFORMERS Digital Twin, which will be published as a foundational component of the DT blueprint. The framework addresses a critical challenge in energy system digitalisation: the need for flexible, semantically-rich data representation that can accommodate heterogeneous data sources while maintaining interoperability with established industry standards. Both of which are essential for a functioning Data Space.

The report documents the Digidities core ontology, developed using Web Ontology Language (OWL), which serves as the foundation for the REFORMERS-specific extension. This core ontology, publicly available on Zenodo, implements a pragmatic approach to semantic data modelling that prioritises the specific data requirements of integrated models and services rather than attempting to create a monolithic global ontology. This design philosophy directly addresses the maintenance challenges and overlapping terminology issues that hinder traditional approaches to energy system data integration.

A comprehensive attribute classification system supports diverse data types including physical measurements, costs, performance curves, categorical states, and geospatial properties. Critically, the framework implements sophisticated temporal data handling through specialised time series classes (**Historic**, **Live**, and **Future**), enabling the Digital Twin to simultaneously manage historical validation data, real-time monitoring streams, and scenario projections within unified semantic structures.

The report demonstrates systematic integration with established vocabularies including QUDT for units and quantities, FIBO for financial data, and mapping capabilities to domain standards such as CGMES. This integration strategy maintains semantic consistency while supporting interoperability across diverse energy data ecosystems. The framework's scenario management capabilities enable comparative analysis of alternative system configurations through dedicated classes and relationship mechanisms, supporting both infrastructure topology variations and parametric assumptions.

Two demonstration use cases validate the ontology's practical application: wind forecasting for the Alkmaar wind farm and urban energy simulation using CESAR-P. These implementations illustrate the complete workflow from ontology extension and data collection through standardised templates to knowledge graph construction and scenario definition. The wind forecasting case specifically demonstrates equivalence mapping between **WindTurbine** instances and CIM's **WindGeneratingUnit** class, exemplifying the framework's capacity to bridge proprietary data models with international standards.

The REFORMERS ontology will be continue to be developed to accommodate the models and services of the REFORMERS DT and will be published alongside the DT Blueprint.

TABLE OF CONTENTS

1. INTRODUCTION.....	7
2. BACKGROUND.....	7
2.1. Dataspaces.....	7
2.2. Existing Standardised Data Models.....	8
2.3. Semantic Technologies and Ontologies	8
2.3.1. CGMES	9
2.3.2. SAREF/SAREF4ENER.....	9
2.3.3. FIWARE Data Models	9
2.3.4. Open Energy Ontology.....	9
2.4. Digital Twin Data Requirements.....	10
3. OBJECTIVE.....	10
4. METHODOLOGY.....	10
4.1. REFORMERS Ontology.....	10
4.1.1. Digidities Ontology.....	10
4.1.2. Handling Interoperability.....	11
4.1.3. Component Classes	12
4.1.4. Attribute Classes	12
4.1.5. Linking Attributes to Components.....	13
4.1.6. Handling Past, Present and Future	14
4.1.7. Mapping to Data Standards.....	14
4.1.8. Defining Scenarios	16
4.2. Service Definition	17
4.3. Creating the Knowledge Graph	17
4.3.1. Classes and Attributes	17
4.3.2. Physical Relations	18
4.4. Creating Scenarios and Assumptions.....	19
4.4.1. Scenarios	19
4.4.1. Assumptions.....	20

5. USECASES 21

5.1. Wind Forecasting21

 5.1.1. Ontology Extension21

 5.1.2. System Graph22

6. CONCLUSION 23

7. ACKNOWLEDGEMENTS..... 23

8. REFERENCES..... 24

9. ANNEX 25

9.1. Example Attribute Instances25

Acronyms	
DT	Digital Twin
WP	Work Package
API	Application Programming Interface
REV	Renewable Energy Valley
GUI	Graphical User Interface
OWL	Web Ontology Language
CIM	Common Information Model
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
FIBO	Financial Industry Business Ontology
SAREF	Smart Applications REFerence ontology
QUDT	Quantity, Unit, Dimension and Type
FAIR	Findability, Accessibility, Interoperability, and Reuse
IDSA	International Data Spaces Association

CGMES	Common Grid Model Exchange Standard
BFO	Basic Formal Ontology
OEO	Open Energy Ontology
RDP	Rapid Deployment Platform

LIST OF FIGURES

Figure 1 – Preliminary REFORMERS Digital Twin Architecture. Solid arrows indicate information flow, dashed arrows indicate control.7

Figure 2: The Digicities data model stack 11

Figure 3: Mapping the instance of dici_onto:WindTurbine to the instance of WindGenerating unit in ENTSOE 16

Figure 4: Specifying the classes and attributes of the use case in a Microsoft Excel spreadsheet 18

Figure 5: The linksComponent object property and sub properties used to link physically connected infrastructure within the knowledge graph 18

Figure 6: The process of defining new scenarios. The solid lines indicate a physical relationship within the knowledge graph. The dotted lines represent scenario configurations. WT = Wind Turbine.20

Figure 7: System graph showing the connection of the wind turbines present at the Alkmaar Wind Park.23

LIST OF TABLES

Table 1 – A summary of the use cases covered in the REFORMERS extension of the Digicities ontology21

Table 2: Extension of the Digicities core ontology to accommodate the Wind Forecasting service.22

1. INTRODUCTION

This deliverable focuses on the data representation and formal definitions necessary for a Digital Twin of an Energy Valley. These data representations are a foundational component of an Energy Data Space. This work will contribute to the Digital Twin blueprint. The expected role of the Energy Dataspace in the Digital Twin Architecture of the blueprint is shown in Figure 1.

