FI.ICT-2011.1.8 FINESCE

D7.13 V1.0

Mapping of FINESCE Trial Sites' Data Models onto a Unified Data Model for the Smart Energy Platform

Contractual Date of Delivery to the CEC:	*not applicable, extra deliverable*
Actual Date of Delivery to the CEC:	
Author(s):	Gianluca Lipari
Participant(s):	*RWTH Aachen*
Workpackage:	*WP7– FINESCE Project Management*
Estimated person months:	*1.0*
Security:	**PU**
Nature:	**R** = Report
Version:	1.0
Total number of pages:	**30**

Abstract: This report is about the unification and consolidation of the trial site specific data models onto a single data model for the Smart Energy Platform.

Keyword list:

FIWARE, Smart Energy, Platform, Data Model

Disclaimer:

All information provided reflects the current status of the trial site testbeds at the time of writing and may be subject to change.

Executive Summary

In the FINESCE project there are seven trials, where each partner has developed a trial site specific data model for data acquisition, storage and processing. However, the main objective of FINESCE is to promote a single Smart Energy Platform (SEP) and to reach this goal it is necessary to develop a unified data model for the entire platform, thus making possible data exchange between all the involved partners and actors.

This report presents this consolidation work on data models from all the FINESCE trial site systems (as actually implemented). The underlying theme of the entire proposed work is the design of a unified data model, the Smart Energy Model (SEM), which is based on the data models of the FINESCE trial sites. It enables interoperability between all the trial sites as well as laying the foundations for future development of the SEP, in a perspective of increasing integration of actors and devices, brought by the possible future new use cases, into the FINESCE ecosystem.

Authors

Partner	Name	e-mail
RWTH	Gianluca Lipari	GLipari@eonerc.rwth-aachen.de

Table of Contents

1. Introduction

The FINESCE project provides, via the FINESCE API, open access to live and historical data from real grids and buildings accessed by the FINESCE Smart Energy use cases. Physical infrastructures, the FINESCE Trial Sites, in different European countries are used to implement the use cases. These use cases were chosen in areas expected to gain the most from the use of ICT.

The aim of the present work is to present a Smart Energy Model (SEM) for the FINESCE Smart Energy Platform (SEP). This model is meant to integrate the existing different data models developed for the trial sites and also to be able to cope with other data models, being proprietary or standardised models, in order to make them compatible with the Platform.

In Chapter 2 we will provide an introduction to the main concepts behind the design of data models and data ontologies, introducing the main concepts and the specific terminology.

In Chapter 3 we will present the modelling approach used for the design of the SEM, from the identification of the semantics and the conceptual model to the definition of the meta-model from which the proposed unified data model has been derived.

The meta-model, which definition has been the starting point for our work, is described in Chapter 4, specifying the guidelines and the design rules applied for its definition and introducing the needed classes and their relationships. The goal of this task was to identify the most suitable data structure and the right design rules to ensure the highest compatibility and extensibility for the SEM.

After the definition of the meta-model we were able apply it to the FINESCE use cases, thus unifying the trial sites specific data model onto the SEM and then, in Chapter 5, the actual classes, derived from those used in each trial site and designed applying the unified data model, are presented. We adopted a use-case-based approach, making a unified data model for only those entities needed for the actual use cases and then adding more use cases as further functionality is later added to SEP.

The work presented here applies the SEM concept to the data accessible via the FINESCE API, mapping the heterogeneous data models used by the different trials to the unified SEM data model.

2. Ontology Engineering and Data Modelling

A computer ontology is said to be an “agreement about a shared, formal, explicit and partial account of a conceptualisation” [2,3,4]. In addition an ontology contains the vocabulary (terms or labels) and the definition of the concepts and their relationships for a given domain. In many cases, the instances of the application (domain) are included in the ontology as well as domain rules that are implied by the intended meanings of the concepts. Domain rules restrict the semantics of concepts and conceptual relationships in a specific conceptualisation of a particular application domain. These rules must be satisfied by all applications that want to use an ontology [1].

A data model represents the structure and integrity of the data elements of the specific applications by which it will be used. Therefore, the conceptualisation and the vocabulary of a data model are not intended a priori to be shared by other applications [7].

A meta-model is an explicit model of the constructs and rules needed to build specific models within a domain of interest. This characterizes a valid meta-model as an ontology, since such constructs and rules represent entities in a domain and their relationships, i.e., a set of building blocks used to build domain models. In other words, a meta-model is an ontology used by modellers. For example, when software developers use UML to construct models of software systems, they actually use an ontology implemented in it. This ontology defines concepts such

as objects, classes, and relations. However, not all ontologies are modelled explicitly as meta-models [13].
Modelling ontologies for a wide usage in an open environment, such as the Semantic Web, obviously is a challenging task. Providing more ontology rules, which are important for effective and meaningful interoperation between applications, may limit the generality of an ontology. However, light ontologies, i.e. holding none or few domain rules, are not very effective for communication between autonomous software agents.

Data models, such as database or XML-schemes, typically specify the structure and integrity of data sets. Thus, building data models usually depends on the specific needs and tasks that have to be performed. In the context of open environments (as is the Semantic Web), ontologies represent knowledge that formally specifies agreed logical theories for an application domain [2]. Ontological theories, i.e. a set of formulas intended to be always true according to a certain conceptualisation [8], consist of domain rules that specify – or more precisely, approximate – the intended meaning of a conceptualisation. Ontologies and data models, both being partial accounts (albeit in a varying degree) of conceptualisations [1], must consider the structure and the rules of the domain that one needs to model. But, unlike task-specific and implementation-oriented data models, ontologies, in principle and by definition, should be as much generic and task-independent as possible. The more an ontology approximates the ideal of being a formal, agreed and shared resource, the more shareable and reusable it becomes. In fact, reusability and reliability are system engineering benefits that derive from the use of ontologies [9].

In 1975 ANSI [5, 6] described three kinds of data-model instance:

- Conceptual schema: describes the semantics of a domain (the scope of the model). For example, it may be a model of the interest area of an organization or of an industry. This consists of entity classes, representing kinds of things of significance in the domain, and relationships assertions about associations between pairs of entity classes. A conceptual schema specifies the kinds of facts or propositions that can be expressed using the model. In that sense, it defines the allowed expressions in an artificial "language" with a scope that is limited by the scope of the model.
- Logical schema: describes the structure of some domain of information. This consists of descriptions of (for example) tables, columns, object-oriented classes, and XML tags.
- Physical schema: describes the physical means used to store data. This is concerned with partitions, CPUs, tablespaces, and the like.

Since an ontology is a model of a domain describing objects that inhabit it, all three types of data models can be thought of as ontologies. They range from the most expressive one that describes business concepts and processes (the conceptual model) to less expressive and progressively moving from describing business semantics to describing physical structures of the data as it is stored in the databases (the logical and physical data model). The physical model can be thought of as an ontology of a particular database.

Usually, early phases of many software-development projects emphasize the design of a conceptual data model. Such a design can be detailed into a logical data model. In later stages, this model may be translated into physical data model. However, it is also possible to implement a conceptual model directly.

A conceptual schema or conceptual data model is a map of concepts and their relationships used for databases. This describes the semantics of an organization and represents a series of assertions about its nature. Specifically, it describes the things of significance to an organization (entity classes), about which it is inclined to collect information, and characteristics of (attributes) and associations between pairs of those things of significance (relationships).

Finally, the model does allow for what is called inheritance in object oriented terms. The set of instances of an entity class may be subdivided into entity classes in their own right. Thus, each instance of a sub-type entity class is also an instance of the entity class's super-type. Each instance of the super-type entity class then is also an instance of one of the sub-type entity classes.

Super-type/sub-type relationships may be exclusive or not. A methodology may require that each instance of a super-type may only be an instance of one sub-type. Similarly, a super-

type/sub-type relationship may be exhaustive or not. It is exhaustive if the methodology requires that each instance of a super-type must be an instance of a sub-type.

In Model Driven Architecture a three-tier infrastructure architecture is usually considered. The infrastructure consists of three levels – meta-model level (M2), model level (M1), and object level (M0). M2 defines the meta-model specifying the abstract syntax for models at M1 whose elements are instances (classes) of the meta-classes at M2. The model at M1 captures the abstraction of object models at M0 whose elements are instances (objects) of the classes M1 [14]. The proposed unified data model, described in Chapter 5, resides on M1 level, where the actual data models used in the FINESCE trials are defined.

However, the first step of our modelling work, presented in Chapter 3, has been the identification of involved entities and their relationships, the rules and the connections between the main classes. Since this first step took place on the M2 level of abstraction its result has been the definition of the meta-model, together with a general upper ontology and the related semantics.

3. Modelling Approach

There are a lot of data models available that either follow a different data format or are designed for specific purposes. Most of them either lack the possibility to be extensible since they are either designed for a specific purpose or are vendor specific and proprietary.

The main problem to be addressed is interoperability between different models and current solutions mainly rely on the specification of a common data exchange format.

The modelling approach adopted for our work is based on the results of the COOPERaTE project1, particularly the Neighbourhood Information Model (NIM) [11], which is the result of a comparative study of different Smart City data modelling approaches. One of the first tasks performed in the COOPERaTE project was the definition of the NIM on the semantic/ontology level. So, following the same path, we decided to start from the definition of the needed ontology, the model of the general Smart Energy domain, and then to the realisation of the conceptual meta-model from the platform on a high level of abstraction.

We will present our meta-model, based on NIM, in the next chapter. The meta-model describes the different parts of the SEP in a higher level of abstraction, so it doesn't depend on the use cases or the trial sites specification.

The reason why we do not present a concrete data model directly is that there are many data entries which must be covered by SEM; this makes the developed SEM huge and difficult to manage. Furthermore we assume that the developed SEM should be extensible and adaptable because of either new requirements for data entries arising in the future or new devices with a different existing information model joining the platform.

The two main concepts inherited from the NIM, that we followed for the design of the meta-model are extensibility and the integration of heterogeneous information models [10].

Thanks to the generic approach the SEM makes possible to easily specify new entries, categories and associations, thus making the data model extensible and able to integrate new devices into the Platform.

The second concept is the integration of heterogeneous information models. For the SEM it is important to be able to read data from different devices and to integrate them into the unified data model. This can be realised by a defined mapping from the specific device information model to the SEM.

Before presenting the proposed unified model in Chapter 5, the used modelling notation and the proposed meta-model will be shortly presented in Chapter 4. For specifying the data model we use UML class diagrams (CD) [12]. Such CDs are used to model the structure of a software system and are thus suitable for specifying the structure of the SEM data. CDs consist of

[1] http://www.cooperate-fp7.eu/

classes, interfaces and associations. Each class consists of a name, available fields and methods including visibility, type and return type. In addition to this, classes can have subclasses or implement interfaces. Interfaces contain methods that each implementing class has to offer. Associations specify a connection between two classes and may have an optional name, optional cardinalities at both ends of the association and optional role names at both ends of the association. The direction of the arrow displays accessibility between the two classes.

4. Meta-Model

A meta-model is a model used to describe other models. This meta-model is a scheme that specifies the guidelines and the design rules, in terms of classes' names and specifications and their relationships and dependencies, for the design of the unified data model, presented in Chapter 5. Therefore the meta-model constitutes the foundation, fixed and not dependant from the actual use cases, of the derived unified model.
First of all we introduce the proposed meta-model and the rules and definitions derived from it.

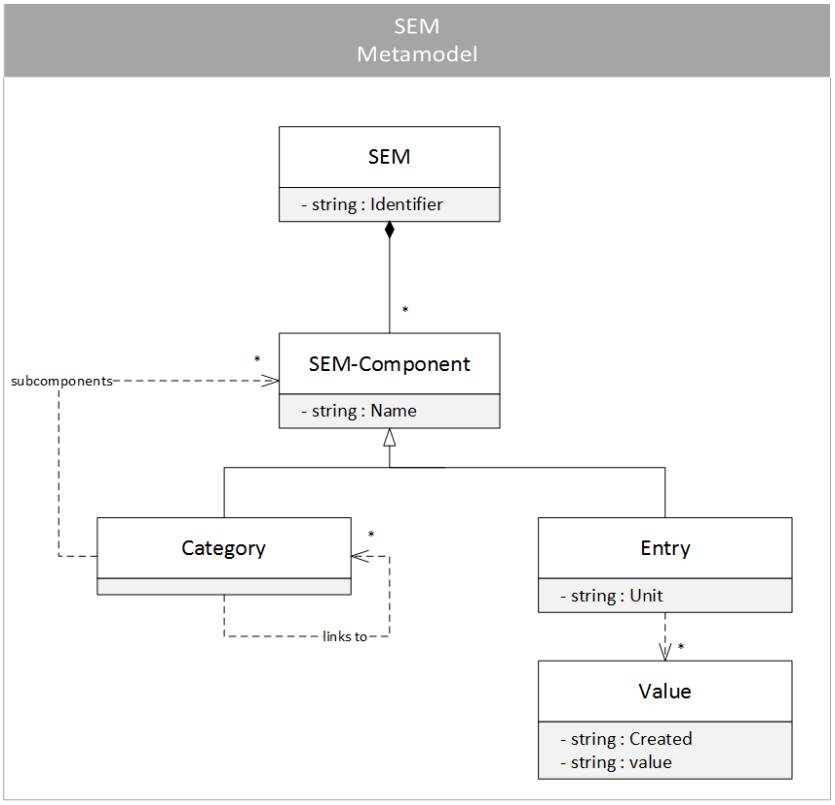

Fig. 1 Meta-model for the Smart Energy Domain

Our meta-model for the Smart Energy domain, based on the NIM meta-model, is depicted in Fig. 1. Every box in Fig. 1 is a class. Generally, a class is composed from structural and behavioural constituents. The structural constituents are the class name (in our case SEM, SEM-Component, Category, Entry and Value) and the class attributes (e.g. in the Value class there are two attributes of type "String" named "Created" and "Value"). The behavioural constituents are called "Methods", but we haven't specified any method in our meta-model for the moment.

Following the approach used in the NIM we have included the following five classes in the SEM:

- **SEM**: identifies an actual real world object (e.g. a house or a vehicle) by its Identifier;
- **SEM-Component**: identifies the general component of the SEM. This class is a generalisation of the two subclasses "Category" and "Entry";
- **Category**: a subclass of the SEM-Component used to structure the data of SEM;

- **Entry**: specialisation of the SEM-Component class, used to contain the data values in the related Value class, on which it depends;
- **Value**: class that contains the actual data value and a timestamp that specifies when the sample was acquired.

Further on we take a closer look on the data entries of the SEM. For each entry different kinds of information can be stored:

- values
- value ranges
- forecast data
- historical data

Fig. 2 shows the underlying model for SEM entries where the Entry class of this figure corresponds to the Entry class of Figure 1. The different kinds of information which can be stored for an entry are modelled as separated classes: `Value`, `ValueRange` and `Forecast` which are described in detail below:

- The value attribute of a `Value` stores a single value that is set, measured, derived or calculated.
- With the help of the ValueRange of an entry, the allowed value range for values inside of Value can be stored. A min and a max value can be specified for this purpose.
- The Forecast of an entry consists of one or more Values. These values represent the result of any forecast algorithm or can be used to store external forecast values. The date when these values are computed or entered (in case of external values) is stored in the created field of the Forecast. As a single forecast can make statements for several different points in time, for each point in time a single Value is created which stores the predicted value as well as the predicted date and time of its occurrence in the timestamp field. In addition the validUntil timestamp specifies at which point in time the forecast gets invalid. Since there are several possible forecast sources we add an identifier to keep the information on the algorithm, external source or calculation that created the forecast.

The Entry class is connected to the classes representing the different kinds of information via a single association. This association can be used to either retrieve the actual data or the historical data. The actual data is found via the latest timestamp whereas historical data is represented by the values that have an earlier timestamp. Each part of a SEM entry which is associated to the Entry class is stored together with a timestamp, which represents the date and time when the specific information was entered. Moreover each Entry has a name and a unit. The name must be unique in the scope of the category which contains the entry. Thus, entries can be identified by their name and category. The unit field is an optional field and can be left empty when no unit needs to be specified for a specific entry.

For each of the different types of content (`Value`, `ValueRange` and `Forecast`) historical data is stored in addition to the current information. Thus, if an entry's content is updated, no information will be overwritten. When accessing historical values range queries can return a subset of the list of values or all stored historical values.

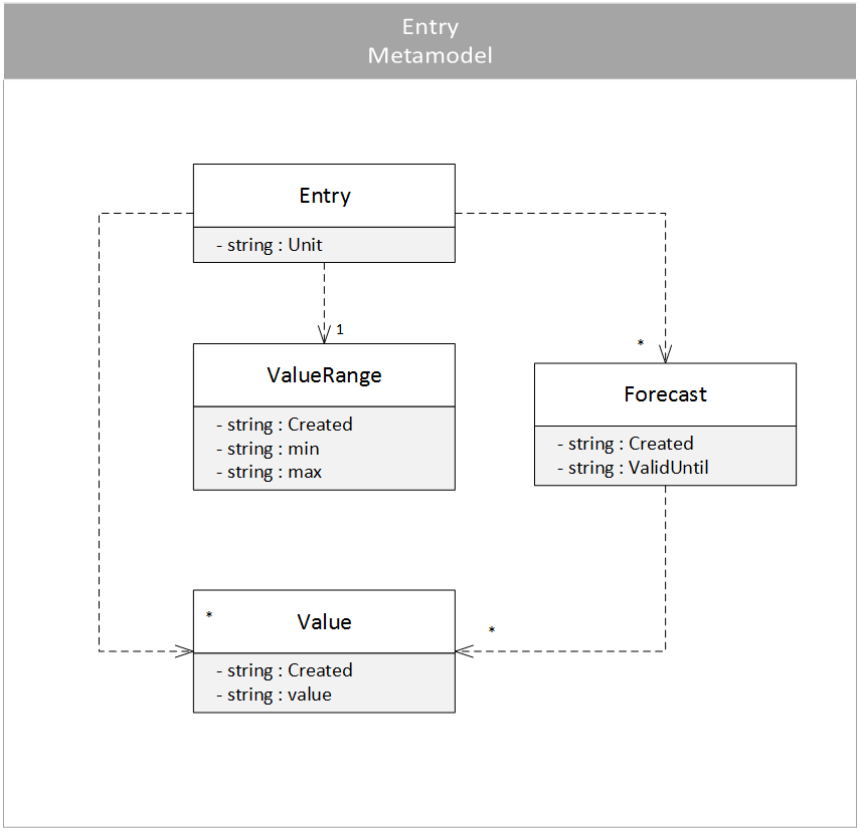

Fig. 2 Meta-model for the Entry Class

5. Classes in Unified Data Model

In this chapter we will present a set of classes in the unified data model, based on the data models of the FINESCE trials. This unified data model makes possible a common interpretation of the different heterogeneous data models used in the FINESCE use cases and allows the data to be stored in a common structure. We considered all the data models used by the different trials in FINESCE and made them compliant with the proposed unified data model, completing in this way the realization of the SEM. This approach guarantees the extensibility and reusability of the SEM, since it makes the data model flexible and able to integrate any kind of information or data that will be needed in the future.

As already specified in the previous chapter the data model follows the rules delineated in the meta-model. We will use the SEM-Component subclasses called “Category” and “Entry” to structure the data and the store them. The actual data values are stored in the Value class, on which every Entry class depends.

5.1 Building Class

This class provides information related to smart buildings available in the trial sites. Each Building has its own unique identifier (id) and a type (house, residential building or commercial building). The Building class also depends on three other classes: an Address class, containing the information about the physical location of the building, a RelatedEntity class, that contains a list of entities related to the specific building (e.g. an electric vehicle or a smart meter), and a MeasurementPoints class that specifies which measurement point are available in the building.

Name	Type	Description
id	String	The id of the building
type	String	The type of the building
measurementpoints	MeasurementPoints	The types of measurements that are associated with the specific building

relatedentities	RelatedEntities	A list of entities associated with the building
address	Address	The address of the building (see Table 2)

Table 1 Attributes of the Building class

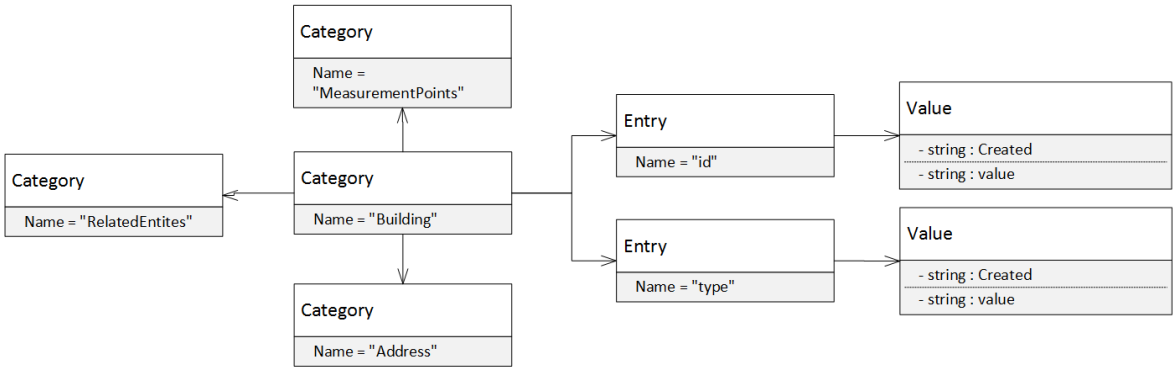

Fig. 3 Class Diagram for the Building class

5.2 Address Class

The Address class contains information about the physical location of an entity (device, building or event). It has six attributes, shown in Table 2.