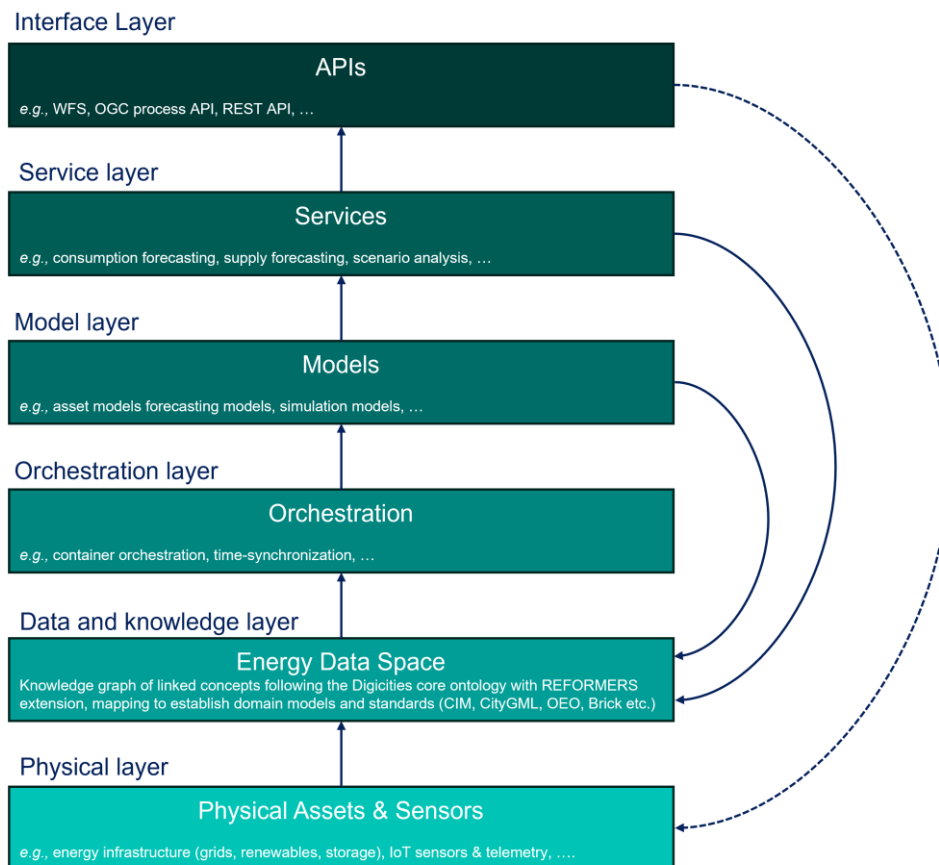


Figure 1 – Preliminary REFORMERS Digital Twin Architecture. Solid arrows indicate information flow, dashed arrows indicate control.

2. BACKGROUND

2.1. Dataspaces

As defined by the European Commission, data spaces represent “secure and privacy-preserving infrastructure to pool, access, share, process and use data”¹ typically emphasising standardised protocols for data exchange between different organisations and systems. This conventional understanding focuses on the technical and governance frameworks that enable secure data sharing across organisational boundaries, with

¹ <https://interoperable-europe.ec.europa.eu/collection/semic-support-centre/data-spaces> [10/10/2025]

initiatives like the International Data Spaces Association (IDSA) developing protocols to facilitate seamless data exchange between disparate entities. They provide a framework driven by an information model [1] to facilitate data sharing across the International Dataspaces Framework; however, specific domains require more detailed and context-specific terms, leading to the use of domain-specific vocabularies [2].

The ontology developed for the REFORMERS DT blueprint will focus on developing domain terms to drive the Digital Twin with the intention of making it useable in a dataspace.

2.2. Existing Standardised Data Models

A standard data model is a shared, structured way of representing and exchanging information across different systems, organizations, or software platforms. To avoid authors inventing their own way of naming concepts, a standard model defines:

- Entities
- Attributes
- Relationships
- Semantics

There are several standardised data models adopted in the energy sector, including CIM (Common Information Model, IEC 61970/61968/62325), CGMES (Common Grid Model Exchange Specification, ENTSO-E), CityGML with Utility ADE and Energy ADE (Application Domain Extension), OPC UA, OpenADR (Open Automated Demand Response).

Each of these standards often apply to their own specific domains within the energy sector; however, there is often considerable overlap in the entities and attributes covered in each data model. This makes interoperability a major challenge, for example, in a situation where a stakeholder adopts one model to store their data and then requires a service that requires a different model. Data conversion between models is challenging and often results in loss of information [3]. In addition to this, a lot of energy data exchange doesn't happen through these formal standards, but through ad-hoc or semi-structured files. The reason is often due to the complexity of adoption, maintenance, a lack of awareness of available standards and incompatible infrastructure [4]. Those working with data often create custom schemas (column headers, naming conventions, units) that are only consistent within their own ecosystem just to get things working [5]. The lack of standard data formats, terms, and definitions is a significant ongoing barrier to performing analytics in the field of building energy performance, resulting in more time spent processing data than analytics [6].

The REFORMERS Digital Twin aims to incorporate a diverse set of models and services that require data from many heterogeneous sources. The internal data architecture must be model agnostic but flexible to map and conform to the existing standards when necessary.

2.3. Semantic Technologies and Ontologies

Semantic technologies, built on formal semantics using Resource Description Framework (RDF), Web Ontology Language (OWL), and SPARQL query language, provide standardized vocabularies to enable semantic interoperability between disparate platforms,

and support automated reasoning over energy data. These technologies are brought together in a knowledge graph comprised of subject-predicate-object relationships representing the modelled system and its relation to the domain. This makes them ideal for use in the Digital Twin that ingests heterogeneous data and provides data to diverse services, models and use cases. Their relevance to the energy sector extends beyond simple data exchange, facilitating cross-domain collaboration, and capturing complex relationships between physical energy infrastructure, operations, and markets. There are several data models in use across the energy sector. Notable examples include: CGMES, SAREF/SAREF4ENER, FIWARE Data Models and Open Energy Ontology. A description of each is provided in the following sections:

2.3.1. CGMES

While CGMES employs RDF/XML for data representation, it represents an interesting case of semantic technology adoption without formal ontological foundations, using CIM classes as RDF classes and CIM attributes as RDF properties to enable semantic querying through SPARQL

2.3.2. SAREF/SAREF4ENER

The Smart Applications REFerence ontology, developed by ETSI Technical Committee SmartM2M, represents a comprehensive semantic approach specifically designed for IoT and smart applications with core concepts including devices, functions, commands, states, and services. The SAREF4ENER extension introduces seven distinct flexibility profile types addressing different device behaviors and energy management scenarios, with the European Commission's adoption in the Code of Conduct for Energy Smart Appliances (2024) demonstrating its practical relevance and industry acceptance.

2.3.3. FIWARE Data Models

The FIWARE platform provides a pragmatic approach to semantic data modeling focused on rapid deployment through its "data-model-in-a-week" agile standardization approach, offering comprehensive energy domain models using JSON Schema definitions with semantic annotations. In smart city deployments across 31 European cities, FIWARE data models support applications ranging from smart grids with distributed energy resources to local energy trading communities, demonstrating the value of implementation-driven development.

2.3.4. Open Energy Ontology

The Open Energy Ontology (OEO) initiative, launched in 2018 and led by a consortium including Fraunhofer IEE and Reiner Lemoine Institute, employs formal ontology engineering principles based on the Basic Formal Ontology (BFO) upper ontology with modular structure and comprehensive coverage of energy system analysis domains. According to an academic evaluation using 21 criteria across four categories, OEO ranks as the highest-quality energy ontology currently available, with success stemming from its

community-driven development model, regular release cycles, and commitment to FAIR principles [7].

2.4. Digital Twin Data Requirements

The REFORMERS DT aims to provide a blueprint in creating a digital representation of the physical infrastructure with a synchronised connection with the physical infrastructure. This definition is consistent with literature [8]. The role of the Digital Twin is to collect, process and forward data and information to models and services that develop insight for decision-making; therefore, the following processes are critical for the REFORMERS DT:

- store historic operational data,
- manage near real-time information flows,
- integrate various energy demand and renewable energy supply forecasts alongside price predictions,
- preserve the outputs of analytical model-based studies
- enable exploratory scenario analysis

3. OBJECTIVE

The report documents an approach to describe the data for the Digital Twin of the Energy Valley using the Web Ontology Language (OWL). This approach is a foundation of a data space designed to handle diverse data types essential for digital twin functionality: learning from historical data, understanding present conditions, and projecting future scenarios to support real-time decision-making processes. The system will reference historic operational data, manage near real-time information flows, track demand and renewable energy forecasts alongside price predictions, and allow exploration of different system configurations to enable the scenario analysis.

4. METHODOLOGY

4.1. REFORMERS Ontology

The data requirements of REFORMERS helped define the Digicities core ontology. The Digicities ontology is a flexible data model focusing on the individual data requirements of the models and services integrated into the Digital Twin framework. The REFORMERS extension used to drive the REFORMERS use cases will be published alongside the REFORMERS DT Blueprint.

4.1.1. Digicities Ontology

The Digicities core data model is written in the Web Ontology Language (OWL). This enables the semantic reasoning while providing an extendable process that can

accommodate a diverse range of data requirements. The Digidities core ontology that acted is extended for the REFORMERS project is published on Zenodo².

4.1.2. Handling Interoperability

As described in Section 2.2, there are several comprehensive ontologies and data models covering the energy domain; however, it is often challenging to accommodate diverse data requirements different services spanning multiple domains using a single ontology or data model. One option is to create a global ontology that imports multiple subontologies [9,10]; however, this quickly gets complicated when there are overlapping terms and is difficult to maintain, especially when the individual subontologies get updated or aren't maintained. The Digidities core aims to address these issues by focusing on the data requirements of model and services, such as those integrated into the REFORMERS Digital Twin, and establishing a flexible process to accommodate different data requirements while also providing a potential path for standardisation. The connected stack of data models considered in Digidities and the trade-off between flexibility and standardisation is shown in Figure 2.

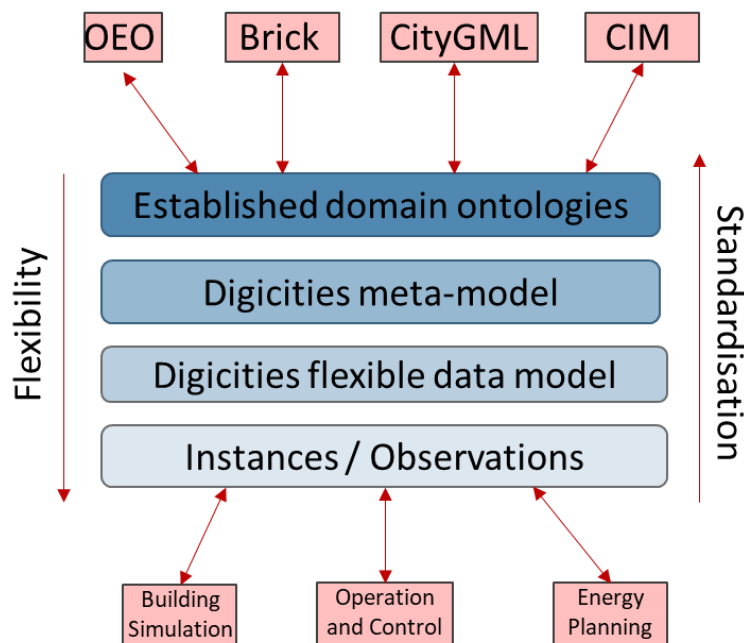


Figure 2: The Digidities data model stack

The components of the ontology are explained below.

²https://zenodo.org/records/17310915?token=eyJhbGciOiJIUzUxMiJ9.eyJpZCI6ImMyMTRkZTgzLWExOWltNDBiZC04M2QwLTFiMTFhM2RjZTQzZSIsImRhGEiOnt9LCJyYW5kb20iOiJlOGZiMjdhMzVkYzJiMTRmZmI5ODVmODkyMjk2YmYzZSJ9.5kTfxESZbaSJJxcHjRIDRc5OvqO5ZaePHbKYs3BEIWQFiQp2TUUVdCRGUoKDxxZdx20_4884nN-Jq1Rkn_GHug

4.1.3. Component Classes

The architecture establishes five primary component categories based on functional analysis of energy systems. **Physical Infrastructure** components encompass devices, networks, and storage systems that constitute tangible energy system elements. **Process Components** represent transformational activities including energy conversion and spatial transport. **Flow Components** capture dynamic transfers of energy carriers, materials, and information between system elements. **Spatial Components** provide geographic context through location classes. **Actor Components** represent stakeholder entities involved in system ownership, operation, and regulation.

The **Device** classification includes sensors for measurement, actuators for control, controllers for coordination, meters for consumption tracking, and switches for flow interruption. The **Process** hierarchy distinguishes conversion processes (energy transformation) from transport processes (spatial movement) and storage processes (temporal shifting). **Flow** components differentiate between **Energy Carrier Flows**, **Material Flows**, and **Information** flows, each with domain-specific subclasses. This organization aims to offer sufficient semantic precision for diverse energy applications.

The hierarchical structure implements inheritance relationships where universal properties apply to all component types through the root class, while specialized properties address specific component categories. This design pattern ensures consistent semantic behavior across the taxonomy while accommodating domain-specific modeling requirements.

4.1.4. Attribute Classes

The attribute classification system implements a comprehensive taxonomy for representing properties across energy system components. The hierarchy addresses measurement types, temporal behaviour, and domain-specific requirements through systematic categorization rooted in the **dici_onto:Attribute** class. The system employs a flexible categorical approach where attributes can possess both static and dynamic characteristics simultaneously, rather than enforcing strict temporal separation.

The attribute taxonomy implements a categorical classification system where attributes are organized by their fundamental data type and representation requirements rather than temporal behaviour. This approach recognizes that temporal characteristics (static versus dynamic) apply across attribute categories, enabling more flexible modeling patterns where individual attributes can exhibit both time-invariant and time-varying properties within unified semantic structures.

Physical Attributes represent quantitative properties with defined measurement units, integrating with QUDT vocabularies through the *hasDefaultUnit* property. These attributes capture measurable physical quantities such as power, temperature, pressure, and flow rates. **Cost** Attributes address economic properties through two primary subclasses: **SimpleCostAttribute** for fixed monetary values and **UnitBasedCostAttribute** for unit-based pricing structures such as energy costs per kilowatt-hour. Both subclasses support currency specification through integration with FIBO currency vocabularies.