Name	Type		Description
street_name	String		The street name of the building address
number	String		The number of the building address
city	String		The city of the building address
province	String		The province of the building address
zip	String		The zip code of the building address
country	String		The country of the building address

Table 2 Attributes of the Address class

Following the design rules explained before we will present the class diagram of the Address Class. The class itself is thus mapped into a Category, where the name attribute is set to “Address”. Each attribute is mapped onto an Entry, while the strings containing the information are stored in the associated value attribute.

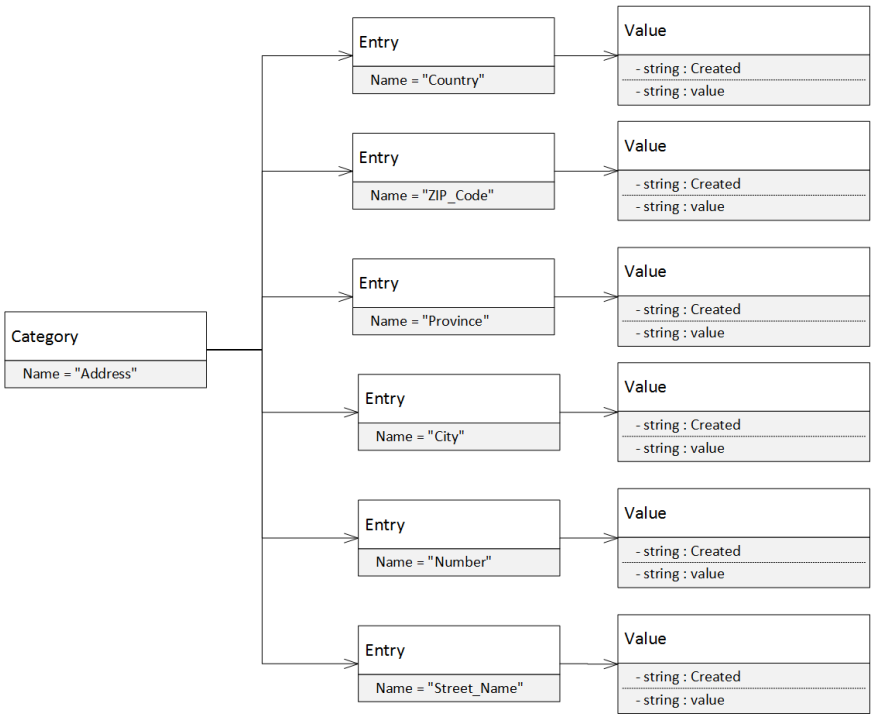

Fig. 4 Class Diagram for the Address class

5.3 RelatedEntities Class

This class is used to emphasize a connection that exists between two entities. It contains the id of the related entity (e.g. a device or a building) and the relationship between the two entities.

Name	Type	Description
id	String	The id of the related entity
relation	String	The relationship between the two entities

Table 3 Attributes of the RelatedEntities class

Below the class diagram of the RelatedEntities Class is depicted. The Category name is set to “RelatedEntites”, while the two attributes are mapped onto Entry-Value pairs, as required by the design rules presented before.

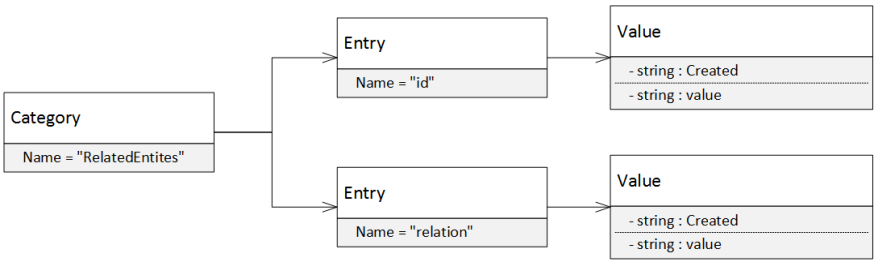

Fig. 5 Class Diagram for the RelatedEntities class

5.4 MeasurementPoint Class

This class is used to identify a measurement point located in a specific physical location (being it a device or a building). It contains the id assigned to the measurement point and the relative name, a brief description of the measurement and its type. There is also ant attribute called “unit”, that specifies the unit used for the measurement. Finally the actual data are stored in as

many Entry-Value pairs as needed, each reporting the actual value and the time when the measurement was carried out.

Name	Type	Description
id	String	The id of the Measurement Point
name	String	The name assigned to the Measurement Point
description	String	The description of the measurement
type	String	The type of measurement carried out
unit	String	The measurement unit used
data	String	The actual measured data

Table 4 Attributes of the MeasurementPoint class

Following is shown the relative class diagram. The multiplicity indicated for the “data” entry highlights how it is possible to associate an unlimited number of data values to a specific measurement point. In this way the measurement object which acts as a container for a number of measured values.

Based on the input gathered from the various FINESCE trials, a number of measurement type names have been compiled and are as follows:

BatteryCapacity, BatteryLevel, ChargingSocketPowerConsumption, ChargingSocketPowerDemand, CloudCover, ColdWaterTemperatureAverage, Distance, EnergyConsumption, EnergyConsumptionGrid, EnergyProduction, EnergyProductionGrid, EnergyProductionSolar, EnergyStatus, EnergyStatusSingleton, GlobalHorizontalIrradiance, HeatConsumption, HeatEnergyAverage, HeatpumpForwardFlowTemperatureAverage, HeatpumpHeatFlow, HeatpumpInletFlowDemand, HeatpumpInletFlowSummation, HeatpumpInletTemperature, HeatpumpOutletFlowDemand, HeatpumpOutletFlowSummation, HeatpumpOutletTemperature, HeatpumpPowerAverage, HeatpumpReturnFlowTemperatureAverage, HotWaterEnergyAverage, HotWaterFlow, HotWaterTemperatureAverage, ICMIndoorCO2LevelAverage, ICMIndoorHumidityAverage, ICMIndoorTemperatureAverage, Illuminance, IndoorTemperatureAverage, MaxTemperature, MinTemperature, ModuleStatus, NaturalGasConsumption, OutdoorTemperatureAverage, Position, PowerDemand, PowerDemandGrid, PowerMixed, PowerSupply, PowerSupplyGrid, RelativeHumidity, Speed, SunriseTime, SunsetTime, Temperature, WindDirection and WindSpeed.

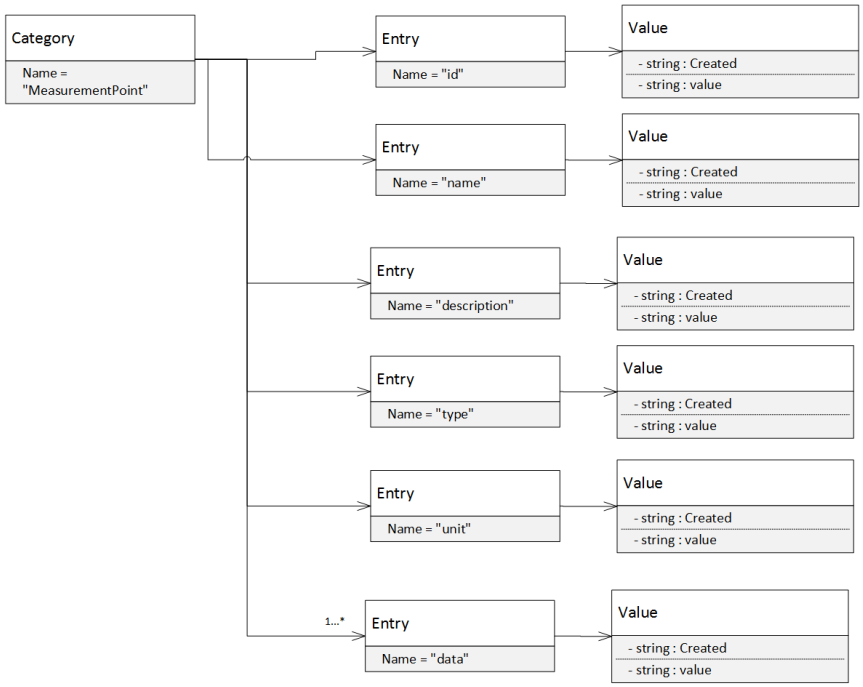

Fig. 6 Class Diagram for the MeasurementPoint class

5.5 Meter Class

This class contains data related to the presentation of the metering equipment of a Smart Energy site, be it a Smart Grid or a Smart Building. It contains information about the customer, the data acquisition, the physical location of the meter and its related entities. The sector attribute contains the classification of the sectors that the meters may be deployed into. The sectors have been defined in the Terni trial site and are: Commercial, Industrial, Lighting, Office and Secondary Substation.