Curve Attributes represent functional relationships between variables through coordinate pair collections. The ontology enforces mandatory x-axis and y-axis unit specifications through blank node structures that associate *xUnit* and *yUnit* properties with the attribute's *hasDefaultUnit* property. This mechanism enables precise definition of performance curves, efficiency characteristics, and operational relationships where dependent variables relate to independent variables.

Custom Physical Ratio Attributes extend the quantitative modeling capabilities by supporting ratio-based measurements with distinct numerator and denominator units. This is to support physical ratios that are not present in the QUDT ontology. The system stores ratio unit specifications through blank nodes containing *numeratorUnit* and *denominatorUnit* properties, enabling representation of derived quantities such as energy intensity (energy per unit area) or specific energy consumption (energy per unit production).

Categorical Attributes handle discrete states and classifications through controlled vocabularies, supporting operational states, equipment classifications, and enumerated system configurations. **Geospatial Attributes** address location-based properties including coordinates and spatial relationships, providing integration capabilities with geographic information systems for spatial analysis applications. The intention is to support geospatial concepts such as **Point**, **Line**, **Polygon** and **Multipolygon**; however, the current version of the ontology does not accommodate these concepts.

Event Attributes capture temporal occurrences with configurable precision levels ranging from annual to timestamp resolution. The *hasDefaultTemporalPrecision* property specifies the granularity of temporal representation (*Year*, *YearMonth*, *Date*, or *DateTime*), accommodating diverse temporal modeling requirements from historical analysis to operational event tracking. **Simple Value Attributes** represent unitless quantities or dimensionless properties that require neither unit specifications nor temporal precision definitions. Examples of simple values could be model configuration parameters and counts.

The attribute creation process enforces type-specific validation requirements: physical and geospatial attributes require unit specifications, cost attributes necessitate unit or currency definitions depending on subclass, curve attributes demand both x and y axis units, ratio attributes require numerator and denominator units, and event attributes must specify temporal precision. These validation constraints ensure semantic consistency across the attribute taxonomy while supporting diverse modeling requirements necessary to support the data requirements of different models and services.

Example ttl structures for all supported attributes are provided in the Annex.

4.1.5. Linking Attributes to Components

The ontology implements attribute-component relationships through hierarchical object properties that enforce semantic constraints and maintain type safety. The linking system follows systematic naming conventions and inheritance patterns to establish consistent relationship semantics across the component taxonomy.

The root property *hasAttribute* establishes the universal relationship between components and attributes, with domain restriction to **Component** and range restriction to **Attribute**. Specialized subproperties implement the pattern *has[ComponentType]Attribute* with corresponding domain and range restrictions that enforce type-specific associations.

Component-specific properties such as *hasDeviceAttribute* restrict their domain to specific component types (e.g., **Device**) and their range to corresponding attribute types (e.g., **DeviceAttribute**). This constraint mechanism prevents inappropriate attribute assignments and ensures semantic consistency across the knowledge graph.

The temporal data linking mechanism connects dynamic attributes to time series representations through the *hasTimeSeries* property and its specialized subproperties. Inverse properties such as *isTimeSeriesOf* enable bidirectional navigation through the knowledge graph, supporting complex queries and relationship discovery patterns.

Domain-specific linking properties address specialized relationships, such as *hasEnergyCarrierEnergyCostAttribute* which specifically associates energy carriers with their cost attributes. This specialization pattern accommodates domain-specific constraints while maintaining inheritance from general linking properties.

4.1.6. Handling Past, Present and Future

The ontology addresses temporal data representation through specialized time series classes and linking mechanisms that distinguish between historical observations, real-time measurements, and future projections. This temporal framework supports comprehensive time horizon modeling within unified semantic structures.

The temporal modeling framework implements three distinct time series subclasses that inherit common temporal metadata properties from the **TimeSeries** class. Each subclass addresses specific temporal characteristics: **HistoricTimeSeries** represents measured or observed past data, **LiveTimeSeries** captures real-time or near real-time measurements, and **FutureTimeSeries** contains forecasted or projected values.

Common time series properties include temporal bounds (*startTime*, *endTime*), *temporalResolution*, and data location information (*storedAt*, *hasFileName*).

Dynamic attributes maintain simultaneous references to multiple temporal variants through specialized data properties (*hasHistoricTimeSeriesReference*, *hasLiveTimeSeriesReference*, *hasFutureTimeSeriesReference*) and corresponding object properties that link to time series instances. This approach supports comprehensive temporal analysis where individual attributes can reference historical validation data, real-time monitoring streams, and future scenario projections within the same data environment.

The temporal framework accommodates different data access patterns and storage mechanisms appropriate to each temporal context while maintaining consistent semantic representation across time horizons.

4.1.7. Mapping to Data Standards



The ontology incorporates established vocabularies and implements systematic mapping capabilities to support interoperability with domain standards. This integration approach maintains semantic consistency while accommodating diverse data ecosystems and analytical frameworks commonly employed in energy system modeling.

The ontology integrates QUDT (Quantities, Units, Dimensions and Data Types) vocabularies for standardized unit and quantity representation. The *hasDefaultUnit* annotation property links attributes to QUDT unit concepts, while the *qudt:unit* object property enables runtime unit specifications. This integration supports dimensional analysis and unit conversion capabilities required for quantitative energy system modeling.

Financial data representation utilizes FIBO currency vocabularies through the currency property in cost attributes. This integration ensures standardized currency handling across international energy system applications.

The mapping framework implements three relationship types for external vocabulary alignment: equivalence relationships (*owl:equivalentClass*, *owl:equivalentProperty*) indicate semantic identity, hierarchical relationships (*rdfs:subClassOf*, *rdfs:subPropertyOf*) establish taxonomic connections, and similarity relationships (*skos:closeMatch*) express semantic proximity without strict equivalence.

The process is compatible with mapping to other established data models. These mapping assertions alongside the ontology following systematic naming conventions, enabling multiple external standard integrations within single knowledge bases. This approach supports energy system models that must conform to diverse domain standards simultaneously, such as building energy models requiring SAREF compliance alongside electrical grid models utilizing CIM standards.

To demonstrate this process, an example mapping and query was implemented to map the classes and attributes in the ENTSOE CGMES³. An instance of a **WindTurbine** was mapped to the component **WindGeneratingUnit** in the CGMES. As a result, the attributes defined in the CGMES become available to the instance, enabling interoperability between the ontology and the standard. The link between the instance and CGMES is shown in Figure 3.

³ ENTSOE_CGMES_v2.4.15_16Feb2016_RDFS source: <https://www.entsoe.eu/data/cim/cim-for-grid-models-exchange/> Accessed [13/05/2025]

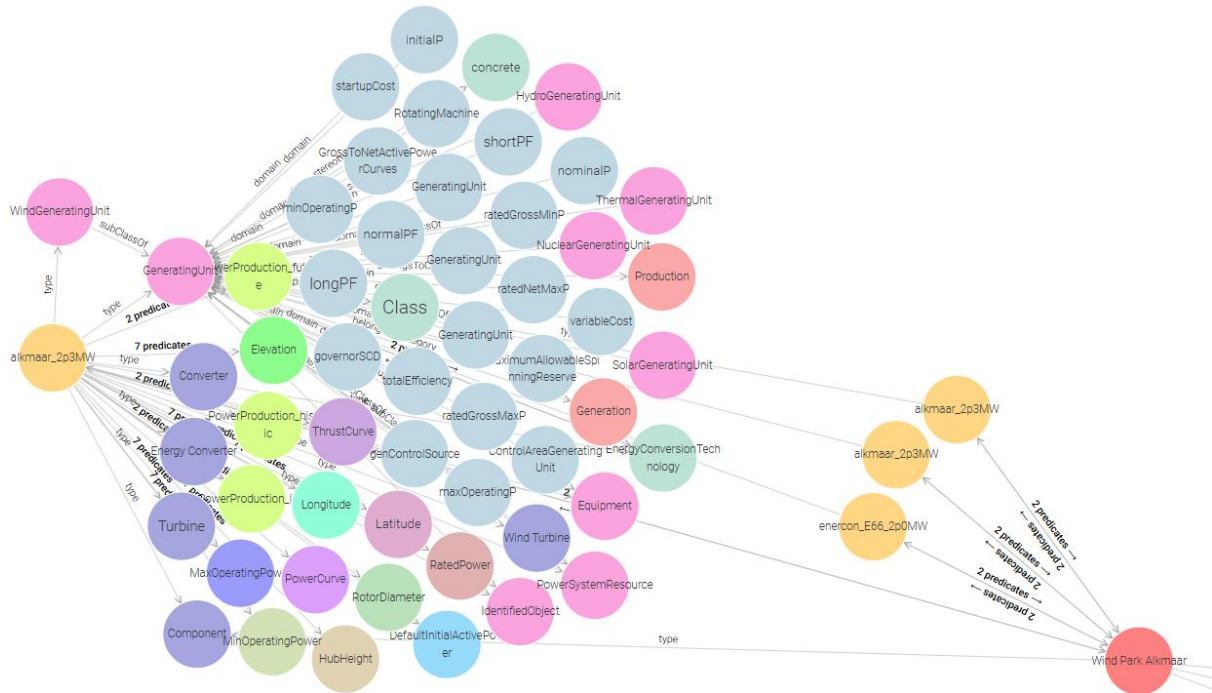


Figure 3: Mapping the instance of `dici_onto:WindTurbine` to the instance of `WindGeneratingUnit` in ENTSOE

Note, this link is established in the terminology layer and the semantic reasoner automatically deduces that the instance of **Wind Turbine** is equivalent to the ENTSOE **WindGeneratingUnit**.

4.1.8. Defining Scenarios

The ontology implements scenario management through dedicated classes and relationship mechanisms that support alternative system configurations and comparative analysis. This framework addresses energy planning applications requiring multiple system variants and temporal modeling scenarios.

The **Scenario** class serves as the foundation for alternative system representation, with the *derivedFrom* property establishing hierarchical relationships between scenarios. This mechanism supports incremental scenario development where derivative scenarios inherit base scenario characteristics while introducing specific modifications.

Component inclusion in scenarios operates through the *usedInScenario* property, which applies to components, attributes, and component links. This selective activation mechanism enables different system configurations within unified knowledge bases, supporting comparative analysis of alternative energy system designs or operational strategies.

The **ComponentLink** class represents relationships between components that may vary across scenarios. Properties *hasInputEntity* and *linksInputEntityTo* define component

connections with scenario-specific applicability, enabling representation of different system topologies, operational patterns, or infrastructure configurations.

The **Assumption** framework implements systematic scenario parameterization through the **Assumption** class and its subclasses. Properties such as *targetComponent*, *targetAttribute*, and modifier enable precise specification of parameter modifications, supporting scenario definition through assumption sets rather than complete model duplication.

4.2. Service Definition

This section details the process of defining service requirements within Digidities, the same principle could be adopted in REFORMERS to indicate the components and attributes required for each service. The data requirements for each service are described using a YAML that references the components and attributes of the Digidities Ontology. The structure begins with a root *service_name* identifier and a *scenario_data* container that encapsulates all component definitions and their relationships. Each component entry in the YAML corresponds to a class from the Digidities ontology that inherits from the base **Component** class, with root-level components defined using direct URI and label references (e.g., **ComponentType.URI**, **ComponentType.label**), while child components utilize a link property containing connection patterns following the format **CL.ParentComponent.ChildComponent** and a nested template structure for their properties. The attribute mapping within each component reflects the ontology's distinction between static and dynamic attributes, where static attributes are represented as direct property references (e.g., **ComponentType.attributeName**), and dynamic attributes incorporate time series references through specialized properties (*hasHistoricTimeSeriesReference*, *hasLiveTimeSeriesReference*, *hasFutureTimeSeriesReference*) that correspond to subclasses of **DynamicAttribute** in the ontology. This structure ensures that the YAML maintains semantic consistency with the Digidities ontology's formal definitions while providing a practical specification format for service data requirements, with component-attribute relationships validated against the ontology's domain and range restrictions as queried from the graph database (Ontotext GraphDB).

4.3. Creating the Knowledge Graph

The knowledge graph of the physical replica comprises of two main components:

- Classes and attributes
- Physical relations

The process of building the graph for each component are detailed in the following subsections.

4.3.1. Classes and Attributes

Individual instances to include in the graph were collected using an Excel table. This specifies the Component name, the Attribute and the information necessary to populate the semantic structure defined in the ontology. This data collection process is shown in Figure

4. The knowledge graph of the components and attributes was then created following the structure in 4.1.3 and 4.1.4.

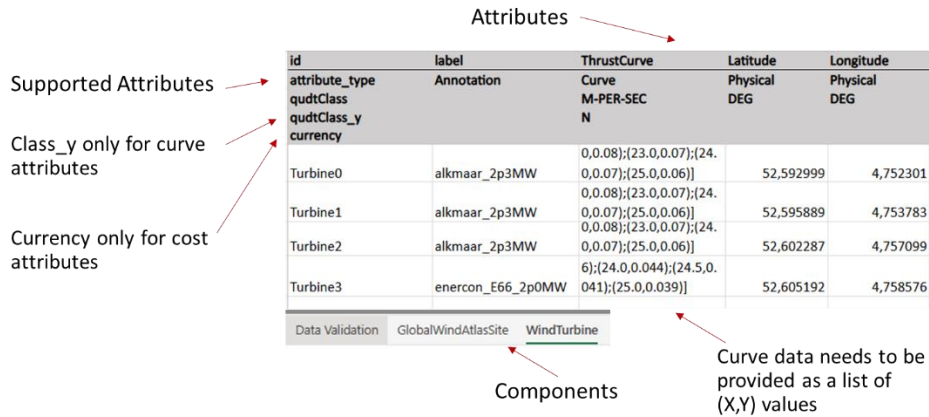


Figure 4: Specifying the classes and attributes of the use case in a Microsoft Excel spreadsheet

4.3.2. Physical Relations

The knowledge graph also contains the Component connections present in the physical system. This is handled by the *linksComponent* object property, which has several sub properties to allow more specific relations within the graph.