Name	Type	Description
id	String	The id of the meter
address	Address	The address of the meter installation (see 5.2)
customer_id	String	The id of the customer associated with the meter
sector	String	The sector the meter (customer) belongs to
sample_identifier	String	An indicator of the metering frequency of the meter
latitude	String	The latitude of the meter installation point
longitude	String	The longitude of the meter installation point
measurementpoints	MeasurementPoint	A list of measurement types supported by the meter (see 5.4 for details on supported types)
related_entities	RelatedEntities	A list of entities related to the meter (e.g. a building)

Table 5 Attributes of the Meter class

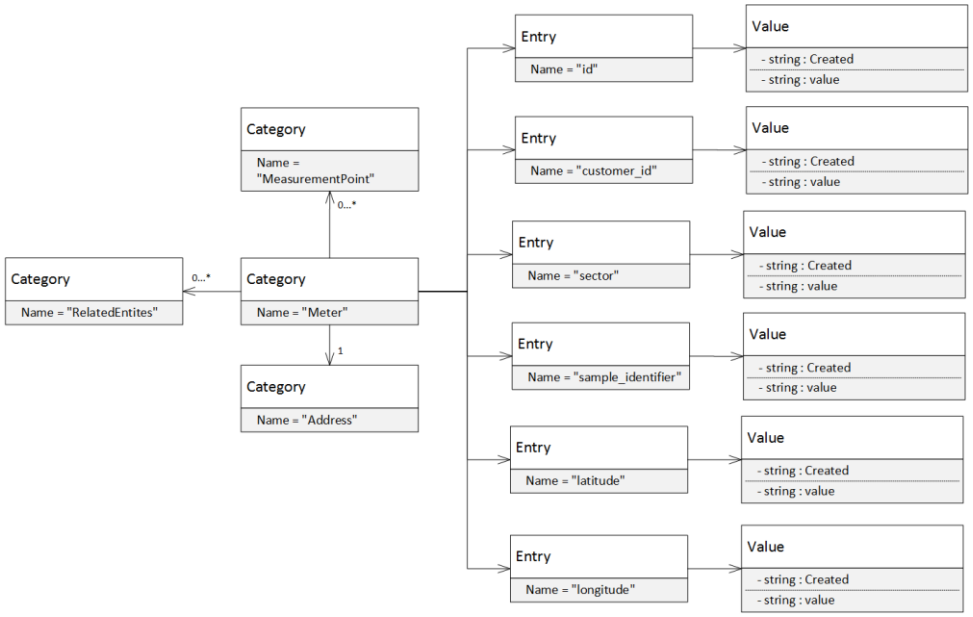

Fig. 7 Class Diagram for the Meter class

5.6 IssueResolutionPlan Class

This class is part of the demand/response and incentive plan services, offered by the Terni trial site, that are also related to contractual energy prices. The definitions of the attributes composing this class are reported in Table 6.

Name	Type	Description
id	String	The id of the issue resolution plan
time	String	The time, in ISO8601 compliant format, of the publication of the issue resolution plan
snapshot	String	The filename of an image object possibly accompanying the issue resolution plan
sector	String	The sector to which the issue resolution plan refers
state	String	The state of the issue resolution plan (submitted, accepted or rejected by the Distribution System Operator - DSO)
aggregator_notes	String	The (optional) notes of the Aggregator
DSO_notes	String	The (optional) notes of the DSO
author	String	The author of the issue resolution plan
author_email	String	The email of the author
actions	Action	The list of actions proposed to the DSO (see 5.9)

Table 6 Attributes of the IssueResolutionPlan class

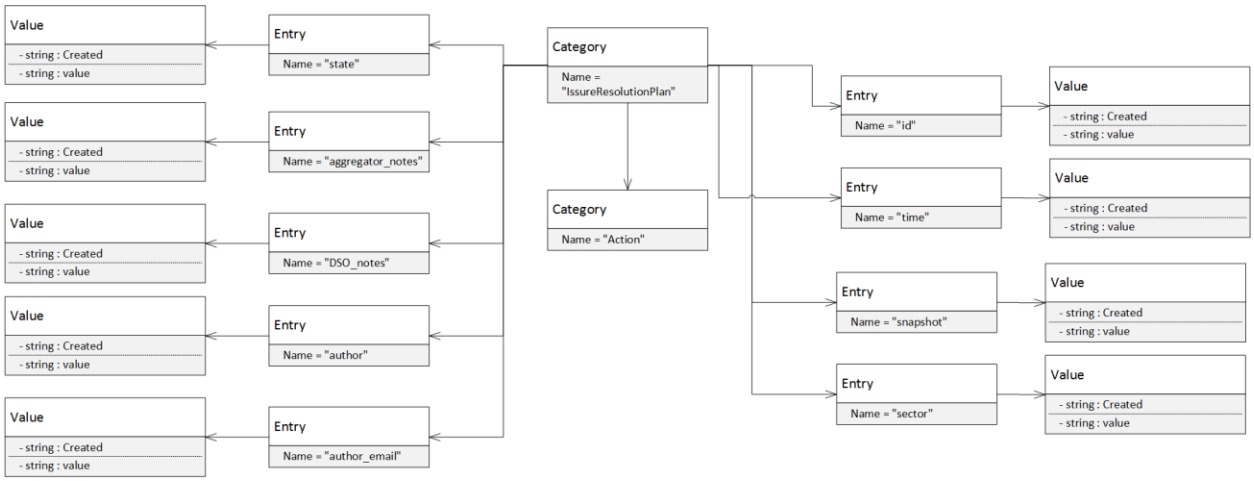

Fig. 8 Class Diagram for the IssueResolutionPlan class

5.7 IncentivePlan Class

This class is also part of the demand/response services and contains the proposed actions to be taken for the fulfilment of the desired behavior of the network.

The Terni trial site differentiates the actions between incentive plan actions, demand/response actions and contract actions. We decided to provide a single Action class that contains all the needed attributes for the original three classes.

Name	Type	Description
id	String	The id of the issue resolution plan
time	String	The time, in ISO8601 compliant format, of the publication of the issue resolution plan
customer_id	String	The id of the customer the incentive plan refers to
meter_id	String	The id of the metering device associated with the aforementioned customer
profile_name	String	The name of profile used
irp_id	String	The id of the issue resolution plan this incentive plan addresses
der_cost	String	The cost of the energy produced by

			renewable sources
primary_energy_cost	String		The cost of the energy produced by traditional methods, at very large scale (e.g. coal)
energy_cost	String		The final cost of the energy attained by the customer
state	String		The state of the incentive plan (Submitted, Compliant or Non-Compliant)
retailer_notes	String		The (optional) notes posed by the energy retailer upon publishing the incentive plan
regulator_notes	String		The (optional) notes posed by the market regulator upon evaluating the incentive plan
author	String		The author of the incentive plan
author_email			The email of the author
actions	Action		A list of actions proposed to the customer associated with the incentive plan in hand (see 5.9)

Table 7 Attributes of the IncentivePlan class

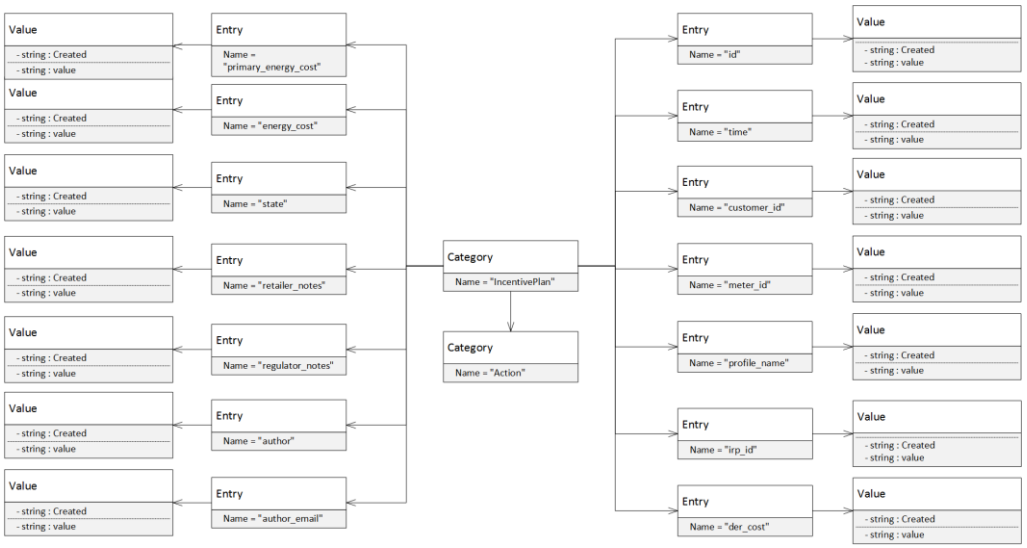

Fig. 9 Class Diagram for the IncentivePlan class

5.8 Contract Class

This class contains information about the contracts signed by the customers. It connects the contract to a incentive plan connected to the specific customer, which id is specified in the "customer_id" Entry. It also contains information about the state of the contract, if it has been just submitted to the customer, accepted or rejected. Finally there is a list of proposed actions.

Name	Type	Description
id	String	The id of the contract in hand
incentive_plan_id	String	The id of the incentive plan corresponding to the contract
customer_id	String	The id of the customer associated with the contract
state	String	The state of the contract
actions	Action	The list of actions proposed to the customer (see 5.9)

Table 8 Attributes of the Contract class

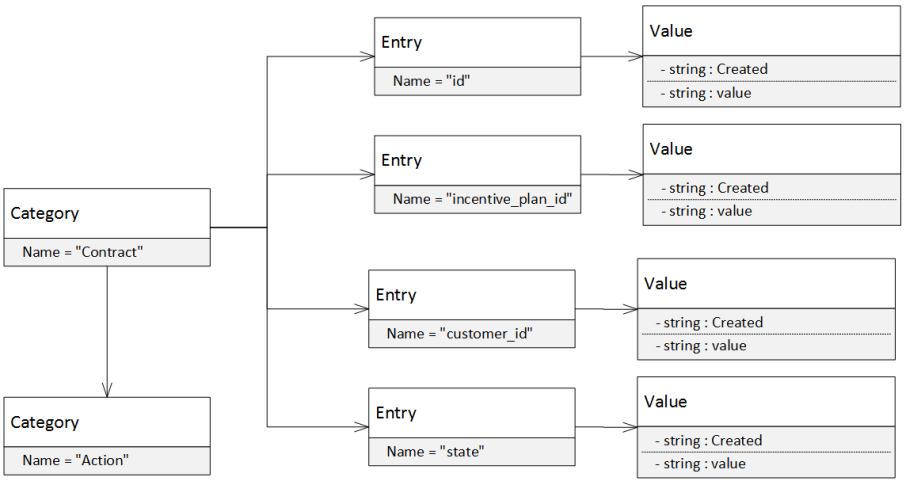

Fig. 10 Class Diagram for the Contract class

5.9 Action Class

This class is also part of the demand/response services and contains the proposed actions to be taken for the fulfilment of the desired behavior of the network.

The Terni trial site differentiates the actions between incentive plan actions, demand/response actions and contract actions. We decided to provide a single Action class that contains all the needed attributes for the original three classes, thus unifying them in a single definition. In fact the contract action class and the demand/response action class just extend the incentive plan action class, so they also comprise the attributes of this class as well. For this reason it is reasonable to unify them in a single, more general, Action class.

Name	Type	Description
target	String	The target of the issue resolution plan action
type	String	The type of the action to be performed (increase, decrease, shift)
value	String	The value of the action to be performed
start_time	String	The start time of the action, in an ISO8601 compliant format
end_time	String	The end time of the action, in an ISO8601 compliant format
offer	String	The offer (incentive) of the energy retailer to the customer
approval	String	Indicates when the customer decided (approved or rejected) the contract

Table 9 Attributes of the Action class

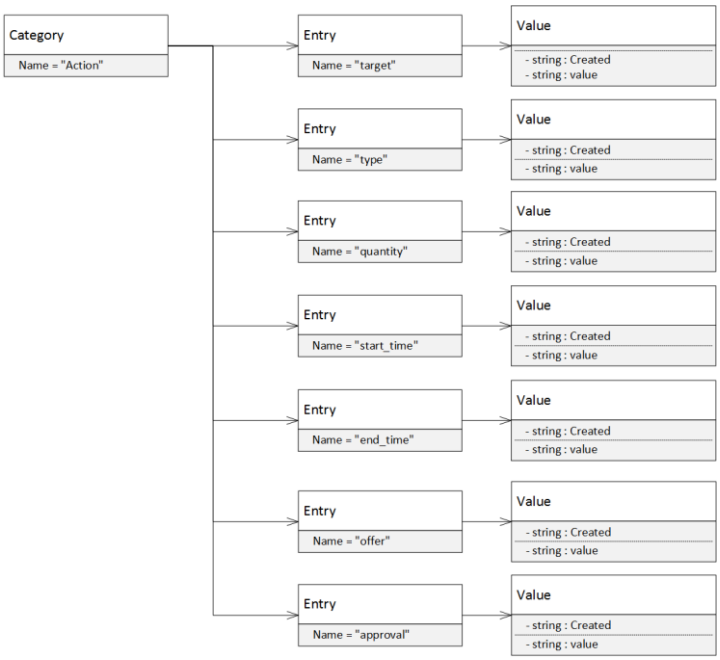

Fig. 11 Class Diagram for the Action class

5.10 EnergyCost Class

This class contains the contracted prices for electricity, based on the type (renewables or bulk sources).

Name	Type	Description
der	String	The cost of the energy coming from renewable energy sources
bulk	String	The cost of the energy coming from bulk generation sources (e.g. coal)
sold	String	The cost of the energy actually sold to the customers
volume	String	The metric of the energy (e.g. kWh)
currency	String	The currency of the energy sold (default is €)
start_date	String	The date when the documented prices are valid from, in an ISO8601 compliant format
end_date	String	The date until when the documented prices are valid, in an ISO8601 compliant format

Table 10 Attributes of the EnergyCost class

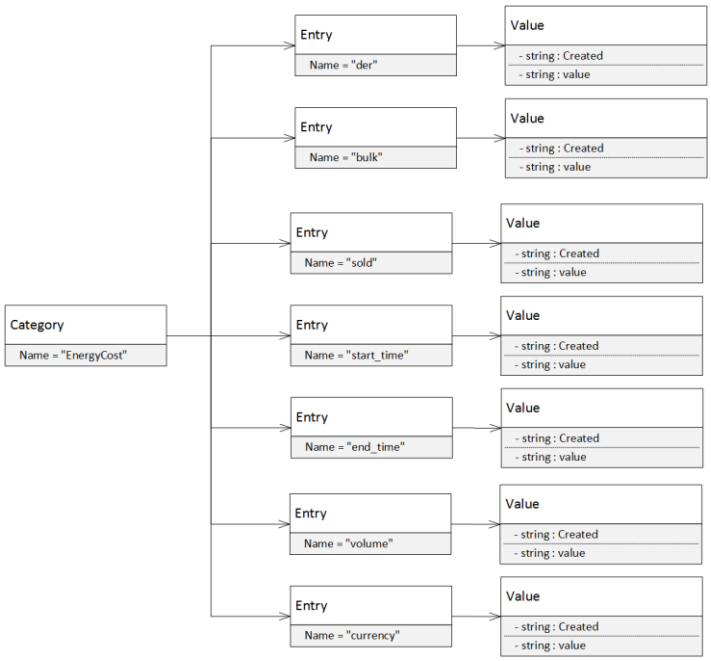

Fig. 12 Class Diagram for the EnergyCost class

5.11 Timeslot Class

This class contains the data needed to allow supporting discrete timeslots for describing an energy report instead using of continuous entities related to normal, continuous date/time presentation.

It contains the id of the timeslot, the start time and its duration.

Name	Type	Description
id	String	The id of the timeslot
start	String	The start time of the timeslot, in an ISO8601 compliant format
duration	String	The duration of the timeslot, in minutes

Table 11 Attributes of the Timeslot class

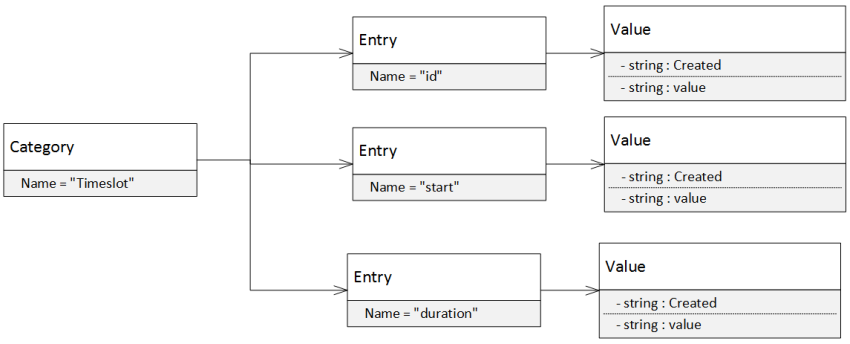

Fig. 13 Class Diagram for the Timeslot class

5.12 Algoweight Class

This structure contains the algorithm weight records associated with a time slot.

Name	Type	Description
id	String	The id of the algorithm weight object
renewables	String	The percentage of renewables optimized by

			the COS algorithm
	interconnect	String	The percentage of energy flows among regions, optimized by the COS algorithm
	specific	String	The percentage of energy consumed for market requirements optimized by the COS algorithm
	timeslot	Timeslot	The timeslot associated with the algorithm weight object (see 5.11)
	region	Region	The region to which the optimization the optimization algorithm weight object refers (see 5.20)

Table 12 Attributes of the Algoweight class

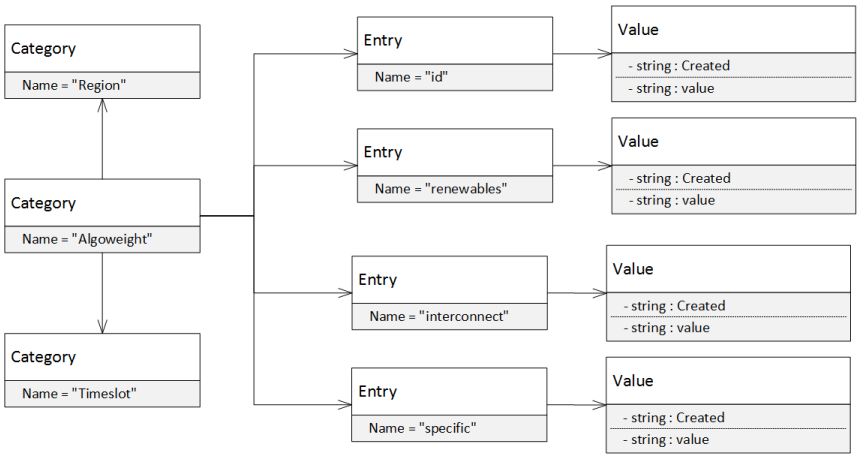

Fig. 14 Class Diagram for the Algoweight class

5.13 ChargingState Class

This class allows the platform to store a collection of charging state records associated with a time slot. It relies not only on its own attributes, regarding the actual charging, but also on four other classes.

	Name	Type	Description
	id	String	The id of the charging state instance
	change_requested	String	Indicates if a state change has been requested
	responded	String	The response from the servo for the state change
	requested_state	String	The state which has been requested on/off
	charging	String	Indicates if the electric vehicle is charging
	connected	String	Indicates if the electric vehicle is connected to an electric vehicle supply equipment
	battery_full	String	Indicates if the vehicle battery is full
	ev	Vehicle	The electric vehicle connected (see 5.16)
	ev_supply_equipment	EVSE	The associated electric vehicle supply equipment (see 5.15)
	timeslot	Timeslot	The timeslot associate with this charging state instance (see 5.10.)
	charging_mode	ChargingMode	The charging mode of the electric vehicle (see 5.14)

Table 13 Attributes of the ChargingState class

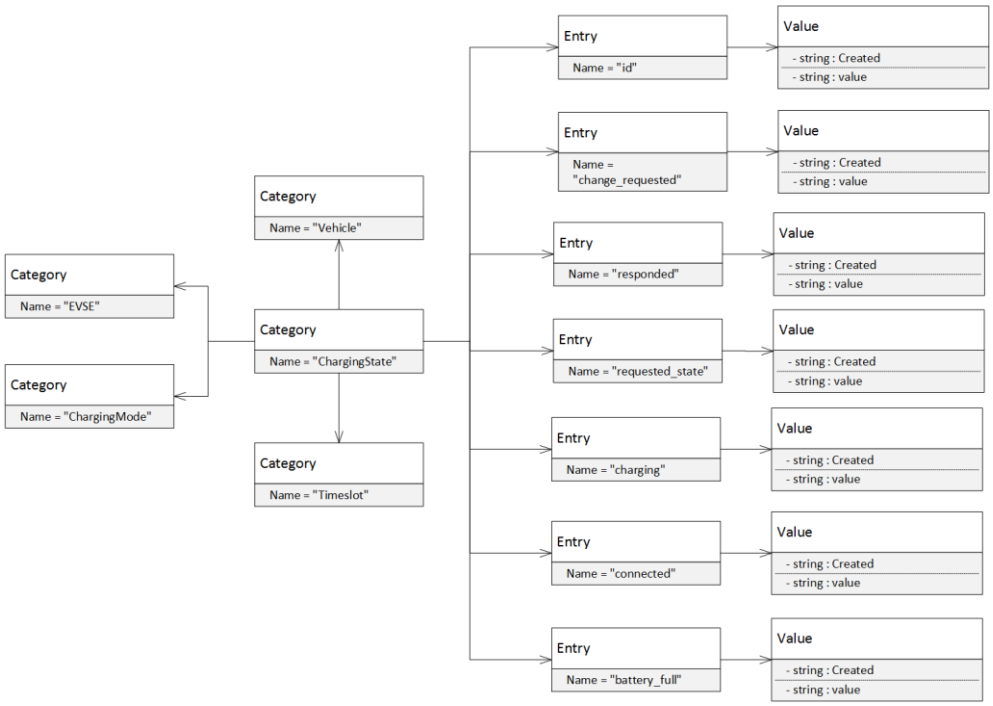

Fig. 15 Class Diagram for the ChargingState class

5.14 ChargingMode Class

This class is used to store information regarding the supported charging modes of an Electric Vehicle Supply Equipment (EVSE).