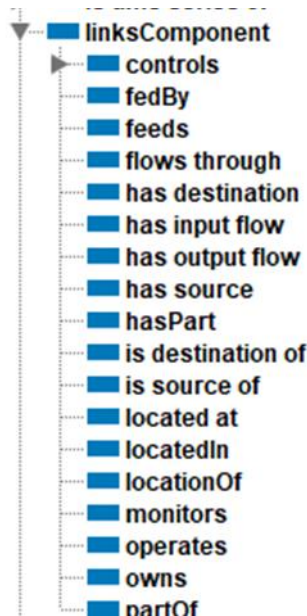


Figure 5: The linksComponent object property and sub properties used to link physically connected infrastructure within the knowledge graph

4.4. Creating Scenarios and Assumptions

The process of defining scenarios is assisted by the knowledge graph. It also allows new combinations of the system and attributes in the model. This process describes how the ontology provides a framework for scenario definitions combining the component and attributes.

4.4.1. Scenarios

The Scenario class enables modelling alternative system configurations, with the *derivedFrom* property establishing hierarchical relationships between scenarios. This mechanism supports incremental scenario development where derivative scenarios inherit base scenario characteristics while introducing specific modifications.

Component inclusion in scenarios operates through the *usedInScenario* property, which applies to components, attributes, and component links. This selective activation mechanism enables different system configurations within unified knowledge bases, supporting comparative analysis of alternative energy system designs or operational strategies.

The **ComponentLink** class represents relationships between components that may vary across scenarios. Properties *hasInputEntity* and *linksInputEntityTo* define component connections with scenario-specific applicability, enabling representation of different system topologies, operational patterns, or infrastructure configurations.

In addition to different configurations of physical assets, the user may wish to investigate the impact of adding new components from a repository of data products. The process of configuring scenarios from entities within the knowledge graph and ingested from data products, is shown in Figure 6.

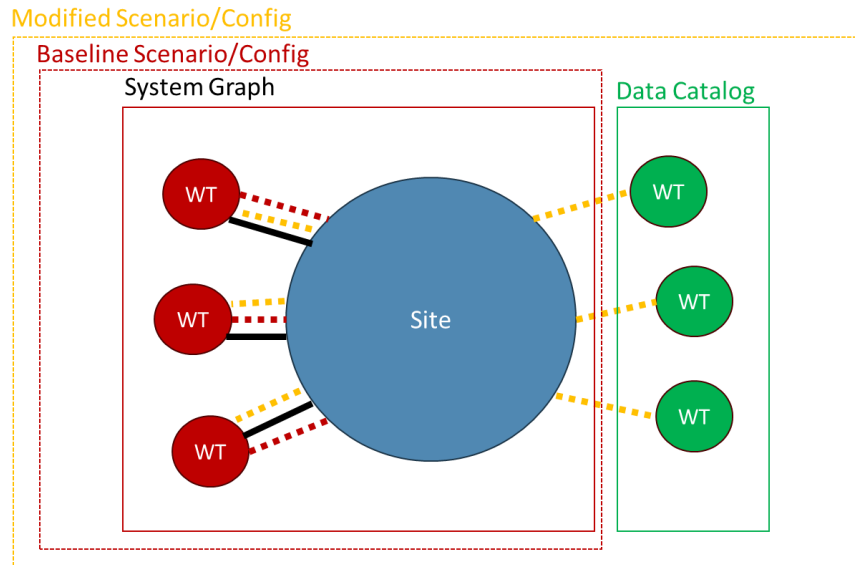


Figure 6: The process of defining new scenarios. The solid lines indicate a physical relationship within the knowledge graph. The dotted lines represent scenario configurations. WT = Wind Turbine.

4.4.1. Assumptions

The assumption framework implements systematic scenario parameterization through the **Assumption** class and its subclasses. The properties *targetComponent*, *targetAttribute*, and *modifier* enable precise specification of parameter modifications, supporting scenario definition through assumption sets rather than complete model duplication.

5. USECASES

A summary of the use cases covered to demonstrate the application of the dataspace (semantic layer) with the REFORMER Digital Twin is shown in Table 1.

Table 1 – A summary of the use cases covered in the REFORMERS extension of the Digicities ontology

<i>WP5 Example Digital Twin Services for Blueprint</i>			
Service Name	Description	Components: Attributes	Scenarios
Wind Forecasting	Predicts the output of wind turbines at a given wind atlas site	GlobalWindAtlasSite: Roughness WindTurbine: RatedPower, HubHeight, RotorDiameter, PowerCurve, ThrustCurve, Latitude, Longitude, Elevation	Baseline
Energy Simulation	Provides input parameters to the CESAR-P urban energy simulation tool	Location: WeatherEPW Building: BuildingType, YearOfConstruction, GroundFloorArea, NumberOfFloors, SpaceHeatingSupply, PrimaryHotWaterSupply	Baseline

5.1. Wind Forecasting

The steps for the accommodating the Wind Forecasting use case are detailed below.

5.1.1. Ontology Extension

The Components and Attributes included in the extension of the Digicities core ontology for the **WindForecasting** use case are shown in Table 2. The table shows the required

components and whether they are existing or additions to the core. Whenever the core is extended, the user must always specify the parent class. In this case, **GlobalWindAtlas Site** is a subclass of **Location** and **WindTurbine** is a subclass of **Turbine**. The table also shows the attributes and indicates existing or additions. The attributes tagged as "New Link" show attributes that were in the ontology but not previously connected to that **Component**. Mappings are an important consideration for interoperability within a dataspace; therefore, we also show which mappings have been included to standard data models. This process is detailed in Section 4.1.7. In this case, the instance of **WindTurbine** is considered equivalent to class **WindGeneratingUnit** of the CIM data model⁴.

Table 2: Extension of the Digicities core ontology to accommodate the Wind Forecasting service.