Name	Type	Description
id	String	The id of the charging mode
name	String	The name of the charging mode
power	String	The power dissipation of the charging mode in kW

Table 14 Attributes of the ChargingMode class

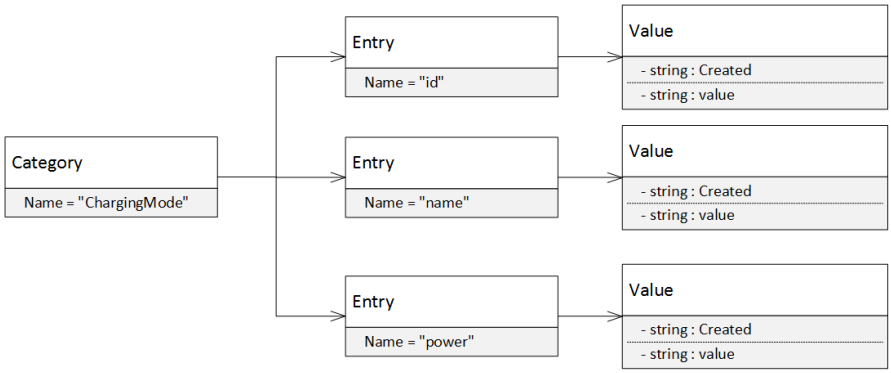

Fig. 16 Class Diagram for the ChargingMode class

5.15 EVSE Class

This class reports the information on an Electric Vehicle Supply Equipment (EVSE).

Name	Type	Description
id	String	The id of the EVSE

manufacturer	String	The manufacturer of the EVSE
model	String	The model of the EVSE
hw_version	String	The hardware version of the EVSE
fw_version	String	The software version of the EVSE
region_id	Region	The region associated with this EVSE (see 5.20)

Table 15 Attributes of the EVSE class

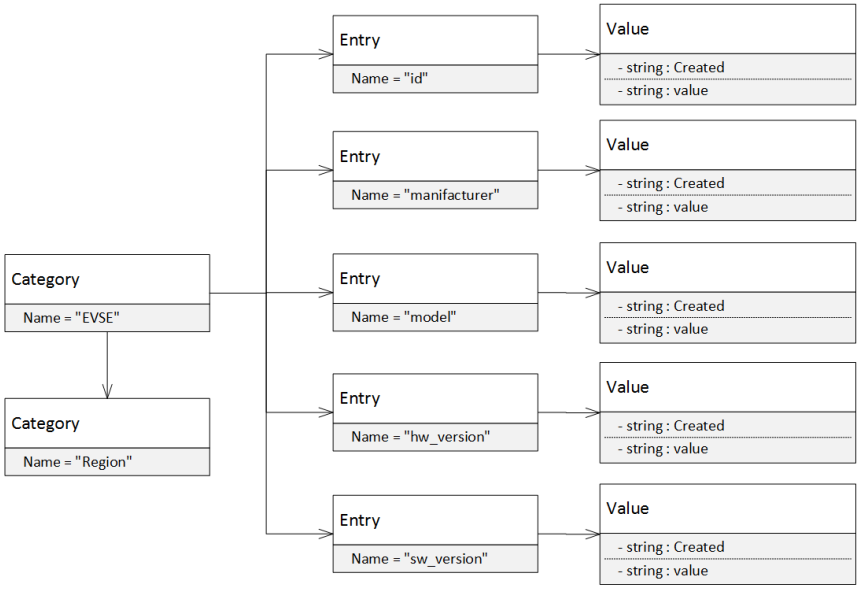

Fig. 17 Class Diagram for the EVSE class

5.16 Vehicle Class

The following class is intended to contain data about vehicles. The structure is similar to that of the "Building" class but instead of having and address there is another support category, called "VehicleType", that contains information about the type of vehicle and that is reported in 5.17.

Name	Type	Description
id	String	The id of the region
type	VehicleType	The name of the region
measurementpoints	MeasurementPoints	The types of measurements that are associated with the specific vehicle
relatedentities	RelatedEntities	A list of entities associated with the vehicle

Table 16 Attributes of the Vehicle class

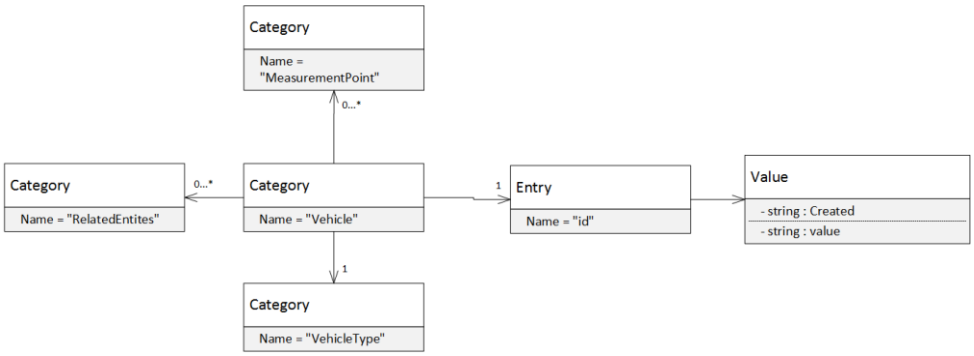

Fig. 18 Class Diagram for the Vehicle class

5.17 VehicleType Class

This class contains data regarding the type of a vehicle, like the manufacturer and the model or the autonomy range. All the attributes of the class are reported in Table 17.

Name	Type	Description
id	String	The id of the vehicle type
brand	String	The brand name of the vehicles of this type
manufacturer	String	The manufacturer of this vehicle type
capacity	String	The capacity of the vehicle battery in kWh
max_range	String	The maximum range of the vehicle type in km
min_range	String	The minimum range of the vehicle type in km

Table 17 Attributes of the VehicleType class

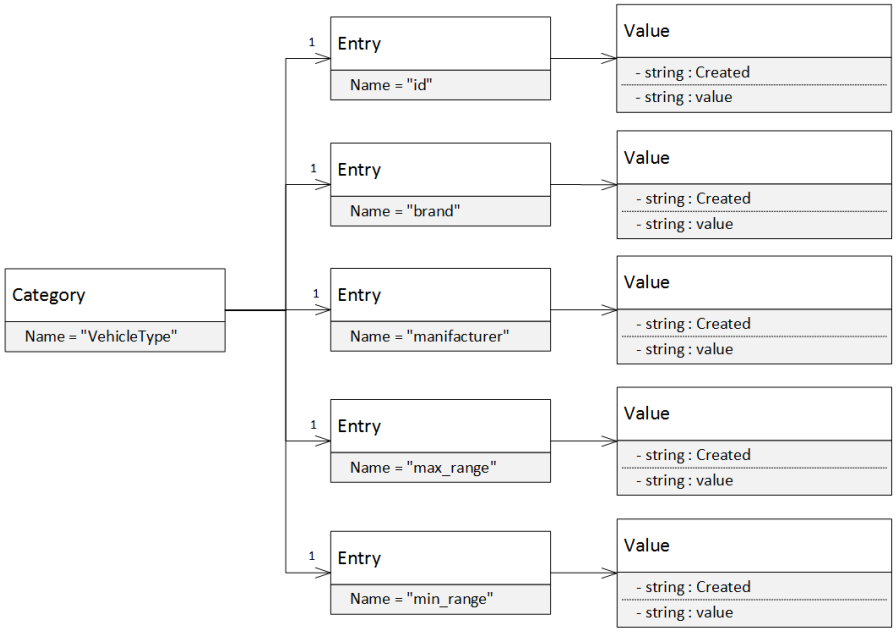

Fig. 19 Class Diagram for the VehicleType class

5.18 Connection Class

This class is used to store a collection of electric vehicle - electric vehicle supply equipment (EVSE) connections. It includes data regarding the exchanged energy, the charging process and the references to the connected vehicle and supply equipment.

Name	Type	Description
id	String	The id of the connection
start	String	The time when the connection state changed, in an ISO8601 compliant format
energy_current	String	Power being consumed by the connection in this minute
energy_previous	String	Power being consumed by the connection in the previous minute
duration	String	Duration of the connection (seconds)
total_energy_per_connection	String	The total power demand of this connection
total_energy	String	The total power demand of all

			connections by this electric vehicle supply equipment
□	**charging_mode_id**	ChargingMode	The id of the associated charging mode (see 5.14)
□	**EVSE_id**	EVSE	The id of the associated charging mode (see 5.15)
□	**connection_state_id**	ConnectionState	The id of the related connection state (see 5.19)
□	**vehicle_id**	Vehicle	The id of the electric vehicle associated with this connection (see 5.16)

Table 18 Attributes of the Connection class

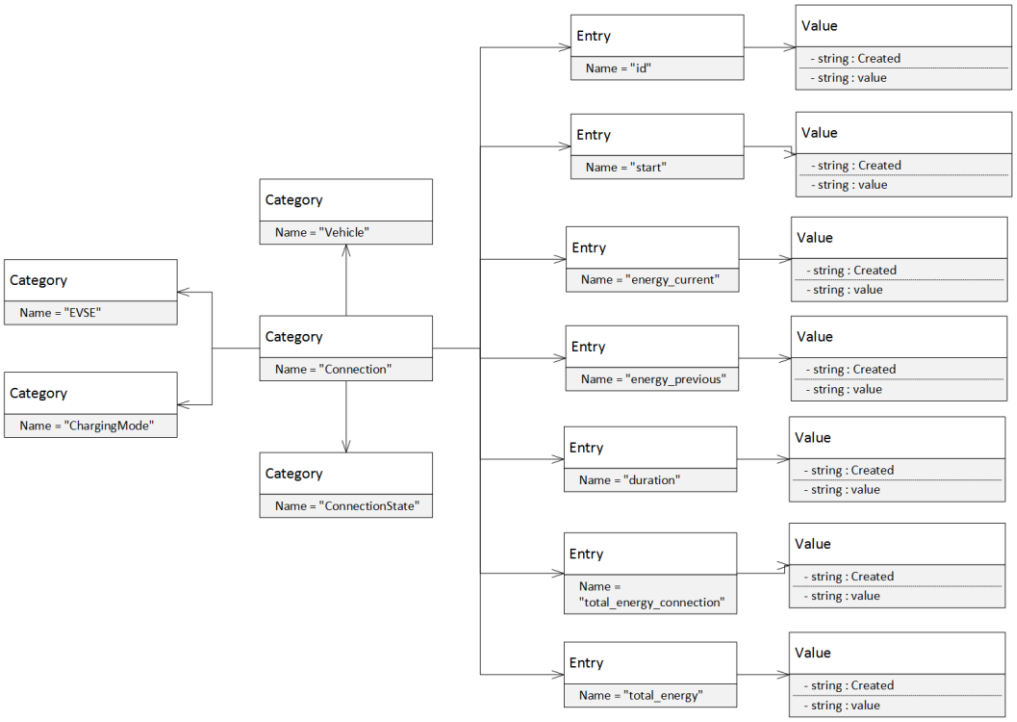

Fig. 20 Class Diagram for the Connection class

5.19 ConnectionState Class

This data structure contains the information regarding the connection state between the electric vehicles and the energy supply equipment.