<i>Digicities core extension for WindForecasting</i>			
Components (Existing/ <u>New</u>)	Parent Class	Attributes (Existing/ <u>New</u> / <u>New Link</u>)	Mappings
<u>Global Wind Atlas Site</u>	Location	<i>Roughness</i>	
<u>Wind Turbine</u>	Turbine	<i><u>RatedPower</u>, <u>HubHeight</u>, <u>RotorDiameter</u>, <u>PowerCurve</u>, <u>ThrustCurve</u>, <i>Latitude</i>, <i>Longitude</i>, <i>Elevation</i></i>	≡ cim:Wind GeneratingUnit

5.1.2. System Graph

The system graph was created to include the existing turbines present at the Alkmaar wind farm. The attributes were collected using the standard template, shown in Figure 4. The **Wind Turbines** were linked to the Alkmaar **Global Wind Atlas** site using the *locatedIn* predicate. A visual graph of this relation is shown in Figure 7.

⁴ <http://iec.ch/TC57/2013/CIM-schema-cim16#>

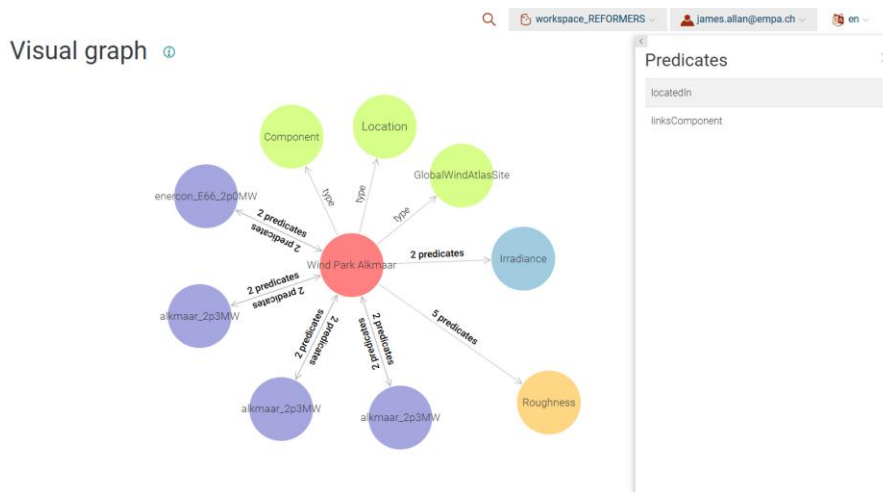


Figure 7: System graph showing the connection of the wind turbines present at the Alkmaar Wind Park.

6. CONCLUSION

This report presents an extension of the Digidities ontology for handling the data requirements of the services implemented in the REFORMERS Digital Twin. The proposed ontological framework demonstrates the capacity to handle any combination of component and attribute pairs and annotate associated temporal data (historic, real-time and future projections). Furthermore, the ontology accommodates the creation of scenario-based analytical models, specifically configured and validated to meet the requirements of the different platform services. The ontology defines component and attribute types and their interconnections. A standardised data collection template was developed and applied across the demonstration use cases, to ensure consistency and completeness in data acquisition. In addition, there have been procedures outline to map to data standards such as CGMES to assist interoperability. The REFORMERS ontology will be further developed to accommodate the models and services of the REFORMERS DT and will be published alongside the DT Blueprint.

7. ACKNOWLEDGEMENTS

Supporting the use cases contributed to the ontology development in the Digidities project which is supported by the Swiss Federal Office of Energy's P&D Program through the framework of the joint programming initiative ERA-Net Smart Energy Systems' focus initiative Digital Transformation for the Energy Transition, with support from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883973.

Thanks also to Edrisi Muñoz and Sergio Acero González who contributed to ontology conceptualisation and development.

The generation of text in the report was assisted by AI tools Grammarly and Claude.ai for readability and structure.

8. REFERENCES

- [1] Christian Mader, Jaroslav Pullmann, Niklas Petersen, Steffen Lohmann, Christoph Lange-Bever. International Data Spaces Information Model 2022.
- [2] Turkmayali A. Semantic interoperability: a common language for data sharing. International Data Spaces 2023. <https://internationaldataspaces.org/semantic-interoperability-a-common-language-for-data-sharing/> (accessed October 10, 2025).
- [3] Chen W, Das M, Gan VJ, Cheng JC. Integrated data model and mapping for interoperable information exchange between BIM and energy simulation tools, Springer; 2020, p. 496–506.
- [4] Fierro G, Pauwels P. Survey of metadata schemas for data driven smart buildings (Annex 81) 2022.
- [5] Luo N, Pritoni M, Hong T. An overview of data tools for representing and managing building information and performance data. Renewable and Sustainable Energy Reviews 2021;147:111224.
- [6] Mercado A, Mitchell R, Earni S, Diamond R, Alschuler E. Enabling interoperability through a common language for building performance data. Proceedings of the 2014 ACEEE Summer Study on Energy Efficiency in Buildings 2014.
- [7] Steinert A, Ferez S, Niese A. Choosing the Right Ontology to Describe Research Data in the Energy Domain. ACM SIGENERGY Energy Informatics Review 2025;4:33–48.
- [8] Aheleroff S, Xu X, Zhong RY, Lu Y. Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model. Advanced Engineering Informatics 2021;47:101225. <https://doi.org/10.1016/j.aei.2020.101225>.
- [9] Yehong L, Garcia-Castro R, Mihindikulasooriya N, O'Donnell J, Vega-Sánchez S. Enhancing energy management at district and building levels via an EM-KPI ontology. Automation in Construction 2019;99:152–67. <https://doi.org/10.1016/j.autcon.2018.12.010>.
- [10] Cuenca J, Larrinaga F, Curry E. DABGEO: A reusable and usable global energy ontology for the energy domain. Journal of Web Semantics 2020;61–62:100550. <https://doi.org/10.1016/j.websem.2020.100550>.

9. ANNEX

9.1. Example Attribute Instances

This section shows example instances of attributes that conform to the core ontology.