Name		Type	Description
id	□	String	The id of the connection
name	□	String	The time when the connection state changed, in an ISO8601 compliant format
code	□	String	Power being consumed by the connection in this minute

Table 19 Attributes of the ConnectionState class

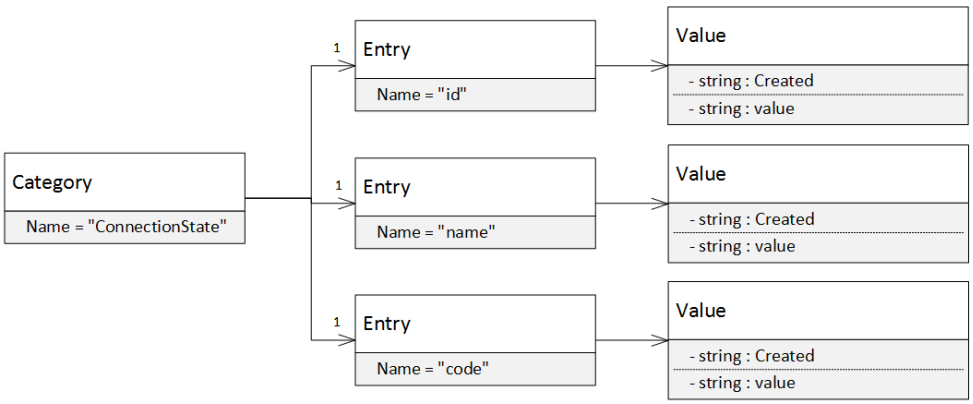

Fig. 21 Class Diagram for the ConnectionState class

5.20 Region Class

The following class allows the platform to store data about the regions in which each trial site can be divided.

Name	Type	Description
id	String	The id of the region
name	String	The name of the region
code	String	The code of the region
country	String	The country of the region

Table 20 Attributes of the Region class

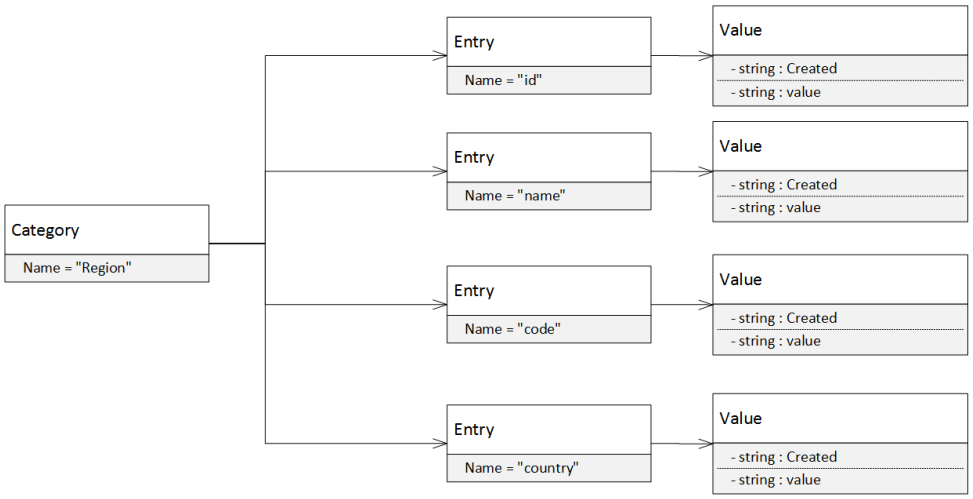

Fig. 22 Class Diagram for the Region class

5.21 Machine Class

This class stores detailed information about a single machine. They include the machine identification and measurement process information and the registered data stored, as previously seen, in a list of Entry-Value pairs called “data”.

Name	Type	Description
id	String	The id of the machine
name	String	The name of the machine
process	String	The process the machine is currently into
plc	String	The PLC id of the machine
monitor	String	The type of the meter monitoring the machine

resolution	String		The period of the measurements
res_unit	String		The unit of the resolution (e.g. minutes)
cumulative	String		Indicates whether the measurement is cumulative
data	String		The list of measurements of the machine

Table 21 Attributes of the Machine class

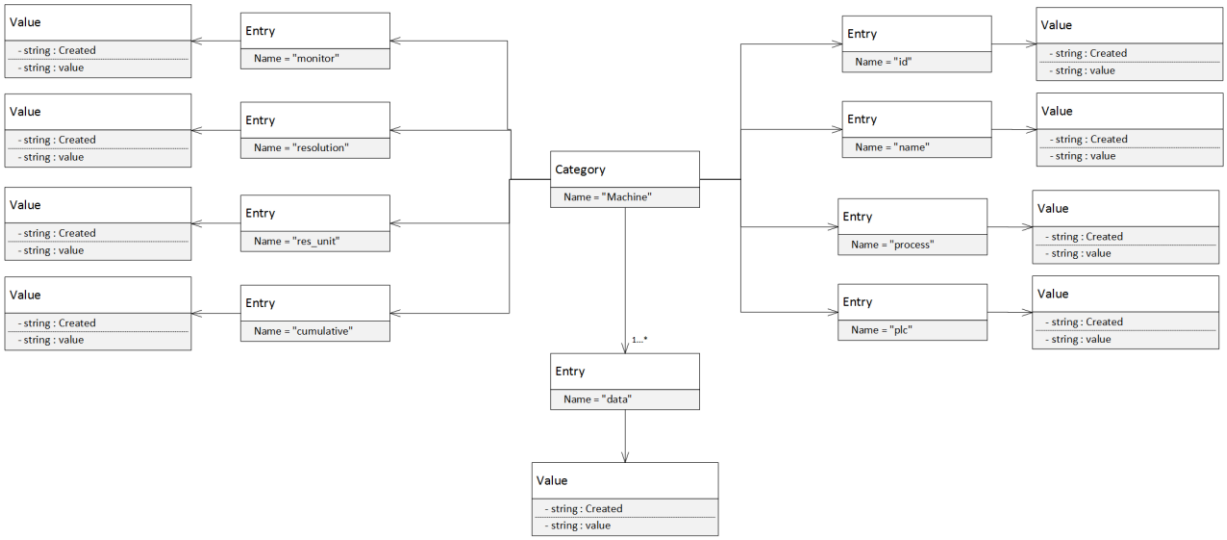

Fig. 23 Class Diagram for the Machine class

5.22 RegionalEnergy Class

This class is used to store a collection of aggregated regional energy records associated with a specific timeslot. It also contains data about forecasted energy for the region. The forecasted data, as explained previously, are stored in a different way than actual or historical data. In fact, instead of having just an Entry-Value pair, there is also a Forecast object between them. The Forecast object stores information about the creation of the forecast and its time validity.

	Name	Type	Description
	id	String	The id of the RegionalEnergy instance
	generated_energy	Double	The generated energy in kWh
	aggregated_energy	Double	The aggregate energy in kWh
	requested_energy	Double	The requested energy in kWh
	actual_energy	Double	The actual energy in kWh
	forecasted_energy	Double	The forecasted energy in kWh
	region_id	Region	The id of the region associated with this regional energy reporting (see 5.20)
	timeslot_id	Timeslot	The id of the timeslot associated with this regional energy reporting (see 5.11)

Table 22 Attributes of the RegionalEnergy class

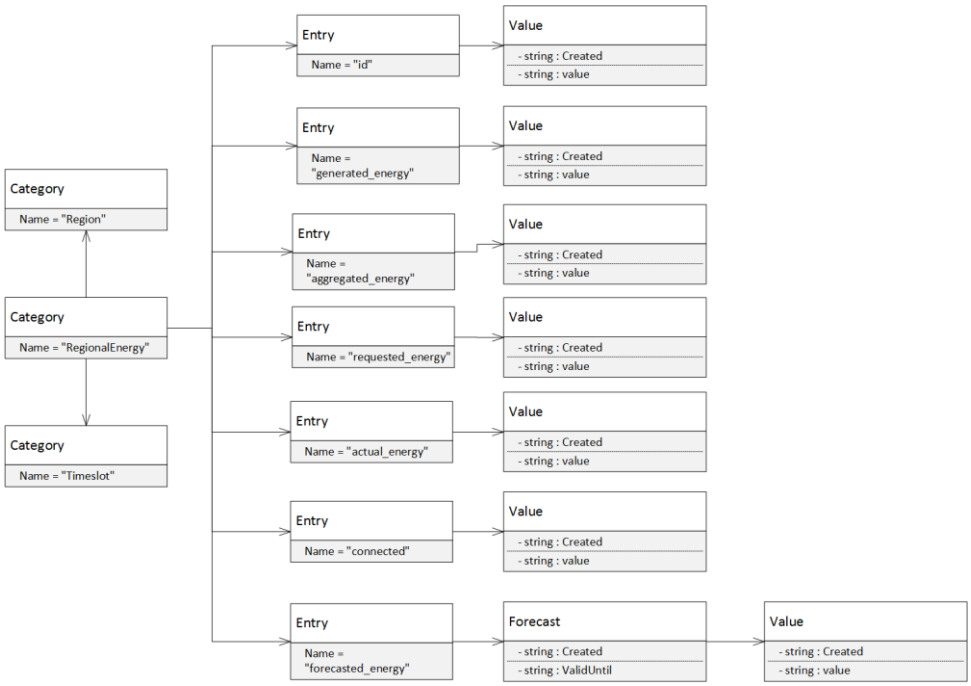

Fig. 24 Class Diagram for the RegionalEnergy class

5.23 SocialEvent Class

This service is meant to offer a list of social events that can temporarily influence the electricity consumption.