Physical Attributes

Physical attributes represent quantitative measurements with standardized units. This example shows a temperature measurement from a sensor:

```
@prefix dici_onto: <https://digicities.info/ontology#> .
@prefix qudt: <http://qudt.org/schema/qudt/> .
@prefix unit: <http://qudt.org/vocab/unit/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/building/sensor_01/temperature>
  a dici_onto:Temperature ;
  a dici_onto:PhysicalAttribute ;
  a dici_onto:SensorAttribute ;
  qudt:unit unit:DEG_C ;
  qudt:value "22.5"^^xsd:decimal .
```

Simple Cost Attribute

Simple cost attributes represent fixed monetary values without unit-based pricing structures:

```
@prefix iso4217: <http://example.org/currency/> .

<http://example.org/project/boiler_01/installation_cost>
  a dici_onto:InstallationCost ;
  a dici_onto:SimpleCostAttribute ;
  a dici_onto:ConverterAttribute ;
  qudt:value "45000.0"^^xsd:decimal ;
  dici_onto:currency iso4217:EUR .
```

Unit-Based Cost Attribute

Unit-based cost attributes associate costs with specific measurement units, such as energy pricing:

```
<http://example.org/grid/electricity_tariff>
  a dici_onto:ElectricityTariff ;
  a dici_onto:UnitBasedCostAttribute ;
  a dici_onto:EnergyCarrierAttribute ;
  qudt:value "0.28"^^xsd:decimal ;
  qudt:unit unit:KiloW-HR ;
```

```
dici_onto:currency iso4217:EUR .
```

Curve Attribute

Curve attributes represent functional relationships between two variables with defined coordinate systems:

```
<http://example.org/heatpump_01/performance_curve> a
dici_onto:PerformanceCurve ; a dici_onto:CurveAttribute ; a
dici_onto:EnergyConverterAttribute ; dici_onto:xUnit
unit:DEG_C ; dici_onto:yUnit unit:PERCENT ;
dici_onto:hasDataPoints ""[ [-10.0, 85.0], [-5.0, 90.0], [
0.0, 95.0], [ 5.0, 100.0], [ 10.0, 103.0], [ 15.0, 105.0]
]"" .
```

Curve Attribute

Categorical attributes represent discrete states or classifications through named individuals:

```
# Attribute instance with categorical type
<http://example.org/building_01/construction_type>
  a dici_onto:ConstructionType ;
  a dici_onto:CategoricalAttribute ;
  a dici_onto:LocationAttribute ;
  a dici_onto:Concrete .

# Named individual representing a category
dici_onto:Concrete
  a owl:NamedIndividual ;
  a dici_onto:ConstructionType ;
  rdfs:label "Concrete Construction" .
```

Geospatial Attribute

Geospatial attributes capture location-based properties with coordinate systems:

```
<http://example.org/building_01/coordinates>
  a dici_onto:GeographicCoordinates ;
  a dici_onto:GeospatialAttribute ;
  a dici_onto:LocationAttribute ;
  qudt:unit unit:DEG ;
  dici_onto:latitude "47.3769"^^xsd:decimal ;
  dici_onto:longitude "8.5417"^^xsd:decimal ;
  dici_onto:elevation "408.0"^^xsd:decimal .
```

Custom Physical Ratio Attribute



Custom physical ratio attributes represent derived quantities with distinct numerator and denominator units that do not exist as standardized units within QUDT. This example shows carbon intensity measured as kilograms of CO₂ equivalent per megawatt-hour of energy produced:

```
<http://example.org/generator_01/carbon_intensity>
  a dici_onto:CarbonIntensity ;
  a dici_onto:CustomPhysicalRatioAttribute ;
  a dici_onto:EnergyGeneratorAttribute ;
  qudt:value "0.385"^^xsd:decimal ;
  dici_onto:numeratorUnit unit:KiloGM-PER-M3 ;
  dici_onto:denominatorUnit unit:MegaW-HR ;
  rdfs:comment "Carbon dioxide emissions per unit energy
generation" .
```

Event Attribute

Event attributes capture temporal occurrences with specified precision levels:

```
<http://example.org/building_01/construction_date>
  a dici_onto:ConstructionDate ;
  a dici_onto:EventAttribute ;
  a dici_onto:LocationAttribute ;
  dici_onto:hasTemporalValue "1957"^^xsd:gYear ;
  dici_onto:hasTemporalPrecision dici_onto:Year .

<http://example.org/system_01/commissioning_timestamp>
  a dici_onto:CommissioningTimestamp ;
  a dici_onto:EventAttribute ;
  a dici_onto:ProcessAttribute ;
  dici_onto:hasTemporalValue "2023-06-
15T14:30:00"^^xsd:dateTime ;
  dici_onto:hasTemporalPrecision dici_onto:DateTime .
```

Simple Value Attribute

Simple value attributes represent unitless or dimensionless quantities:

```
<http://example.org/system_01/occupancy_count>
  a dici_onto:OccupancyCount ;
  a dici_onto:SimpleValueAttribute ;
  a dici_onto:LocationAttribute ;
  qudt:value "150"^^xsd:integer ;
  rdfs:comment "Number of building occupants" .
```

Dynamic Attribute with Time Series



Dynamic attributes link to time series data representing temporal variations:

```
# Dynamic attribute instance
<http://example.org/building_01/electricity_demand>
  a dicit_onto:ElectricityDemand ;
  a dicit_onto:PhysicalAttribute ;
  a dicit_onto:DynamicAttribute ;
  a dicit_onto:EnergyConsumerAttribute ;
  dicit_onto:hasHistoricTimeSeries
<http://example.org/building_01/electricity_demand/historic/
ts> ;
  dicit_onto:hasHistoricTimeSeriesReference
"historical_demand_2023.csv"^^xsd:string ;
  dicit_onto:hasLiveTimeSeries
<http://example.org/building_01/electricity_demand/live/ts> ;
  dicit_onto:hasLiveTimeSeriesReference
"mqtt://broker.example.org/building_01/demand"^^xsd:string .

# Historic time series instance
<http://example.org/building_01/electricity_demand/historic/
ts>
  a dicit_onto:HistoricTimeSeries ;
  dicit_onto:storedAt
"/data/timeseries/historical_demand_2023.csv"^^xsd:string ;
  dicit_onto:hasFileName
"historical_demand_2023.csv"^^xsd:string ;
  qudt:unit unit:KiloW ;
  dicit_onto:startTime "2023-01-01T00:00:00"^^xsd:dateTime ;
  dicit_onto:endTime "2023-12-31T23:45:00"^^xsd:dateTime ;
  dicit_onto:temporalResolution "PT15M"^^xsd:duration .
```

Resource Attribute

Resource attributes reference external data sources:

```
<http://example.org/weather_station_01/climate_data>
  a dicit_onto:ClimateData ;
  a dicit_onto:ResourceAttribute ;
  a dicit_onto:LocationAttribute ;
  dicit_onto:hasDataPath
"/data/weather/zurich_climate_2023.epw"^^xsd:string ;
  rdfs:comment "EnergyPlus weather file for location" .
```

Identifier Attribute

Identifier attributes capture external identification schemes:



```
# Component with identifier reference
<http://example.org/building_01>
  a dici_onto:Building ;
  dici_onto:hasIdentifier
<http://example.org/building_01/egid> .

# Identifier instance
<http://example.org/building_01/egid>
  a dici_onto:BuildingIdentifier ;
  a dici_onto:IdentifierAttribute ;
  dici_onto:identifierValue "123456789"^^xsd:string ;
  rdfs:label "Swiss Federal Building Identifier (EGID)" .
```