Name	Type	Description
location	Location	The location of the event (see 5.24)
type	String	The type of the event
significance	String	The significance of the event, in terms of influence to the smart grid load
event_time	String	The time of the event, in an ISO8601 compliant format
register_time	String	The registration time of the event, in an ISO8601 compliant format
address	Address	The address of the location

Table 23 Attributes of the SocialEvent class

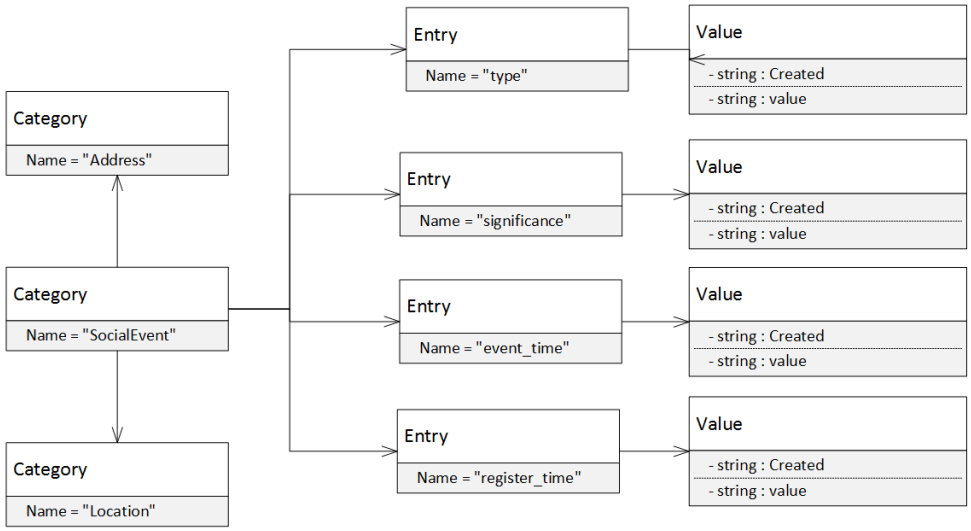

Fig. 25 Class Diagram for the SocialEvent class

5.24 Location Class

This class contains information on the physical location of an event. It reports the exact location by means of longitude and latitude.

Name	Type	Description
id	String	The id of the location
name	String	The name of the location
description	String	A description of the location
url	String	The URL for the location (if any)
latitude	String	The latitude of the location
longitude	String	The longitude of the location
timezone	String	The timezone of the location

Table 24 Attributes of the Location class

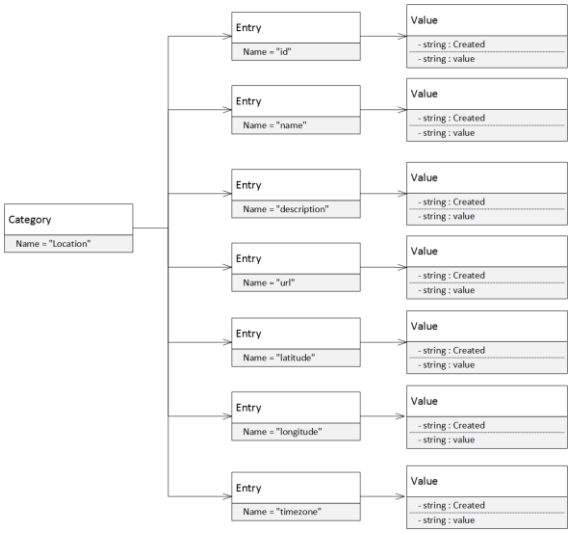

Fig. 26 Class Diagram for the Location class

5.25 Price Class

Allows a user to get information regarding the energy prices a certain area. The price in fact is connected to a specific location and, if possible, to a region.

Name	Type	Description
id	String	The id of the price object
location	Location	The location/region the price refers to
region	String	The name of the price region this location belongs to (if any)
start_time	String	The time from which on the price is valid, in an ISO8601 compliant format
end_time	String	The time until which on the price is valid, in an ISO8601 compliant format
measurementpoints	MeasurementPoint	The measurements available for the specific price location
currency	String	The currency of the prices listed
electricity_price	String	The price for electricity
heating_price	String	The price for heating energy
heating_offset	String	A parameter to calculate the heating energy price

Table 25 Attributes of the Location class

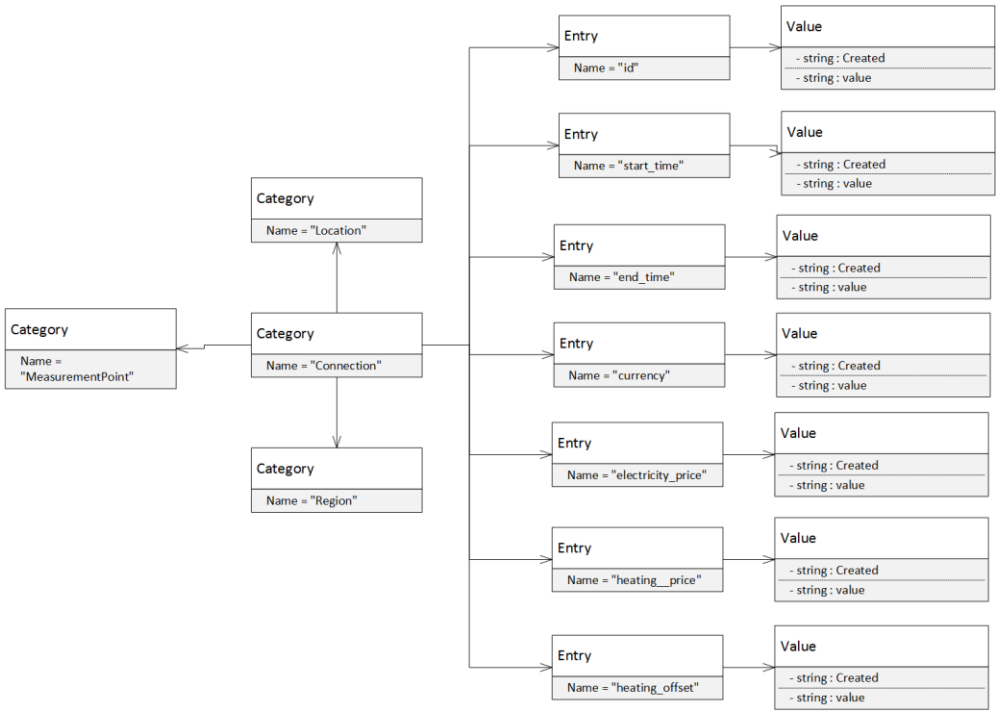

Fig. 27 Class Diagram for the Location class

6. Conclusion

This report documents the proposed consolidation of the FINESCE trial infrastructures' specific data models onto a single unified data model, the Smart Energy Model (SEM), which forms a core part of the Smart Energy Platform. A brief introduction on data models and data ontologies has been provided, in order to make the reader aware of the main concepts and the specific terminology. Then the modelling approach has been presented, with the needed references to the starting point used for the design of the meta-model and the unified data model. Its main

pillars are extensibility, reusability for future needs and ease of integration of other devices and data sources, using standardised or proprietary data models, into the Smart Energy platform. Finally the modelling approach has been applied to the data structures accessible over the FINESCE API. These data structures reflect that the FINESCE trials use different data models which partly overlap. In this work, a set of classes has been defined which unify the different data models exposed over the FINESCE API. These classes together constitute the unified data model (SEM).

7. References

1. Gruber T., (1995), Towards Principles for the Design of Ontologies Used for Knowledge Sharing, International Journal of Human-Computer studies, 43 (5/6): 907 – 928.
2. Guarino N. & Giaretta P., (1995), Ontologies and Knowledge Bases: Towards a Terminological Clarification, in Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, N. Mars (ed.), IOS Press, Amsterdam, pp 25 – 32.
3. Guarino N., (1998), Formal Ontologies and Information Systems, in Guarino N. (ed.), Proc. of FOIS98, IOS Press, pp. 3 – 15.
4. Guarino N. & Welty C., (2002), Evaluating Ontological Decisions with OntoClean, in Communications of the ACM, 45 (2): 61 – 65.
5. American National Standards Institute. 1975. ANSI/X3/SPARC Study Group on Data Base Management Systems; Interim Report. FDT (Bulletin of ACM SIGMOD) 7:2.
6. ISO, (1982), Terminology for the Conceptual Schema and Information Base, ISO Technical Report TR9007
7. Sheth A. & Kashyap V., (1992), So far (schematically) yet so near (semantically), in Hsiao D., Neuhold E. & Sacks-Davis R. (eds.), Proc. of the IFIP WG2.6 Database Semantics Conf. on Interoperable Database Systems (DS-5), Lorne, Victoria, Australis. North-Holland, pp. 283 – 312.
8. Ushold M. & King M., (1995), Towards a Methodology for Building Ontologies, in Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI95 workshop), also available as AIAI-TR-183 [ftp.aiai.ed.ac.uk/pub/documents/1995/95-ont-ijcai95-ont-method.ps.gz]
9. Ushold M. & Gruninger M., (1996), Ontologies: Principles, methods and applications, in The Knowledge Engineering Review, 11 (2): 93 – 155.
10. D1.1 Report on Requirements and Use Cases Specification. (2013). COOPERATE Control and Optimisation for energy positive Neighbourhoods
11. D1.2 Report detailing Neighbourhood Information Model Semantics. (2013). COOPERATE Control and Optimisation for energy positive Neighbourhoods
12. Gašević, D. & Devedžić, V. (2007), Interoperable Petri net models via ontology, in International Journal of Web Engineering and Technology, Vol. 3, no. 4, 2007, pp. 374–396.
13. Gašević, D., Djurić, D. & Devedžić, V. (2009), “Model Driven Engineering and Ontology Development”, 2nd edition, Springer, Berlin, Heidelberg
14. Kim, D., Yang, H., Jang, H, Hong, D., Falk, H., Kim, S. & Lee, B., A metamodeling approach to unifying IEC 61850 and IEC 61970, Innovative Smart Grid Technologies (ISGT), *IEEE PES,* (*2013*), *page 1-6.*

8. List of Abbreviations

SEM	Smart Energy Model
SEP	Smart Energy Platform
API	Application Programming Interface
XML	eXtensible Markup Language
ANSI	American National Standards Institute
NIM	Neighbourhood Information Model
UML	Unified Modeling Language
ISO	International Organization for Standardization
DSO	Distribution System Operator
DER	Distributed Energy Resources
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment