move us

ICT cloud-based platform and mobility services available,
universal and safe for all users

D3.1 Data models, object models and ontology definition

Deliverable Id :	**D3.1**
Deliverable Name :	**Data models, object models and ontology definition**
Status :	**Ready for quality check**
Dissemination Level :	**PU**
Due date of deliverable :	**M10**
Actual submission date :	**31/07/2014**
Work Package :	**WP3**
Organization name of lead contractor for this deliverable :	**TEC**
Author(s):	**Sergio Campos, Iraide Unanue**
Partner(s) contributing :	**ALL**

Abstract: This document describes the current outcome of Task 3.1, which addresses the definition of MoveUs data and objects model. At this stage, the focus relays on the main functional modules, supporting the provision of multimodal, personalized and eco-efficient mobility services.

HISTORY

Version	Date	Modification reason	Modified by
0.1	17/06/2014	TOC	TEC
0.2	03/07/2014	Registry, PT Operation, Traffic Management, Feedback	TEC with contributions from SOF, TEC
03	10/07/2014	Incentives, Energy Efficiency	TEC with contributions from SOF, QRY,CDG, TUT, TRE
04	18/07/2014	Madrid use-cases analysis, PT Operation, Journey Planning update, conclusions	TEC with contributions from SICE, EMT
0.5	21/07/2014	Description of model aspects	TEC
0.6	23/07/2014	Description of model aspects (Public Transport)	TEC
0.7	24/07/2014	Description of model aspects (Traffic Management)	TEC
0.9	11/08/2014	Quality Check	ATOS
1.0	11/08/2014	Final Version	ATOS

Contents

List of Figures

List of Tables

List of Abbreviations

<Abbreviation>	<Explanation>
App	Application
ATOS	ATOS SPAIN
BSM	Basic Safety Message
BT	Bluetooth
BTFix	Fix Bluetooth Reader
CF	Carbon Footprint
D	Deliverable
DATEX2	Standard for ITS on European Roads
EC	Energy Consumption
EMT	Empresa Municipal de Transportes de Madrid
FCD	Floating Car Data
ID	Identification
IFOPT	Identification of Fixed Objects in Public Transport
ISO	International Organization for Standardization
IT	Information Technology
ITS	Intelligent Transport System
KPI	Key Performance Indicator
MAP	Map Data
MSG	Message
POI	Point of Interest
PT	Public Transport
RSU	Road Side Unit
RT	Real Time
SAE	SAE international (Society of Automotive Engineers)
SICE	Sociedad Ibérica de Construcciones Eléctricas, S.A.
SIRI	Service Interface for Real Time Information

SOF	Softeco Sismat Srl
SPAT	Signal Phase And Timing Message
SRM	Signal Request Message
SSM	Signal Status Message
TECNALIA	Tecnalia Research and Innovation
TPEG	Transport Protocol Experts Group
TRE	Tampereen Kaupunki
TUT	Tampere University of Technology
UC	Use-case
UML	Unified Modelling Language
URL	Uniform resource locator (internet)
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
VIM	Vehicle Interface Module
WP	Work Package

Executive Summary

WP3 as a whole is concerned with the MoveUs Architecture design and the platform component specification, to be implemented and deployed in the different city pilots. Specifically, this deliverable addresses the **underlying data and object models**, able to support the **information needs** of the different **processes** and **interfaces**, with existing data sources, field devices and involved actors.

The methodology to implement this data model, starts **eliciting the information needs** by analysing the different **use-cases**, continues with the **validation of these data requirements**, checks their **coverage by previous reference data models** (from eMotion, In-time and Co-Cities projects) and finally, **fills existing gaps**.

The conceptual MoveUs Data Model has been set up in Unified Modelling Language (UML), by using the Enterprise Architect CASE Tool. This ensures standard compliance and service generation support.

The main conclusion remarks that **existing models cover a significant portion of the concepts** needed for the storage of information and provision of MoveUs services. Nevertheless, the most innovative project goals: **incentive management, energy efficiency, services customization and specific intelligent traffic management** have not been previously addressed, so **appropriate extensions have been provided**.

Furthermore, the work package scheduling determines an **iterative approach**, constituting this document a quite mature version, which will probably need to be updated to perfectly match the platform and services specification and design, which will be completed by the end of the 1st project year (M12).

1 MoveUs Overview

This Deliverable D3.1 is the first deliverable expected for **WP3 – Analysis, Specification and Design of the MoveUs Architecture and City Services.**

The **objectives** of WP3 are:

- To define the data models relevant for MoveUs operation.
- To define the high-level architecture for the MoveUs Cloud-based platform and its functional specifications in detail.
- To provide detailed specifications and design for the set of services to be provided in MoveUs pilots.
- To identify the data security and privacy issues to be taken into account in the MoveUs architecture and include them in the definition of the platform and services.
- Develop innovative business models determining the users' willingness to pay for the uptake of MoveUs services.

Specifically, Task 3.1 is aimed at collecting and identifying major data protocols and data and object models relevant for MoveUs. After this analysis versus previously elicited use-cases, and aligned with architectural and service design, a common data model is defined.

2 Methodology

The data and objects models definition included in this document has been developed following the next key concepts and methodology:

It is important to remark that one of the main objectives of the MoveUs project is to develop a data model following the more relevant related standards and previous European projects. In this way, the data model will not start from the scratch, reusing previous development and aligning itself with other European initiatives and standardization activities, and consequently, making the whole project more interoperable. Another point to stand out is that the model is formally defined in UML (Unified Modelling Language) by using the EA (Enterprise Architect) tool.

The methodology followed to implement this data model encompasses the following steps. First of all, an initial identification of MoveUs use cases requirements has been done in order to detect the information needed in the project. After this analysis, a complete resource evaluation from previous projects has been done identifying 3 European projects: eMotion [1], In-Time [2] and Co-Cities [3]. The next step of this methodology is to clearly identify data requirements for each pilot according to the use cases, the validation of these data requirements with each pilot and, finally, to check whether these data requirements are covered by eMotion, In-Time or Co-Cities data models by filling, eventually, the possible gaps. The result of this process is the **MoveUs data model**.

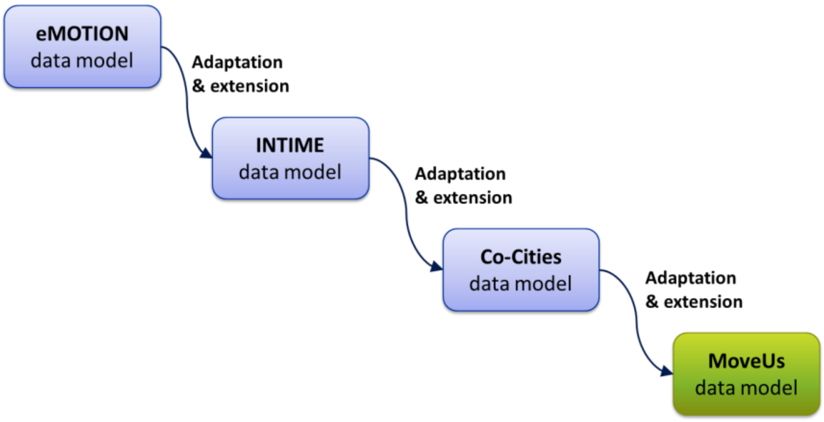


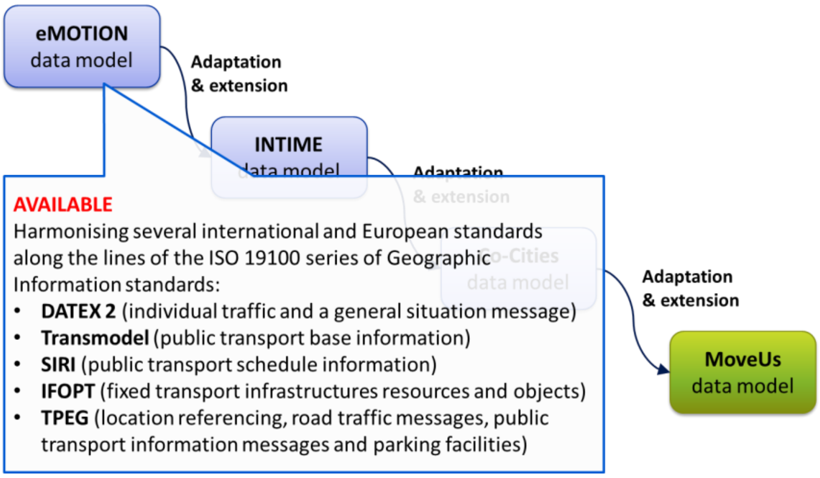
Figure 1 MoveUs data model definition process

Before going deeply into MoveUs data model, a concise description of the identified European projects is presented in this document:

eMOTION was an initiative co-funded by the European Commission under the thematic area Sustainable Development, Global Change and Ecosystems of the 6th Framework Programme for Research and Development, May 2006-July 2008. In relation to the work presented herein, its main contribution was to develop a data model (available at [1]) harmonising several international and European standards along the lines of the ISO 19100 series of Geographic Information standards:

- **DATEX 2:** individual traffic and a general situation message.

- **Transmodel:** public transport base information.
- **SIRI:** public transport schedule information.
- **IFOPT:** fixed transport infrastructures resources and objects.
- **TPEG:** location referencing, road traffic messages, public transport information messages and parking facilities.



AVAILABLE
Harmonising several international and European standards along the lines of the ISO 19100 series of Geographic Information standards:

- **DATEX 2** (individual traffic and a general situation message)
- **Transmodel** (public transport base information)
- **SIRI** (public transport schedule information)
- **IFOPT** (fixed transport infrastructures resources and objects)
- **TPEG** (location referencing, road traffic messages, public transport information messages and parking facilities)

Figure 2 eMotion status & scope

In-Time project was co-funded by the European Competitiveness and Innovation Programme, PSP-ICT for adaptive urban transport management infrastructure and services, starting 1st of April 2009 with duration of 3 years. Based on eMotion, In-Time data model is already available at [2] covering the following concepts:

- Static road traffic
- Dynamic road traffic & weather
- Points of interest + static and dynamic parking
- Static and dynamic public transport
- Dynamic traffic event
- Static and dynamic flight
- Dynamic multimodal journey planning

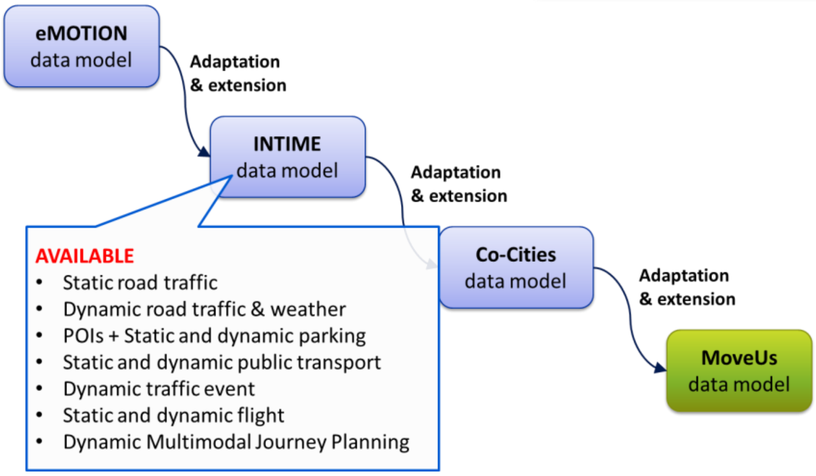


Figure 3 In-Time status & scope

Co-Cities was an European pilot project aiming to extend and validate existing mobility services to improve current traffic information management in cities and urban areas. The novelty about the Co-Cities services was their cooperative feature permitting the end users to report their feedback to the traffic management centres. This point is the main contribution to the data model, as it is also based on In-Time and eMotion projects. While Co-Cities data model is not yet publicly available at the time of writing the present document, formal contact has been established between project coordinators in order to facilitate the re-use and adaptation of this model to MoveUs needs, thanks to the mediation of SOF and ATOS as partners involved in both projects.

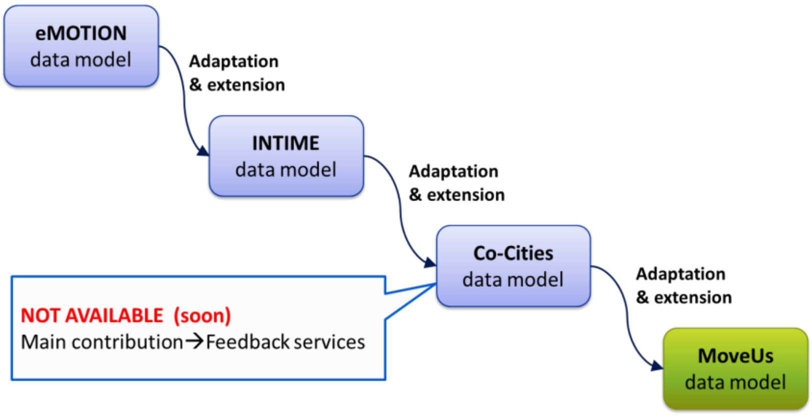


Figure 4 Co-Cities status & scope

3 MoveUs data model

Although performed in parallel with task T3.2, the data model is aligned with the different functional blocks so far identified on the functional architecture. Moreover, the fact of relying on the FRAME methodology [4] helps to identify high-level information needs by means of the DFD’s data stores, conceptual repositories of information supporting selected functionalities.

At this stage of the project, the following functional blocks can be identified:

- User Management
- Traffic Management
- Public Transport Operation Management
- Incentive Management
- CF/EC Estimation
- Feedback
- Register

3.1 Functional block User Management

The requirements in terms of User Management, identity provision and access rights emerging from the Use Case definition suggest that in principle a basic level of service could be provided for anonymous users. A user profile, instead, is required for:

- Personalized access to the mobility services
- Management of the incentives-related operations

The level of complexity of the user profile, associated to the two previous aspects can vary from a simple pair: userID - password to a complete personal profile made of a rich set of datatypes for the storage of preferences, habits, personal settings etc.

3.1.1 Existing specifications

For the present and future needs in MoveUs, the definition a complete user profile, based on the possibilities identified in the Use Cases is foreseen. This includes also personal information that is non-mandatory[1] in the data model.

[1] The feature types introduced in the data model can have attributes with different multiplicity, identified with square brackets and two identifiers: one for the lower limit and one for the upper limit of possible instances of the attribute. A multiplicity of [0..1] for instance indicates that the attribute is not mandatory (zero or one instances are allowed) like in the case of personal data that are introduced and defined in the current data model definition but may be not used in the city services.

Considering the general requirements and especially those specifically related to the incentive management, the analysis of the existing Co-Cities model led to the decision that a completely new, dedicated package was appropriate.

3.1.2 Extensions

A basic user profile is defined as a super class of the more specific user types.

The basic user profile is defined with attributes necessary for granting secure access and basic access management operations:

- Activation status
- Logging
- UserID and password recovery
- Role management

Roles can be created to grant different access levels to the city services in addition to the mechanisms already foreseen for the activation/deactivation of certain functions.

The main characterization in terms of user types comes from the FRAME-based methodology adopted for the design of the MoveUs architecture.

Specifically, considering the actors according to the FRAME definitions and the above mentioned requirements on user management, the following user types and related features have been identified as the most significant ones:

- Incentive-related User Types (*MV_UserType_I*)
- Drivers (*MV_UserType_D*)
- Travellers (*MV_UserType_T*)

An additional User Type is defined for convenience as *MV_UserType_MV*. This is the generic MoveUs User type that can be assigned to both anonymous and registered users whenever necessary.

The classes for other FRAME user types are defined as placeholders for future use.

All user types and the sub-types defined as enumerations are used as part of the City Services access profile (see also section **Error! Reference source not found.** on the Registry)

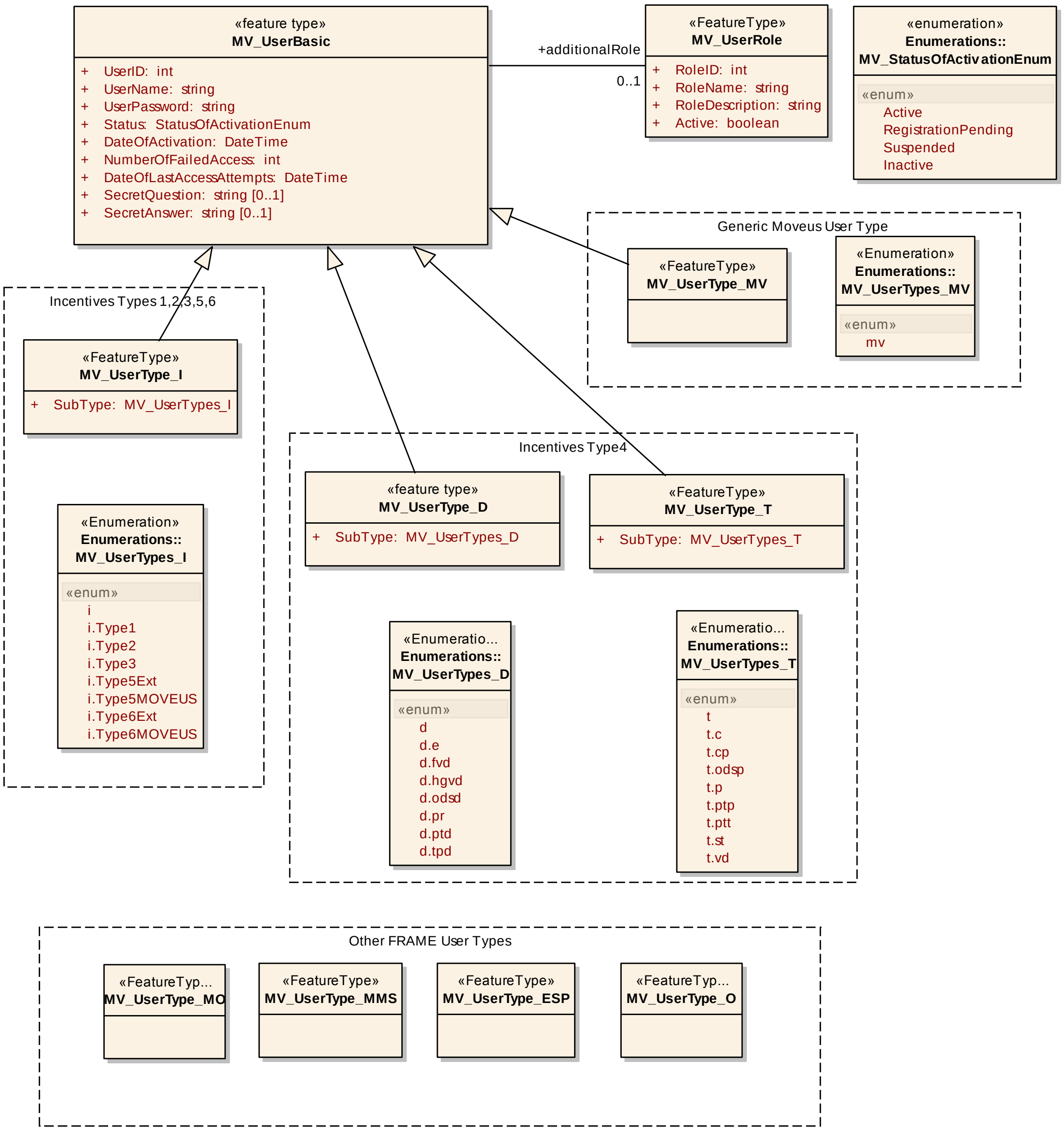


Figure 5 User management extensions (Taxonomy)

The sub-types attribute for each user type defines the specific actor (user) within a main user category.

d.e	Emergency Vehicle Driver
d.fvd	Freight Vehicle Driver
d.hgvd	Hazardous Goods Vehicle Driver
d.odsd	On-Demand Service Driver
d.pr	Private Driver
d.ptd	Public Transport Driver
d.tpd	Trip Planning Driver
t.c	Cyclist
t.cp	Car-Pooler
t.odsp	On-Demand Service Passenger
t.p	Pedestrian
t.ptp	Public Transport Passenger
t.ptt	Pre-Trip Traveller

t.st	Static Traveller
t.vd	Vehicle Driver

Table 1 MoveUs users

The characterization of *MV_UserType_I* comes from the definitions given for the Incentives-related operations[2]:

- **Electronic Wallet Registry**: set of payments systems;
- **Coupon**: a digital code that allow you to get discounts;
- **Voucher**: a digital code that corresponds to a purchase prepaid;
- **Advertisement**: a set of information and data that can be used to publish or link an advertisement;
- **Incentives**: set of material and virtual objects that help modifying the mobility behavior to obtain a reduction of driving and/or an use of alternative modes (i.e. from private vehicle to public transportation, or to a higher Euro class). Incentives can be also defined as the generic 'money' that can be spent to get benefits or coupons.

The user types defined for these objects are:

- **Type 1**: Entity defining RULES
- **Type 2**: Entity providing INCENTIVES
- **Type 3**: Entity where INCENTIVES can be spent, entity providing awards (benefits that can be obtained with a certain amount of incentives)
- **Type 4**: Final users (these are defined in the model as User Types "D" and "T")
- **Type 5**: Entity providing COUPONS. They can be:
    - **UT5_MOVEUS:** MoveUs Internal module usable by other entities to provide COUPONS.
    - **UT5_EXT:** External entities providing directly COUPONS via a MoveUs interface available for this purpose.
- **Type 6**: Entity providing ADVERTISEMENTS
    - **UT6_MOVEUS:** MoveUs Internal module usable by other entities to provide ADVERTISEMENT
    - **UT6_EXT:** External entities providing directly ADVERTISEMENT via a MoveUs interface available for this purpose

From this definition, the different types of users involved in the incentive model can be clustered into two main categories:

- Final users (private or professional users): users of Type 4
- Organizations: all other types of users

Two main sets of attributes can then be defined and associated to the above categories. These attributes are identified considering that the "Organization" object is mainly defined for the purposes of incentives assignment and definition while the users of type 4 are those who typically access the MoveUs mobility City Services and can do this in a personalized way thanks to the mobility profile they have defined and that is constantly updated and refined within the normal service

[2] Refer to Deliverable D2.2 [6] for more information on the Incentives model

usage. The assumption, given these considerations is that the users of type 4 are exactly the **Drivers** and the **Travellers**, each with a specific characterization:

- **Drivers**: generally professional drivers that can use a specific profile of MoveUs services
- **Travellers**: end users that can use the common MoveUs services and in a few cases specific parts of them (e.g. the carpooling)

3.1.2.1 User Type 4

The User type 4 is described by a personal profile where all strictly private information is non-mandatory.

Each main Class (*MV_UserType_D* and *MV_UserType_T*) has specific feature types associated to it that form an extended profile used for the operations in MoveUs.

The **Traveller** profile comprises:

- Interest
- Mobility-related preferences and parameters
- Settings on the possibility of receiving notifications or get tracked.
- The Carpooling-related profile with the preferences and parameters necessary for the carpooling service according to the definition given in the Use Case definition (D2.2) [6]. This data type support the definition and storage of the carpooler’s profile and can be used eventually at applicative level to find and match the different trip offerings:
    - Indicators about habits of the carpooler (smoker, has animals etc.)
    - Temporal thresholds allowed for departure and arrival time
    - Spatial thresholds for departure and arrival position
    - Size of an ideal corridor around the journey path where possible pick-ups are allowed
    - Preferences about other carpoolers

The Vehicle profile and the Preferences are, instead associated to the personal profile because these are in common to both Drivers and Travellers.

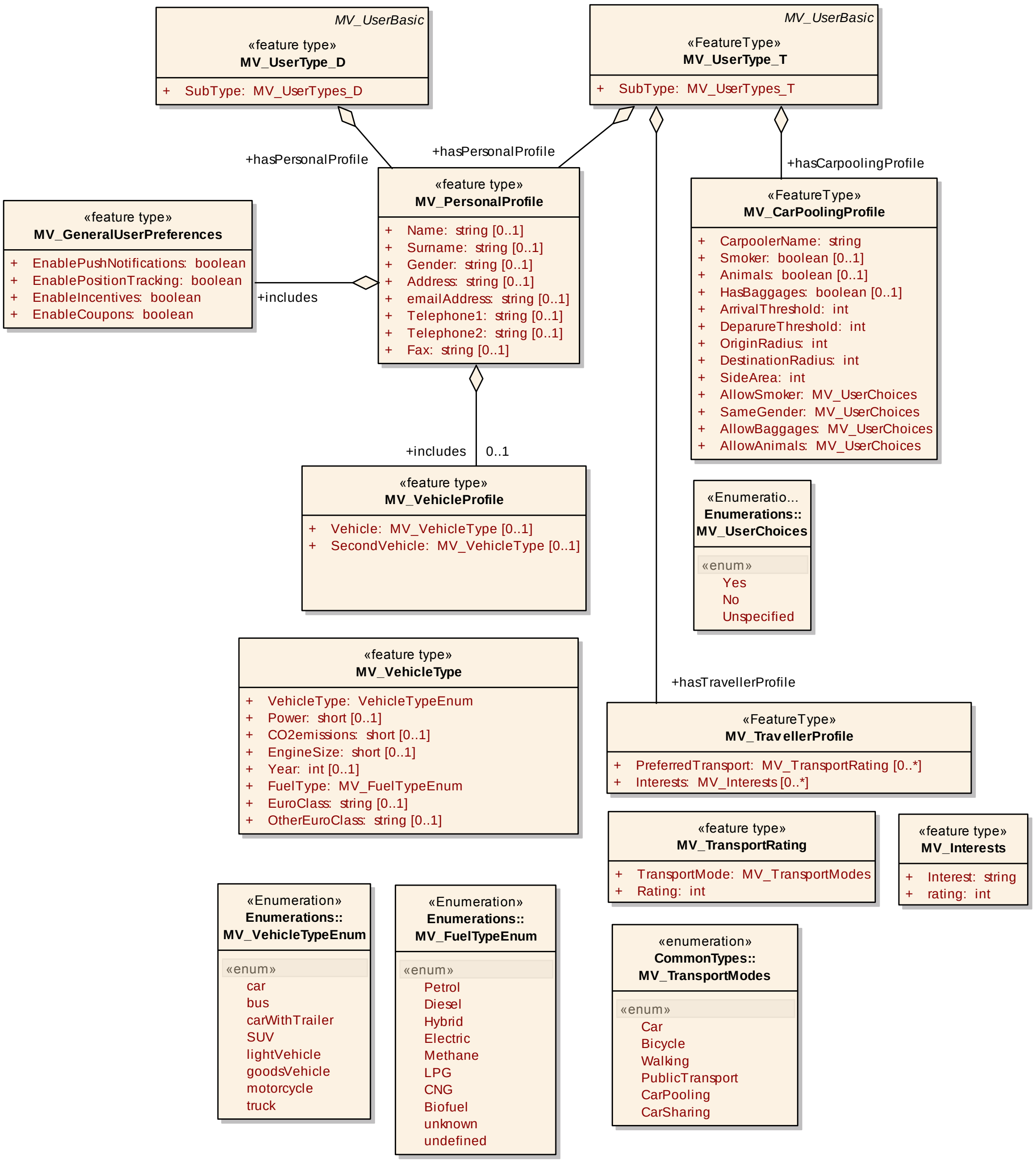


Figure 6 User management extensions (Information)

3.1.2.2 Organizations

The Organization feature type describes the User types that can be classified as organizations or companies. These are especially user types "I" but can be also other FRAME actors.

A personal profile is present as an attribute (Contact Person) and is described by the same feature type that describes the user type 4.

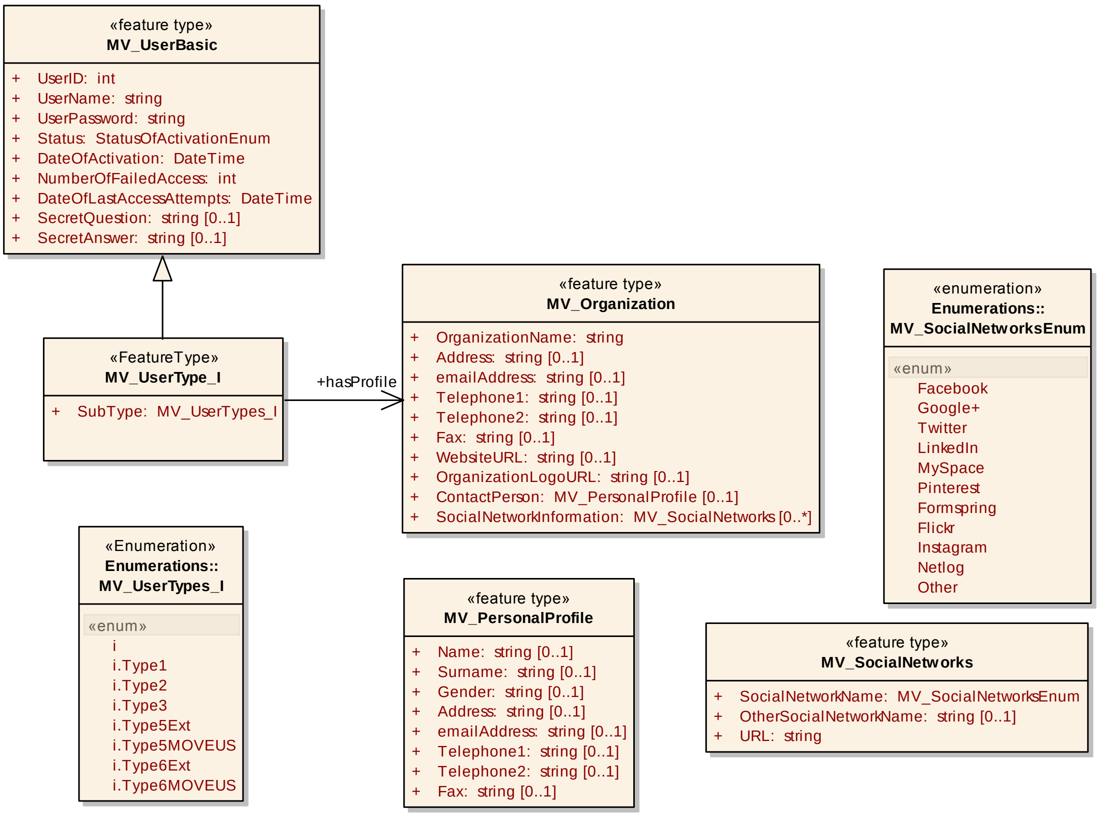

«feature type»
MV_UserBasic
+ UserID: int
+ UserName: string
+ UserPassword: string
+ Status: StatusOfActivationEnum
+ DateOfActivation: DateTime
+ NumberOfFailedAccess: int
+ DateOfLastAccessAttempts: DateTime
+ SecretQuestion: string [0..1]
+ SecretAnswer: string [0..1]
«FeatureType»
MV_UserType_I
+ SubType: MV_UserTypes_I
+hasProfile
«feature type»
MV_Organization
+ OrganizationName: string
+ Address: string [0..1]
+ emailAddress: string [0..1]
+ Telephone1: string [0..1]
+ Telephone2: string [0..1]
+ Fax: string [0..1]
+ WebsiteURL: string [0..1]
+ OrganizationLogoURL: string [0..1]
+ ContactPerson: MV_PersonalProfile [0..1]
+ SocialNetworkInformation: MV_SocialNetworks [0..*]
«enumeration»
Enumerations::
MV_SocialNetworksEnum
«enum»
Facebook
Google+
Twitter
LinkedIn
MySpace
Pinterest
Formspring
Flickr
Instagram
Netlog
Other
«Enumeration»
Enumerations::
MV_UserTypes_I
«enum»
i
i.Type1
i.Type2
i.Type3
i.Type5Ext
i.Type5MOVEUS
i.Type6Ext
i.Type6MOVEUS
«feature type»
MV_PersonalProfile
+ Name: string [0..1]
+ Surname: string [0..1]
+ Gender: string [0..1]
+ Address: string [0..1]
+ emailAddress: string [0..1]
+ Telephone1: string [0..1]
+ Telephone2: string [0..1]
+ Fax: string [0..1]
«feature type»
MV_SocialNetworks
+ SocialNetworkName: MV_SocialNetworksEnum
+ OtherSocialNetworkName: string [0..1]
+ URL: string

Figure 7 User management extensions (Organizations)

3.2 Functional block Traffic Management

This functional block encompasses the different functionalities to be included to monitor in real-time the state of roads and public spaces, to detect and manage the impact of incidents and provide road transport operation improvements in terms of energy-efficiency and final user safety and satisfaction.

It also establishes appropriate connections with external service and information providers, both to receive information and request/execute commands.

Several data stores or information repositories have been identified as relevant. The first distinction can be stablished between static and dynamic (real-time) information; also the nature of information and further treatment determines the different data stores.

Urban Road Static Data. This static data covers the actual layout, topology and configuration of the urban road network, being used as reference by a variety of functions to monitor, regulate and predict road traffic. It could be also used by Public Transport Operation System (buses, tram) to define services, routes and schedules, overlapping both networks and other systems.

Inter-urban Road Static Data. It shall contain the static data for the inter-urban traffic road network managed by the system, being its meaning analogous to the urban one. In the scope of MoveUs, only segments coming to urban city have significance (as traffic source or sink).

Urban Traffic Data. It contains traffic flow and other traffic related data for the urban road network. The data in the store shall be divided into two parts comprising historic and current data.

Inter-urban Traffic Data. Analogous semantics as previous one, only segments coming to urban city are relevant.

Incident Data. Collected data about mobility incidences (e.g. traffic, maintenance, events, environmental). Commonly, the information is captured and refreshed iteratively, evolving In-Time aspects.

Urban Car Park Data. Static and dynamic data related to the car parks available in the urban zone: location, availability and occupancy level.

Road Traffic Prediction Data. In MoveUs, these data will be produced from previously collected data (e.g. historical register), by application of prediction rules involving relevant parameters (e.g. day of year/week, hour, incidences). Detailed information needs will be elicited based on WP5 algorithms.

Environmental Data. It integrates data about the environmental conditions within the geographic area managed by the System.

3.2.1 Existing specifications

Urban Road && Inter-urban Road Static Data

In-Time project already defines a **network data model** including a **location reference** and **road network specification,** being both specifications general enough to be applicable to different domains (Figure 8).

Starting with the location reference specification, it can be used to give **points** (*PointLocationReference*), **lines** (*LinearLocationReference*) or **areas** (*ArealocationReference*) addressing different geometry types (Figure 8). It is important to remark that different locations can be grouped by using *LocationReferenceCollection* class. For extended details see [2].

In-Time network model defines 2 abstract feature types to represent **network points** (nodes, *NetNode*) and **linear areas** (links, *NetLink*). Both features are *NetElements* although they differ in their attributes: *NetNode* has a mandatory *point* attribute (*GM_Point)* representing a point geometry and *NetLink* are edges in a network graph with *curve* attribute (*GM_Curve*).

Going deeply into the **road networks**, it is important to point out that In-Time specification uses concepts from EuroRoads project and defines GML as encoding for data exchange, based on the ISO TC211 framework of "geo-standards". This model is even simpler than EuroRoads model as it is limited to just nodes (*RoadNetNode* and links (*RoadNetLink*) maintaining its attributes e.g. *formOfNode* for *RoadNetNode* describing the network node type, such as junction, roundabout, etc. For ferry (*FerryLink*) and road (*RoadNetLink*) links, the class *RoadNetLink* is defined. For extended details see [2].

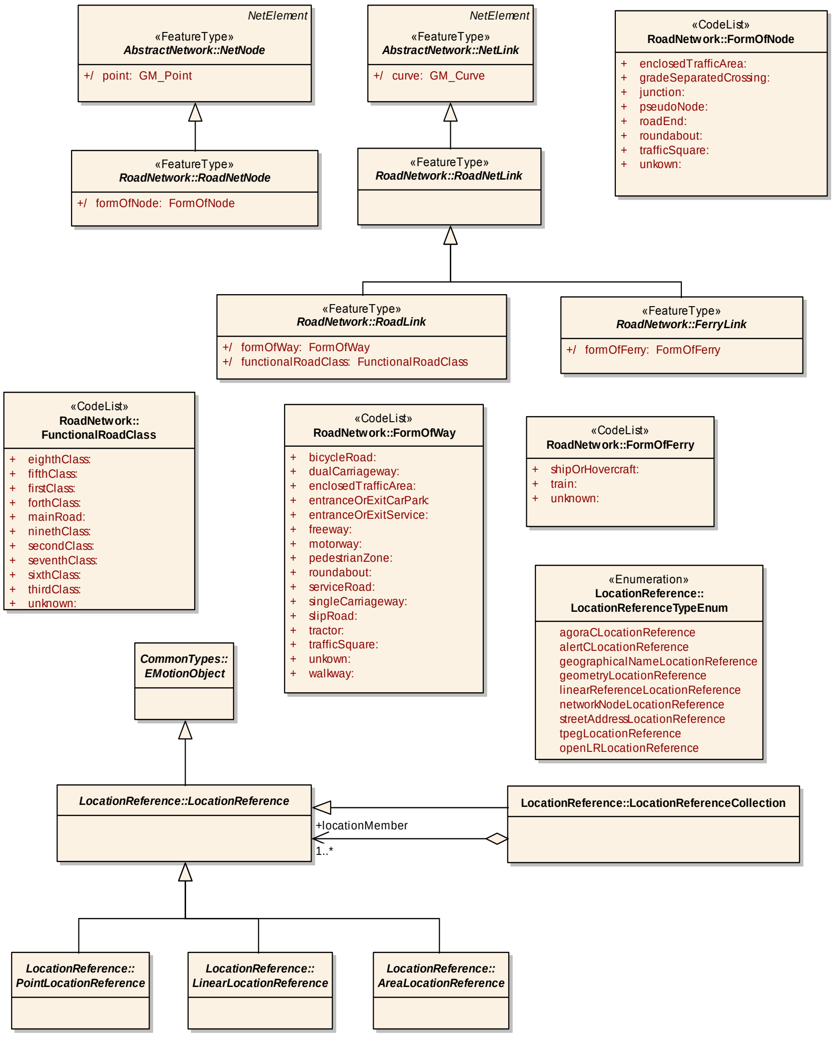


Figure 8 Road Data Model

Urban Traffic Data && Inter- urban Traffic Data && Environmental Data && Road Traffic Prediction Data

Regarding dynamic traffic information, the In-Time data model provides a specific package *TrafficRelatedInformation*, mainly based on DATEX II, that differences between *TrafficRelatedData* and *TrafficRelatedSituation* (that will be analysed later, in incident data section). *TrafficRelatedData* manages both **sensored data**, comprising measured: volume, density, velocity/speed, individual travel times and delay time and also **derived** (**processed**) **data** like segment level of service. Any of these values could represent a **real** or **forecasted measure**.

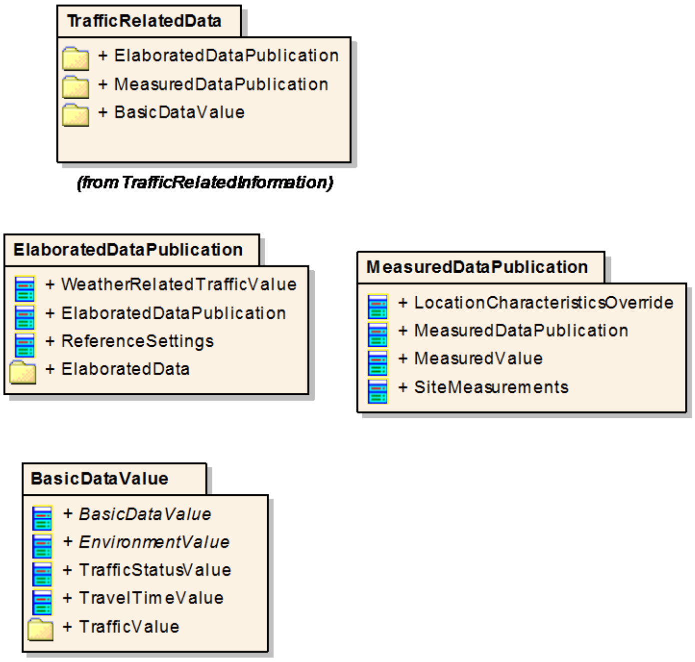


Figure 9 Traffic Related Data

Measured data are captured, usually periodically, by direct sensors or equipment (e.g. loops, cameras, weather stations) as traffic values (flow, speed, traffic density and individual vehicle data (FCD)), environmental/weather values (pollution, temperature, wind and precipitations), travel times and traffic status.

A measurement data set is represented by the *SiteMeasurements* class, associated locally to a site or location (*measurementSiteReference)* and temporal period *(measurementTimeDefault*).

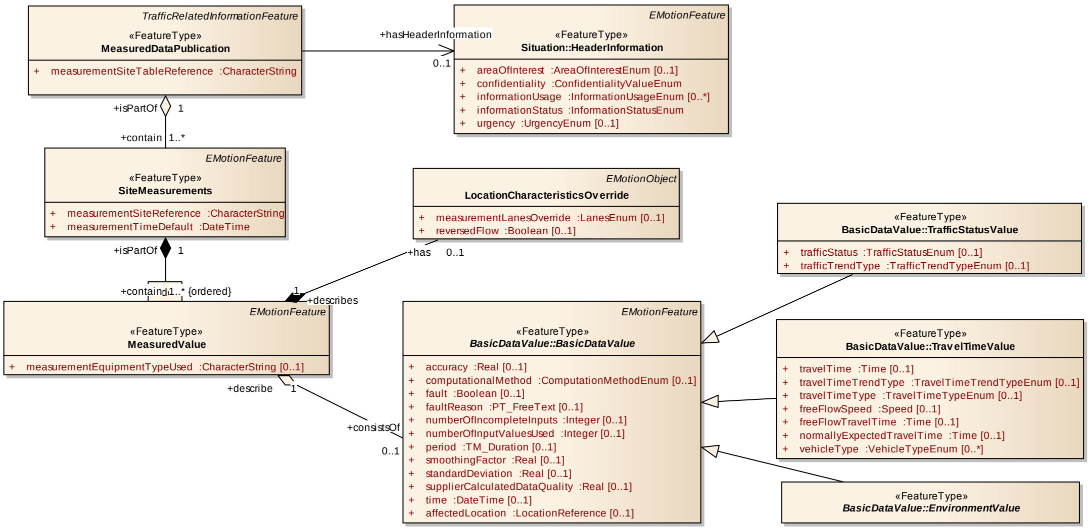

TrafficRelatedInformationFeature
«FeatureType»
MeasuredDataPublication
+ measurementSiteTableReference :CharacterString
+hasHeaderInformation
0..1
EMotionFeature
«FeatureType»
Situation::HeaderInformation
+ areaOfInterest :AreaOfInterestEnum [0..1]
+ confidentiality :ConfidentialityValueEnum
+ informationUsage :InformationUsageEnum [0..*]
+ informationStatus :InformationStatusEnum
+ urgency :UrgencyEnum [0..1]
+isPartOf 1
+contain 1..*
EMotionFeature
«FeatureType»
SiteMeasurements
+ measurementSiteReference :CharacterString
+ measurementTimeDefault :DateTime
+isPartOf 1
+contain 1..* {ordered}
EMotionObject
LocationCharacteristicsOverride
+ measurementLanesOverride :LanesEnum [0..1]
+ reversedFlow :Boolean [0..1]
+has 0..1
1 +describes
EMotionFeature
«FeatureType»
MeasuredValue
+ measurementEquipmentTypeUsed :CharacterString [0..1]
+describe 1
+consistsOf 0..1
EMotionFeature
«FeatureType»
BasicDataValue::BasicDataValue
+ accuracy :Real [0..1]
+ computationalMethod :ComputationMethodEnum [0..1]
+ fault :Boolean [0..1]
+ faultReason :PT_FreeText [0..1]
+ numberOfIncompleteInputs :Integer [0..1]
+ numberOfInputValuesUsed :Integer [0..1]
+ period :TM_Duration [0..1]
+ smoothingFactor :Real [0..1]
+ standardDeviation :Real [0..1]
+ supplierCalculatedDataQuality :Real [0..1]
+ time :DateTime [0..1]
+ affectedLocation :LocationReference [0..1]
«FeatureType»
BasicDataValue::TrafficStatusValue
+ trafficStatus :TrafficStatusEnum [0..1]
+ trafficTrendType :TrafficTrendTypeEnum [0..1]
«FeatureType»
BasicDataValue::TravelTimeValue
+ travelTime :Time [0..1]
+ travelTimeTrendType :TravelTimeTrendTypeEnum [0..1]
+ travelTimeType :TravelTimeTypeEnum [0..1]
+ freeFlowSpeed :Speed [0..1]
+ freeFlowTravelTime :Time [0..1]
+ normallyExpectedTravelTime :Time [0..1]
+ vehicleType :VehicleTypeEnum [0..*]
«FeatureType»
BasicDataValue::EnvironmentValue

Figure 10 Measured data publication

This class constitutes the basis for the specific types of measures (TrafficStatusValue, TravelTimeValue, TrafficValue, WeatherValue and *supplementaryPositionalDescription*). The image bellow shows the most relevant types of measures managed.

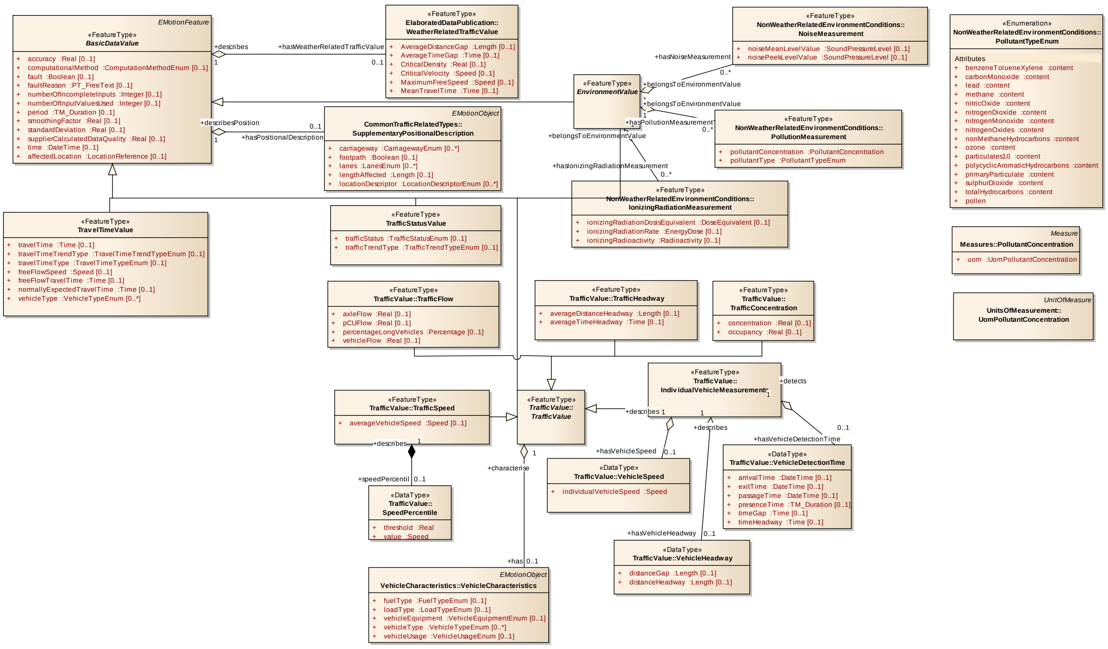

EMotionFeature
«FeatureType»
BasicDataValue
+ accuracy :Real [0..1]
+ computationalMethod :ComputationMethodEnum [0..1]
+ fault :Boolean [0..1]
+ faultReason :PT_FreeText [0..1]
+ numberOfIncompleteInputs :Integer [0..1]
+ numberOfInputValuesUsed :Integer [0..1]
+ period :TM_Duration [0..1]
+ smoothingFactor :Real [0..1]
+ standardDeviation :Real [0..1]
+ supplierCalculatedDataQuality :Real [0..1]
+ time :DateTime [0..1]
+ affectedLocation :LocationReference [0..1]
+describes
1
+hasWeatherRelatedTrafficValue
0..1
«FeatureType»
ElaboratedDataPublication::
WeatherRelatedTrafficValue
+ AverageDistanceGap :Length [0..1]
+ AverageTimeGap :Time [0..1]
+ CriticalDensity :Real [0..1]
+ CriticalVelocity :Speed [0..1]
+ MaximumFreeSpeed :Speed [0..1]
+ MeanTravelTime :Time [0..1]
+describesPosition
1
+hasPositionalDescription
0..1
EMotionObject
CommonTrafficRelatedTypes::
SupplementaryPositionalDescription
+ carriageway :CarriagewayEnum [0..*]
+ footpath :Boolean [0..1]
+ lanes :LanesEnum [0..*]
+ lengthAffected :Length [0..1]
+ locationDescriptor :LocationDescriptorEnum [0..*]
«FeatureType»
EnvironmentValue
+hasNoiseMeasurement
0..*
+belongsToEnvironmentValue
1
+belongsToEnvironmentValue
1
+hasPollutionMeasurement
0..*
+belongsToEnvironmentValue
+hasIonizingRadiationMeasurement
0..*
«FeatureType»
NonWeatherRelatedEnvironmentConditions::
NoiseMeasurement
+ noiseMeanLevelValue :SoundPressureLevel [0..1]
+ noisePeekLevelValue :SoundPressureLevel [0..1]
«FeatureType»
NonWeatherRelatedEnvironmentConditions::
PollutionMeasurement
+ pollutantConcentration :PollutantConcentration
+ pollutantType :PollutantTypeEnum
«FeatureType»
NonWeatherRelatedEnvironmentConditions::
IonizingRadiationMeasurement
+ ionizingRadiationDosisEquivalent :DoseEquivalent [0..1]
+ ionizingRadiationRate :EnergyDose [0..1]
+ ionizingRadioactivity :Radioactivity [0..1]
«Enumeration»
NonWeatherRelatedEnvironmentConditions::
PollutantTypeEnum
Attributes
+ benzeneTolueneXylene :content
+ carbonMonoxide :content
+ lead :content
+ methane :content
+ nitricOxide :content
+ nitrogenDioxide :content
+ nitrogenMonoxide :content
+ nitrogenOxides :content
+ nonMethaneHydrocarbons :content
+ ozone :content
+ particulates10 :content
+ polycyclicAromaticHydrocarbons :content
+ primaryParticulate :content
+ sulphurDioxide :content
+ totalHydrocarbons :content
+ pollen
Measure
Measures::PollutantConcentration
+ uom :UomPollutantConcentration
UnitOfMeasure
UnitsOfMeasurement::
UomPollutantConcentration
«FeatureType»
TravelTimeValue
+ travelTime :Time [0..1]
+ travelTimeTrendType :TravelTimeTrendTypeEnum [0..1]
+ travelTimeType :TravelTimeTypeEnum [0..1]
+ freeFlowSpeed :Speed [0..1]
+ freeFlowTravelTime :Time [0..1]
+ normallyExpectedTravelTime :Time [0..1]
+ vehicleType :VehicleTypeEnum [0..*]
«FeatureType»
TrafficStatusValue
+ trafficStatus :TrafficStatusEnum [0..1]
+ trafficTrendType :TrafficTrendTypeEnum [0..1]
«FeatureType»
TrafficValue::TrafficFlow
+ axleFlow :Real [0..1]
+ pCUFlow :Real [0..1]
+ percentageLongVehicles :Percentage [0..1]
+ vehicleFlow :Real [0..1]
«FeatureType»
TrafficValue::TrafficHeadway
+ averageDistanceHeadway :Length [0..1]
+ averageTimeHeadway :Time [0..1]
«FeatureType»
TrafficValue::
TrafficConcentration
+ concentration :Real [0..1]
+ occupancy :Real [0..1]
«FeatureType»
TrafficValue::TrafficSpeed
+ averageVehicleSpeed :Speed [0..1]
+describes
1
+speedPercentil
0..1
«DataType»
TrafficValue::
SpeedPercentile
+ threshold :Real
+ value :Speed
«FeatureType»
TrafficValue::
TrafficValue
1
+characterise
+has
0..1
EMotionObject
VehicleCharacteristics::VehicleCharacteristics
+ fuelType :FuelTypeEnum [0..1]
+ loadType :LoadTypeEnum [0..1]
+ vehicleEquipment :VehicleEquipmentEnum [0..1]
+ vehicleType :VehicleTypeEnum [0..*]
+ vehicleUsage :VehicleUsageEnum [0..1]
«FeatureType»
TrafficValue::
IndividualVehicleMeasurements
+detects
1
+describes
1
1
+describes
0..1
+hasVehicleDetectionTime
+hasVehicleSpeed
0..1
«DataType»
TrafficValue::VehicleSpeed
+ individualVehicleSpeed :Speed
«DataType»
TrafficValue::VehicleDetectionTime
+ arrivalTime :DateTime [0..1]
+ exitTime :DateTime [0..1]
+ passageTime :DateTime [0..1]
+ presenceTime :TM_Duration [0..1]
+ timeGap :Time [0..1]
+ timeHeadway :Time [0..1]
+hasVehicleHeadway
0..1
«DataType»
TrafficValue::VehicleHeadway
+ distanceGap :Length [0..1]
+ distanceHeadway :Length [0..1]

Figure 11 Data values

On the other hand, *ElaboratedData* provides a representation for the main **aggregated measures** per road segment and temporal interval (this time at Traffic Centre level, that is integrating measures from the different deployed equipment): ***TravelTimes*** (elaborated time, free flow time, normally expected time), ***Traffic status*** that identifies five different values (free flow, heavy, congested, impossible, unknown) and ***Traffic values***: flow, speed, headway, concentration.

To specify the **aggregation criteria**, *ElaboratedData,* defines specific classes: *BasicDataValue* (describing **accuracy**, **applied method**, standard deviation and temporal/geographical, data quality), ***Validity*** *(*defining a time period intervals (*Period, TimePeriodOfDay, DayweekMonth*) by means of interval data definition, intersection and union operations) and ***SourceInformation*** (identification, location and mainly reliability). All these characteristics are relevant for the data fusion algorithm.

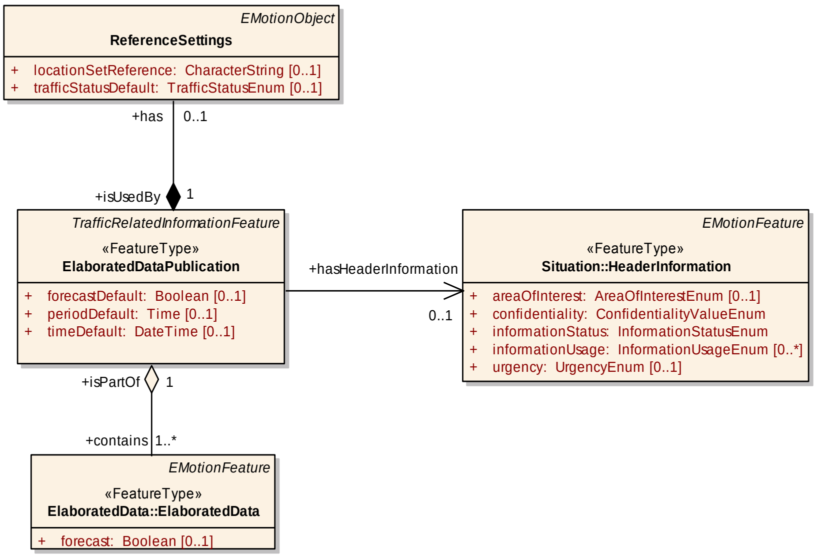

EMotionObject
ReferenceSettings
+ locationSetReference: CharacterString [0..1]
+ trafficStatusDefault: TrafficStatusEnum [0..1]
+has
0..1
+isUsedBy
1
TrafficRelatedInformationFeature
«FeatureType»
ElaboratedDataPublication
+ forecastDefault: Boolean [0..1]
+ periodDefault: Time [0..1]
+ timeDefault: DateTime [0..1]
+hasHeaderInformation
0..1
EMotionFeature
«FeatureType»
Situation::HeaderInformation
+ areaOfInterest: AreaOfInterestEnum [0..1]
+ confidentiality: ConfidentialityValueEnum
+ informationStatus: InformationStatusEnum
+ informationUsage: InformationUsageEnum [0..*]
+ urgency: UrgencyEnum [0..1]
+isPartOf
1
+contains
1..*
EMotionFeature
«FeatureType»
ElaboratedData::ElaboratedData
+ forecast: Boolean [0..1]

Figure 12 Elaborated data publication

Incident Data

Based on identified data requirements, the incident data model already included in In-Time already covers MoveUs data necessities.

Since eMotion/In-Time information model has been developed following the encoding rules defined in the ISO 19100 series of international standards, some changes had been applied to the original DATEX 2 model, mainly of formal nature. As defined in eMotion and In-Time documentation, the changes refer especially to:

- Stereotypes of the classes,
- Data types of the attributes,
- Addition of role names,
- Addition of a few attributes,
- Variation of the Location Reference and
- Adaptation of enumerations.

The incidents are *TrafficRelatedSituation* objects having 4 main categories:

- *TrafficElement* (road or traffic related event).
- An *OperatorAction*.
- A *NonRoadEventInformation*.
- Weather and environmental events affecting road users.

This is, *TrafficRelatedSituation* refers to incidents and accidents, congestions, weather and environmental events, road works and road closures for specific points on the road, routes or administrative areas and can contain and be described by several concepts as shown in the next plots. Each incident or message represents a *SituationRecord*. A *SituationRecord* is one element of a *Situation* and is characterised by values at a given time, defining one version of this element. It is important to remark that a *TrafficRelatedSituation* is associated with a

LocationReference and it should then be always possible to identify the traffic disturbance on the network.

On the one hand, and as shown in the next plot, **activities**, **accidents**, **abnormal traffic states** and **obstructions** (general obstructions, environmental obstructions and vehicle obstructions) are already covered as different *TrafficElements* to be recorded (*TrafficRelatedSituationRecorded*). On the other hand, the actions or activities undertaken by the operator (*OperatorAction*) are also included distinguishing between **maintenance** (*MaintenanceWorks*) and **construction activities** (*ConstructionWorks*). Generally speaking, they are actions implemented to prevent or help correct dangerous or poor driving conditions, including maintenance of the road infra-structure.

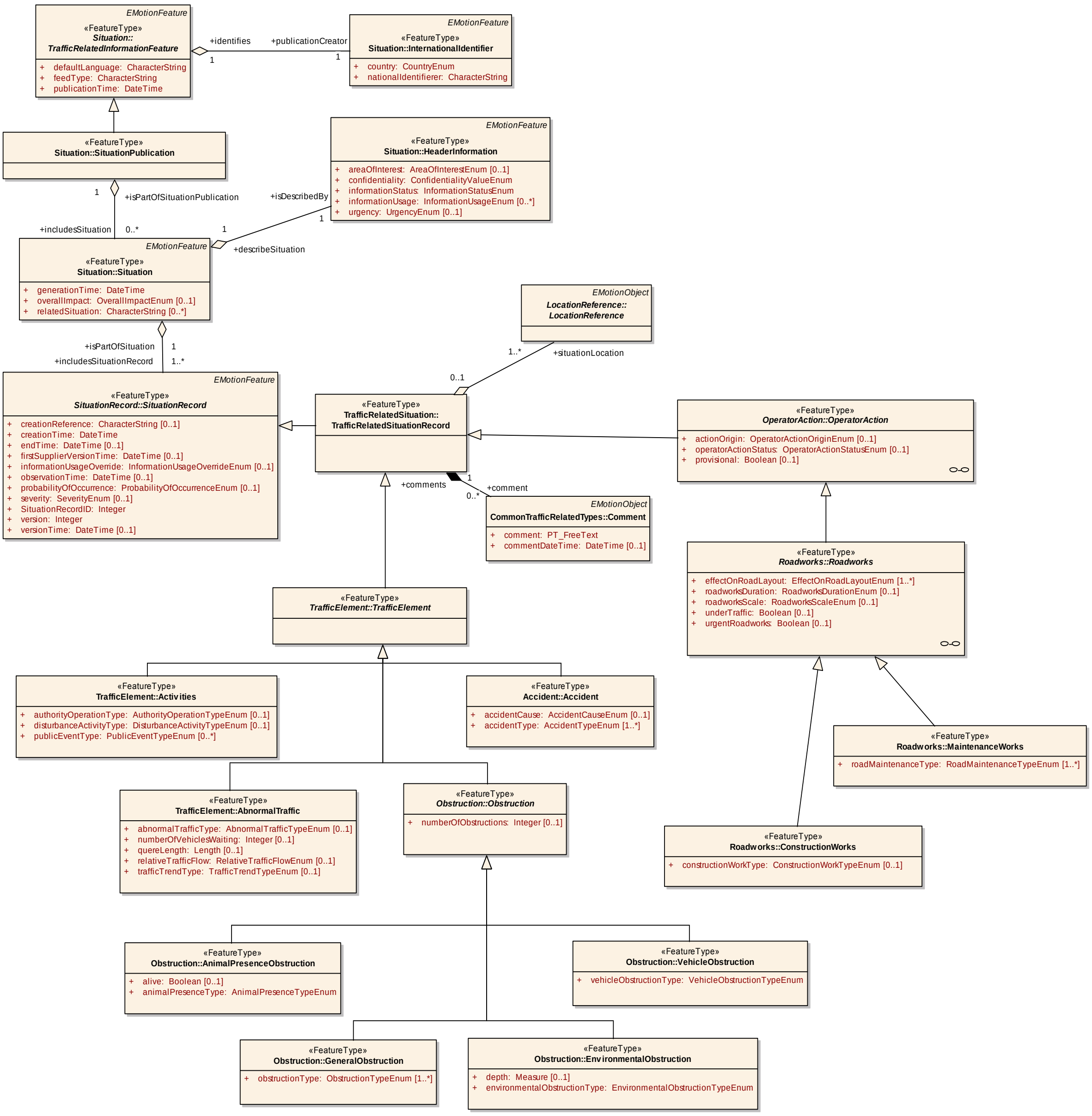

EMotionFeature
«FeatureType»
Situation::TrafficRelatedInformationFeature
+ defaultLanguage: CharacterString
+ feedType: CharacterString
+ publicationTime: DateTime
+identifies
1
+publicationCreator
1
EMotionFeature
«FeatureType»
Situation::InternationalIdentifier
+ country: CountryEnum
+ nationalIdentifierer: CharacterString
«FeatureType»
Situation::SituationPublication
EMotionFeature
«FeatureType»
Situation::HeaderInformation
+ areaOfInterest: AreaOfInterestEnum [0..1]
+ confidentiality: ConfidentialityValueEnum
+ informationStatus: InformationStatusEnum
+ informationUsage: InformationUsageEnum [0..*]
+ urgency: UrgencyEnum [0..1]
1
+isPartOfSituationPublication
+isDescribedBy
1
+includesSituation
0..*
1
+describeSituation
EMotionFeature
«FeatureType»
Situation::Situation
+ generationTime: DateTime
+ overallImpact: OverallImpactEnum [0..1]
+ relatedSituation: CharacterString [0..*]
EMotionObject
LocationReference::
LocationReference
+isPartOfSituation
1
+includesSituationRecord
1..*
1..*
+situationLocation
0..1
EMotionFeature
«FeatureType»
SituationRecord::SituationRecord
+ creationReference: CharacterString [0..1]
+ creationTime: DateTime
+ endTime: DateTime [0..1]
+ firstSupplierVersionTime: DateTime [0..1]
+ informationUsageOverride: InformationUsageOverrideEnum [0..1]
+ observationTime: DateTime [0..1]
+ probabilityOfOccurrence: ProbabilityOfOccurrenceEnum [0..1]
+ severity: SeverityEnum [0..1]
+ SituationRecordID: Integer
+ version: Integer
+ versionTime: DateTime [0..1]
«FeatureType»
TrafficRelatedSituation::
TrafficRelatedSituationRecord
«FeatureType»
OperatorAction::OperatorAction
+ actionOrigin: OperatorActionOriginEnum [0..1]
+ operatorActionStatus: OperatorActionStatusEnum [0..1]
+ provisional: Boolean [0..1]
+comments
1
+comment
0..*
EMotionObject
CommonTrafficRelatedTypes::Comment
+ comment: PT_FreeText
+ commentDateTime: DateTime [0..1]
«FeatureType»
Roadworks::Roadworks
+ effectOnRoadLayout: EffectOnRoadLayoutEnum [1..*]
+ roadworksDuration: RoadworksDurationEnum [0..1]
+ roadworksScale: RoadworksScaleEnum [0..1]
+ underTraffic: Boolean [0..1]
+ urgentRoadworks: Boolean [0..1]
«FeatureType»
TrafficElement::TrafficElement
«FeatureType»
TrafficElement::Activities
+ authorityOperationType: AuthorityOperationTypeEnum [0..1]
+ disturbanceActivityType: DisturbanceActivityTypeEnum [0..1]
+ publicEventType: PublicEventTypeEnum [0..*]
«FeatureType»
Accident::Accident
+ accidentCause: AccidentCauseEnum [0..1]
+ accidentType: AccidentTypeEnum [1..*]
«FeatureType»
Roadworks::MaintenanceWorks
+ roadMaintenanceType: RoadMaintenanceTypeEnum [1..*]
«FeatureType»
TrafficElement::AbnormalTraffic
+ abnormalTrafficType: AbnormalTrafficTypeEnum [0..1]
+ numberOfVehiclesWaiting: Integer [0..1]
+ quereLength: Length [0..1]
+ relativeTrafficFlow: RelativeTrafficFlowEnum [0..1]
+ trafficTrendType: TrafficTrendTypeEnum [0..1]
«FeatureType»
Obstruction::Obstruction
+ numberOfObstructions: Integer [0..1]
«FeatureType»
Roadworks::ConstructionWorks
+ constructionWorkType: ConstructionWorkTypeEnum [0..1]
«FeatureType»
Obstruction::AnimalPresenceObstruction
+ alive: Boolean [0..1]
+ animalPresenceType: AnimalPresenceTypeEnum
«FeatureType»
Obstruction::VehicleObstruction
+ vehicleObstructionType: VehicleObstructionTypeEnum
«FeatureType»
Obstruction::GeneralObstruction
+ obstructionType: ObstructionTypeEnum [1..*]
«FeatureType»
Obstruction::EnvironmentalObstruction
+ depth: Measure [0..1]
+ environmentalObstructionType: EnvironmentalObstructionTypeEnum

Figure 13 Incidence Related Data

On the other hand, In-Time simplified data model also covers weather and environmental events affecting road users (*RoadWeatherAndEnviromentEvent*). Concretely, it defines road weather events according to a selection of the TMC Event Code List (next plot).

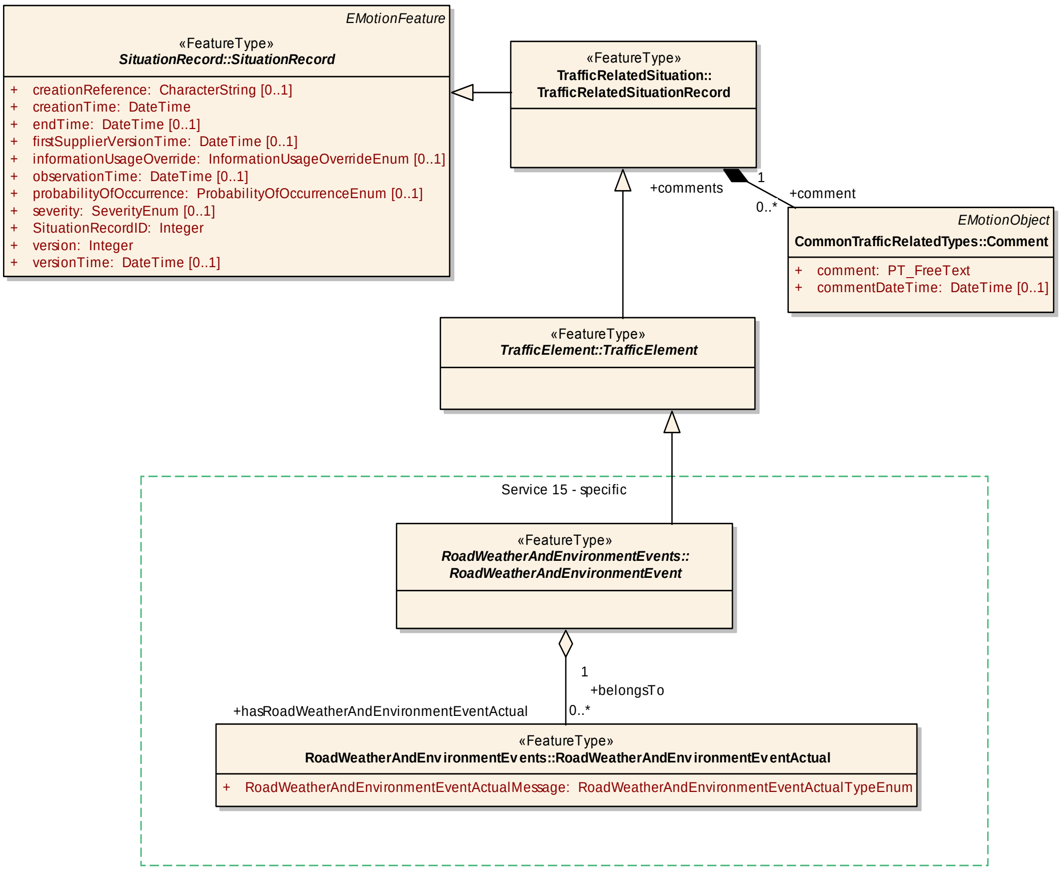

EMotionFeature
«FeatureType»
SituationRecord::SituationRecord
+ creationReference: CharacterString [0..1]
+ creationTime: DateTime
+ endTime: DateTime [0..1]
+ firstSupplierVersionTime: DateTime [0..1]
+ informationUsageOverride: InformationUsageOverrideEnum [0..1]
+ observationTime: DateTime [0..1]
+ probabilityOfOccurence: ProbabilityOfOccurrenceEnum [0..1]
+ severity: SeverityEnum [0..1]
+ SituationRecordID: Integer
+ version: Integer
+ versionTime: DateTime [0..1]
«FeatureType»
TrafficRelatedSituation::
TrafficRelatedSituationRecord
+comments
1
+comment
0..*
EMotionObject
CommonTrafficRelatedTypes::Comment
+ comment: PT_FreeText
+ commentDateTime: DateTime [0..1]
«FeatureType»
TrafficElement::TrafficElement
Service 15 - specific
«FeatureType»
RoadWeatherAndEnvironmentEvents::
RoadWeatherAndEnvironmentEvent
1
+belongsTo
+hasRoadWeatherAndEnvironmentEventActual
0..*
«FeatureType»
RoadWeatherAndEnvironmentEvents::RoadWeatherAndEnvironmentEventActual
+ RoadWeatherAndEnvironmentEventActualMessage: RoadWeatherAndEnvironmentEventActualTypeEnum

Figure 14 Dynamic Weather Model

Finally, In-Time data model has an special service so as to included incidences having no relation with road events (*NonRoadEventInformation* in next plot) but which may affect drivers behaviour and therefore the traffic flow: **service disruptions** relevant to road users (e.g. petrol shortage or rest area closed), availability of **transit services** and information relating to their departures, limiting to transit services which are of direct relevance to road users (e.g. connecting rail or ferry service) and **car parks**.

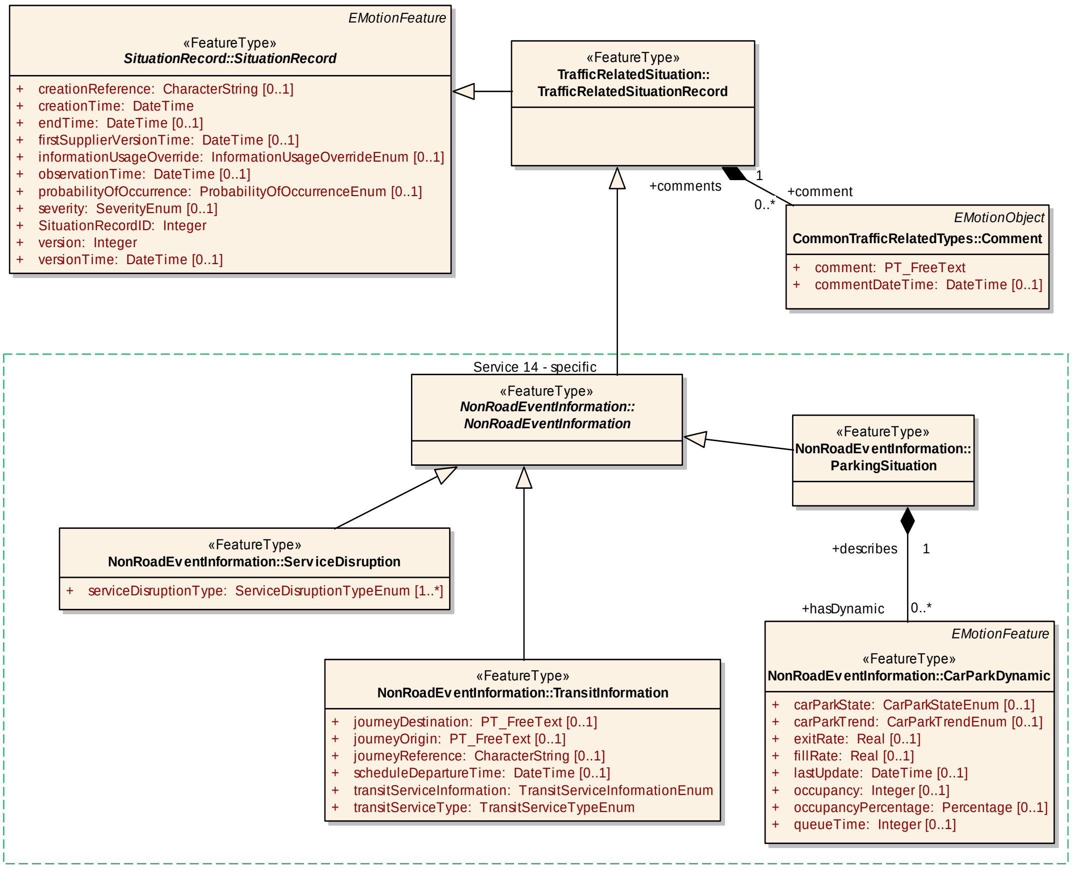

EMotionFeature
«FeatureType»
SituationRecord::SituationRecord
+ creationReference: CharacterString [0..1]
+ creationTime: DateTime
+ endTime: DateTime [0..1]
+ firstSupplierVersionTime: DateTime [0..1]
+ informationUsageOverride: InformationUsageOverrideEnum [0..1]
+ observationTime: DateTime [0..1]
+ probabilityOfOccurrence: ProbabilityOfOccurrenceEnum [0..1]
+ severity: SeverityEnum [0..1]
+ SituationRecordID: Integer
+ version: Integer
+ versionTime: DateTime [0..1]
«FeatureType»
TrafficRelatedSituation::
TrafficRelatedSituationRecord
+comments
1
0..*
+comment
EMotionObject
CommonTrafficRelatedTypes::Comment
+ comment: PT_FreeText
+ commentDateTime: DateTime [0..1]
Service 14 - specific
«FeatureType»
NonRoadEventInformation::
NonRoadEventInformation
«FeatureType»
NonRoadEventInformation::
ParkingSituation
«FeatureType»
NonRoadEventInformation::ServiceDisruption
+ serviceDisruptionType: ServiceDisruptionTypeEnum [1..*]
+describes
1
+hasDynamic
0..*
EMotionFeature
«FeatureType»
NonRoadEventInformation::CarParkDynamic
+ carParkState: CarParkStateEnum [0..1]
+ carParkTrend: CarParkTrendEnum [0..1]
+ exitRate: Real [0..1]
+ fillRate: Real [0..1]
+ lastUpdate: DateTime [0..1]
+ occupancy: Integer [0..1]
+ occupancyPercentage: Percentage [0..1]
+ queueTime: Integer [0..1]
«FeatureType»
NonRoadEventInformation::TransitInformation
+ journeyDestination: PT_FreeText [0..1]
+ journeyOrigin: PT_FreeText [0..1]
+ journeyReference: CharacterString [0..1]
+ scheduleDepartureTime: DateTime [0..1]
+ transitServiceInformation: TransitServiceInformationEnum
+ transitServiceType: TransitServiceTypeEnum

Figure 15 Dynamic Traffic Event Information

Urban Car Park / Bike Sharing Data

In-Time simplified data model defines data structures for static and dynamic information related with car parking (next plot). The *ParkingPoints* are identified as *PointOfInterest* places specifying the location and the category among others concepts. Each parking has static information as tariffs and a complete description of its facilities (e.g. toilets for the disabled available, total capacity, user types, etc.) whereas dynamic information is related to the real time occupancy (e.g. fill rate, queue time, etc.). It is important to remark, that the *CarparkDynamic* information is also linked to the *NonRoadEventInformation* in the incidents (as seen in previous section).

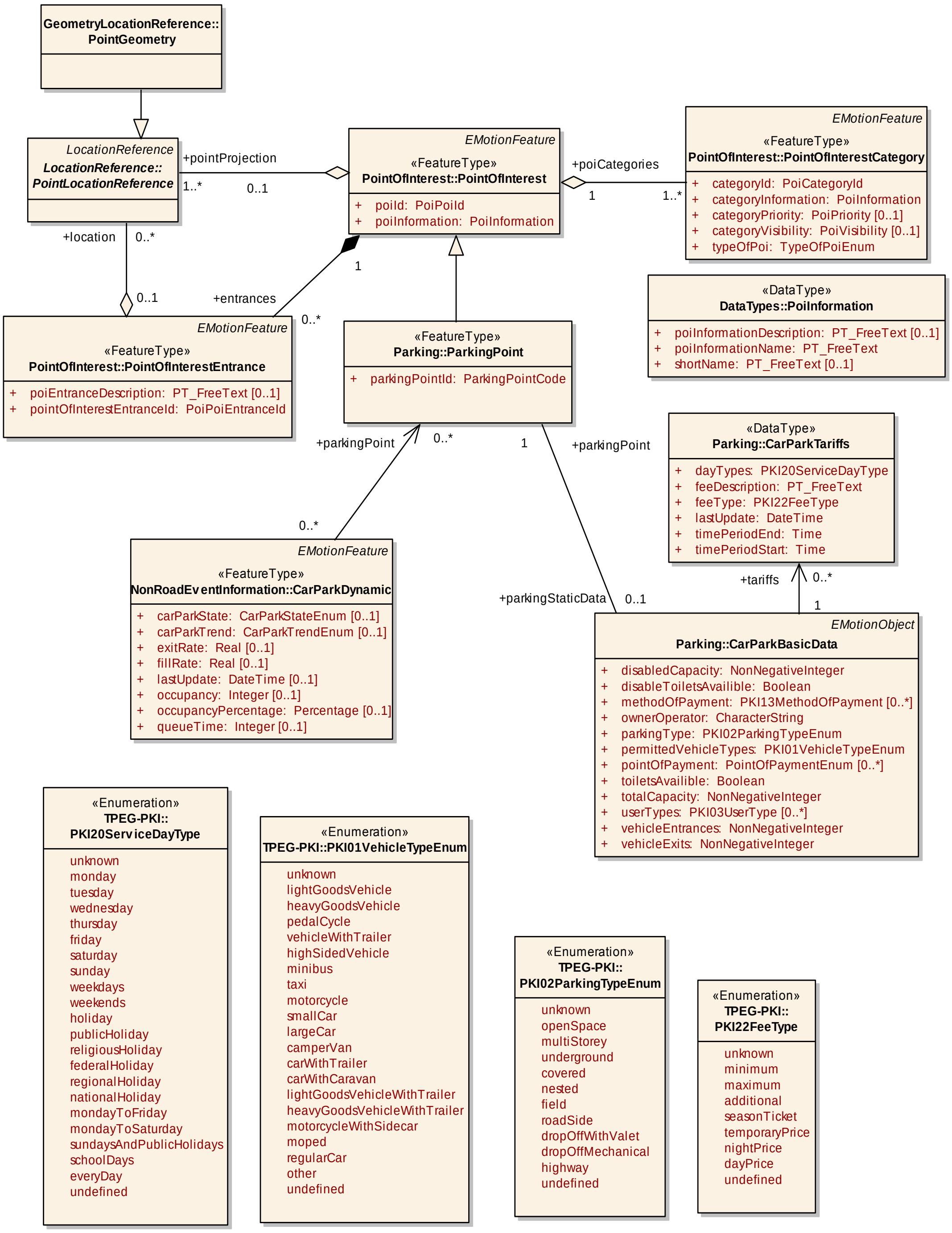

GeometryLocationReference::
PointGeometry
LocationReference
LocationReference::
PointLocationReference
+pointProjection
1..*
0..1
+location
0..*
0..1
+entrances
1
0..*
EMotionFeature
«FeatureType»
PointOfInterest::PointOfInterest
+ poiId: PoiPoiId
+ poiInformation: PoiInformation
+poiCategories
1
1..*
EMotionFeature
«FeatureType»
PointOfInterest::PointOfInterestCategory
+ categoryId: PoiCategoryId
+ categoryInformation: PoiInformation
+ categoryPriority: PoiPriority [0..1]
+ categoryVisibility: PoiVisibility [0..1]
+ typeOfPoi: TypeOfPoiEnum
«DataType»
DataTypes::PoiInformation
+ poiInformationDescription: PT_FreeText [0..1]
+ poiInformationName: PT_FreeText
+ shortName: PT_FreeText [0..1]
EMotionFeature
«FeatureType»
PointOfInterest::PointOfInterestEntrance
+ poiEntranceDescription: PT_FreeText [0..1]
+ pointOfInterestEntranceId: PoiPoiEntranceId
«FeatureType»
Parking::ParkingPoint
+ parkingPointId: ParkingPointCode
+parkingPoint
0..*
1
+parkingPoint
«DataType»
Parking::CarParkTariffs
+ dayTypes: PKI20ServiceDayType
+ feeDescription: PT_FreeText
+ feeType: PKI22FeeType
+ lastUpdate: DateTime
+ timePeriodEnd: Time
+ timePeriodStart: Time
0..*
EMotionFeature
«FeatureType»
NonRoadEventInformation::CarParkDynamic
+ carParkState: CarParkStateEnum [0..1]
+ carParkTrend: CarParkTrendEnum [0..1]
+ exitRate: Real [0..1]
+ fillRate: Real [0..1]
+ lastUpdate: DateTime [0..1]
+ occupancy: Integer [0..1]
+ occupancyPercentage: Percentage [0..1]
+ queueTime: Integer [0..1]
+parkingStaticData
0..1
+tariffs
0..*
1
EMotionObject
Parking::CarParkBasicData
+ disabledCapacity: NonNegativeInteger
+ disableToiletsAvailible: Boolean
+ methodOfPayment: PKI13MethodOfPayment [0..*]
+ ownerOperator: CharacterString
+ parkingType: PKI02ParkingTypeEnum
+ permittedVehicleTypes: PKI01VehicleTypeEnum
+ pointOfPayment: PointOfPaymentEnum [0..*]
+ toiletsAvailible: Boolean
+ totalCapacity: NonNegativeInteger
+ userTypes: PKI03UserType [0..*]
+ vehicleEntrances: NonNegativeInteger
+ vehicleExits: NonNegativeInteger
«Enumeration»
TPEG-PKI::
PKI20ServiceDayType
unknown
monday
tuesday
wednesday
thursday
friday
saturday
sunday
weekdays
weekends
holiday
publicHoliday
religiousHoliday
federalHoliday
regionalHoliday
nationalHoliday
mondayToFriday
mondayToSaturday
sundaysAndPublicHolidays
schoolDays
everyDay
undefined
«Enumeration»
TPEG-PKI::PKI01VehicleTypeEnum
unknown
lightGoodsVehicle
heavyGoodsVehicle
pedalCycle
vehicleWithTrailer
highSidedVehicle
minibus
taxi
motorcycle
smallCar
largeCar
camperVan
carWithTrailer
carWithCaravan
lightGoodsVehicleWithTrailer
heavyGoodsVehicleWithTrailer
motorcycleWithSidecar
moped
regularCar
other
undefined
«Enumeration»
TPEG-PKI::
PKI02ParkingTypeEnum
unknown
openSpace
multiStorey
underground
covered
nested
field
roadSide
dropOffWithValet
dropOffMechanical
highway
undefined
«Enumeration»
TPEG-PKI::
PKI22FeeType
unknown
minimum
maximum
additional
seasonTicket
temporaryPrice
nightPrice
dayPrice
undefined

Figure 16 Dynamic Parking Model

3.2.2 Extensions

Urban Road Static Data && Road Static Data

An extension of In-Time data model is needed in order to cover all information requirements for Madrid use-cases. This extension is mainly related to the necessity of a deep description of each intersection (number and type of lines, possible movements, etc.) and the equipment installed around them, concretely Bluetooth readers.

The SAE J2735 "Dedicated short Range Communications Message Set Dictionary" standard [5] has been identified as starting point for the static and dynamic information related with intersections. The aim of this standard is to specify all messages, data frames and data elements used for both Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) exchanges.

Based on this standard and taking into account the requirements identified for Madrid use-cases, a selection of the messages and data frames has been done in order to cover both, static and dynamic information related with intersections. There are 4 messages defined in this standard to support intersection mapping and signal phase and timing data:

- **Signal Phase and Timing Message (SPAT)**: Relates the current intersection signal light phases [5].
- **Map Data (MAP):** Relates the Physical Geometry of the intersection [5].
- **Signal Request Message (SRM):** Requests preemption or priority services [5].
- **Signal Status Messages (SSM):** Related to the internal state of the signal controller [5].

Map Data message has been selected to include the static information related with intersections while the other 3 messages are going to be used for dynamic information (next subsection).

The following plot (Figure 17) shows the *IntersectionInfo* class where a *Map Data message* and geographical information (*Circle* and *ValidRegion*) are combined in order to cover Madrid uses cases requirements.

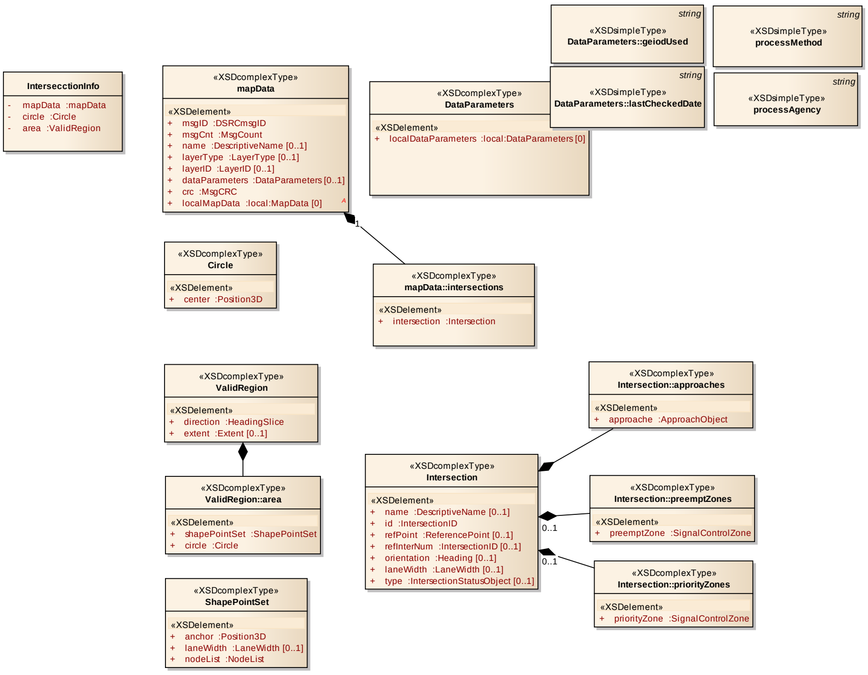

IntersecctionInfo
- mapData :mapData
- circle :Circle
- area :ValidRegion
«XSDcomplexType» mapData
«XSDelement»
+ msgID :DSRCmsgID
+ msgCnt :MsgCount
+ name :DescriptiveName [0..1]
+ layerType :LayerType [0..1]
+ layerID :LayerID [0..1]
+ dataParameters :DataParameters [0..1]
+ crc :MsgCRC
+ localMapData :local:MapData [0]
A
1
«XSDcomplexType» DataParameters
«XSDelement»
+ localDataParameters :local:DataParameters [0]
string
«XSDsimpleType» DataParameters::geiodUsed
string
«XSDsimpleType» processMethod
string
«XSDsimpleType» DataParameters::lastCheckedDate
string
«XSDsimpleType» processAgency
«XSDcomplexType» Circle
«XSDelement»
+ center :Position3D
«XSDcomplexType» mapData::intersections
«XSDelement»
+ intersection :Intersection
«XSDcomplexType» ValidRegion
«XSDelement»
+ direction :HeadingSlice
+ extent :Extent [0..1]
«XSDcomplexType» ValidRegion::area
«XSDelement»
+ shapePointSet :ShapePointSet
+ circle :Circle
«XSDcomplexType» ShapePointSet
«XSDelement»
+ anchor :Position3D
+ laneWidth :LaneWidth [0..1]
+ nodeList :NodeList
«XSDcomplexType» Intersection
«XSDelement»
+ name :DescriptiveName [0..1]
+ id :IntersectionID
+ refPoint :ReferencePoint [0..1]
+ refInterNum :IntersectionID [0..1]
+ orientation :Heading [0..1]
+ laneWidth :LaneWidth [0..1]
+ type :IntersectionStatusObject [0..1]
«XSDcomplexType» Intersection::approaches
«XSDelement»
+ approache :ApproachObject
«XSDcomplexType» Intersection::preemptZones
«XSDelement»
+ preemptZone :SignalControlZone
0..1
«XSDcomplexType» Intersection::priorityZones
«XSDelement»
+ priorityZone :SignalControlZone
0..1

Figure 17 Intersection information static model (partial view)

- ***Circle:*** to define a circle centered at a given point and extended to the given radius. It is typically used to describe the location of signs so that the receiving vehicle can determine if the sign applies to them and their current path [5].
- ***ValidRegion*** is used to describe one or more geographic locations to which a message (typically road signs or advisories of some sort) is applied or considered valid [5].

Going deeply into the ***Map Data message***, it contains all unchanging information of one or more intersections in the *intersection* data frame. This message, not only describes the lane geometry paths and the allowed movements in each lane but also additional information related with barriers, pedestrian walks, etc. is provided.

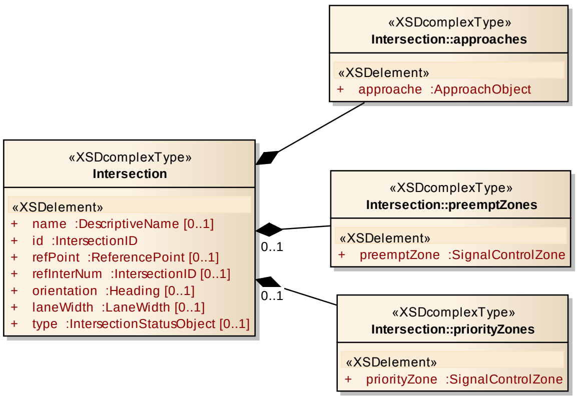

«XSDcomplexType»
Intersection::approaches
«XSDelement»
+ approache :ApproachObject
«XSDcomplexType»
Intersection
«XSDelement»
+ name :DescriptiveName [0..1]
+ id :IntersectionID
+ refPoint :ReferencePoint [0..1]
+ refInterNum :IntersectionID [0..1]
+ orientation :Heading [0..1]
+ laneWidth :LaneWidth [0..1]
+ type :IntersectionStatusObject [0..1]
0..1
0..1
«XSDcomplexType»
Intersection::preemptZones
«XSDelement»
+ preemptZone :SignalControlZone
«XSDcomplexType»
Intersection::priorityZones
«XSDelement»
+ priorityZone :SignalControlZone

Figure 18 Intersection data frame

A ***Map Data message*** can contain a sequence of intersections (*intersections* attribute of *Intersection* type). In this standard, an intersection is a collection of approaches while an approach (Figure 18) is a collection of related lanes. The *ApproachObject* structure (Figure 19) allows arbitrary groupings of lanes being these lanes both driven vehicle use type lanes as well as other lane types defined by the standard: "pedestrian" lanes (cross walks) and "special" lanes for shared lanes, rail track and other multi-modal uses, and "barriers" for various dividers. Approach lanes are also divided into approach (ingress, incoming) and egress (outgoing) lanes.

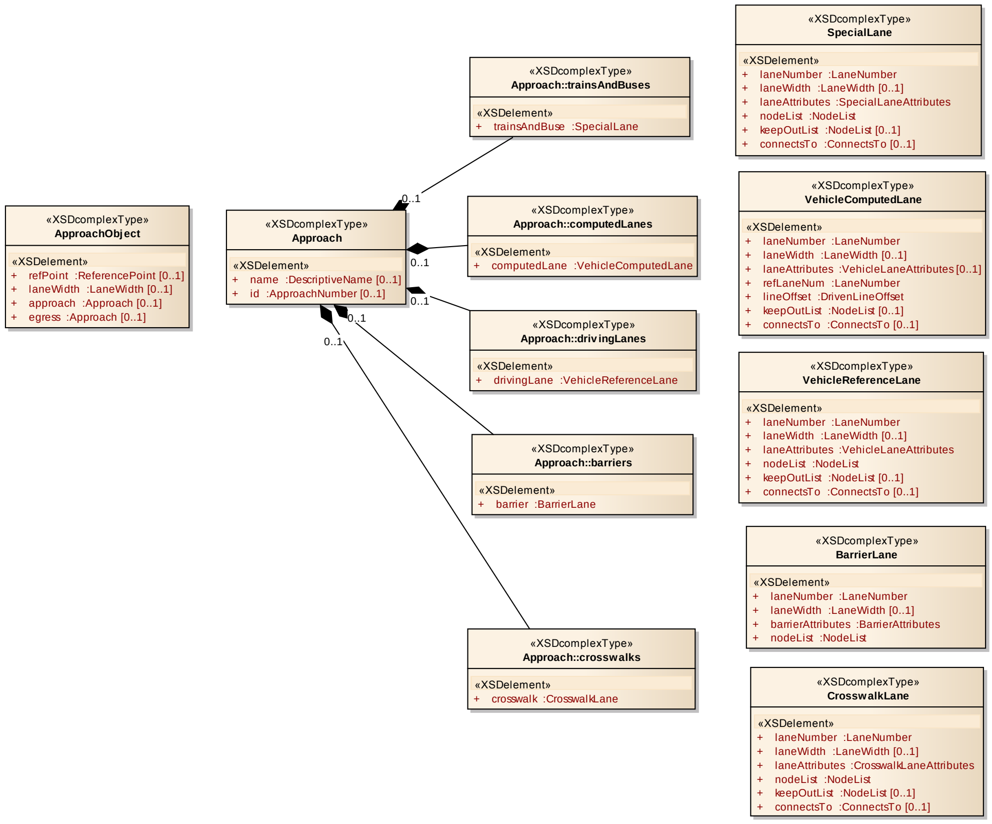

«XSDcomplexType»
ApproachObject
«XSDelement»
+ refPoint :ReferencePoint [0..1]
+ laneWidth :LaneWidth [0..1]
+ approach :Approach [0..1]
+ egress :Approach [0..1]
«XSDcomplexType»
Approach
«XSDelement»
+ name :DescriptiveName [0..1]
+ id :ApproachNumber [0..1]
0..1
0..1
0..1
0..1
0..1
«XSDcomplexType»
Approach::trainsAndBuses
«XSDelement»
+ trainsAndBuse :SpecialLane
«XSDcomplexType»
Approach::computedLanes
«XSDelement»
+ computedLane :VehicleComputedLane
«XSDcomplexType»
Approach::drivingLanes
«XSDelement»
+ drivingLane :VehicleReferenceLane
«XSDcomplexType»
Approach::barriers
«XSDelement»
+ barrier :BarrierLane
«XSDcomplexType»
Approach::crosswalks
«XSDelement»
+ crosswalk :CrosswalkLane
«XSDcomplexType»
SpecialLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :SpecialLaneAttributes
+ nodeList :NodeList
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]
«XSDcomplexType»
VehicleComputedLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :VehicleLaneAttributes [0..1]
+ refLaneNum :LaneNumber
+ lineOffset :DrivenLineOffset
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]
«XSDcomplexType»
VehicleReferenceLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :VehicleLaneAttributes
+ nodeList :NodeList
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]
«XSDcomplexType»
BarrierLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ barrierAttributes :BarrierAttributes
+ nodeList :NodeList
«XSDcomplexType»
CrosswalkLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :CrosswalkLaneAttributes
+ nodeList :NodeList
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]

Figure 19 Approach Object data frame

As additional attributes in *Intersection* (Figure 18) the *preemptionZones* and *priorityZones* are defined in order to provide support for priority and preemption requests at the intersection. These two concepts are used to determine which specific request to make, allowing the mapping of the intersection geometry into specific request zones and values (0~7).

The global model of the static information related with intersections is shown in Figure 20.

For further details see [5].

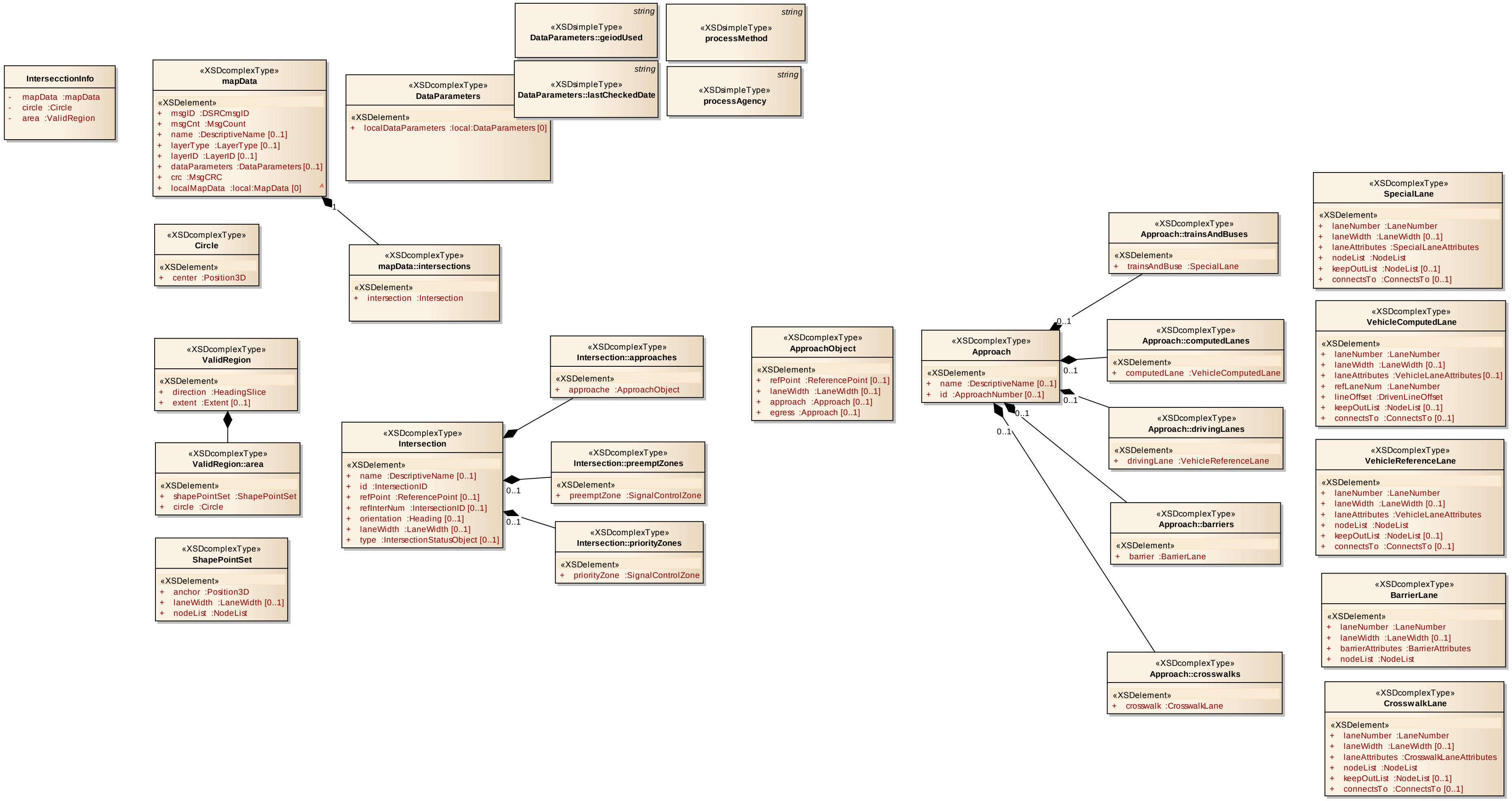

IntersecctionInfo
- mapData :mapData
- circle :Circle
- area :ValidRegion
«XSDcomplexType» mapData
«XSDelement»
+ msgID :DSRCmsgID
+ msgCnt :MsgCount
+ name :DescriptiveName [0..1]
+ layerType :LayerType [0..1]
+ layerID :LayerID [0..1]
+ dataParameters :DataParameters [0..1]
+ crc :MsgCRC
+ localMapData :local:MapData [0]
A
1
«XSDcomplexType» DataParameters
«XSDelement»
+ localDataParameters :local:DataParameters [0]
string
«XSDsimpleType» DataParameters::geiodUsed
string
«XSDsimpleType» DataParameters::lastCheckedDate
string
«XSDsimpleType» processMethod
string
«XSDsimpleType» processAgency
«XSDcomplexType» Circle
«XSDelement»
+ center :Position3D
«XSDcomplexType» mapData::intersections
«XSDelement»
+ intersection :Intersection
«XSDcomplexType» ValidRegion
«XSDelement»
+ direction :HeadingSlice
+ extent :Extent [0..1]
«XSDcomplexType» ValidRegion::area
«XSDelement»
+ shapePointSet :ShapePointSet
+ circle :Circle
«XSDcomplexType» ShapePointSet
«XSDelement»
+ anchor :Position3D
+ laneWidth :LaneWidth [0..1]
+ nodeList :NodeList
«XSDcomplexType» Intersection
«XSDelement»
+ name :DescriptiveName [0..1]
+ id :IntersectionID
+ refPoint :ReferencePoint [0..1]
+ refInterNum :IntersectionID [0..1]
+ orientation :Heading [0..1]
+ laneWidth :LaneWidth [0..1]
+ type :IntersectionStatusObject [0..1]
0..1
0..1
«XSDcomplexType» Intersection::approaches
«XSDelement»
+ approache :ApproachObject
«XSDcomplexType» Intersection::preemptZones
«XSDelement»
+ preemptZone :SignalControlZone
«XSDcomplexType» Intersection::priorityZones
«XSDelement»
+ priorityZone :SignalControlZone
«XSDcomplexType» ApproachObject
«XSDelement»
+ refPoint :ReferencePoint [0..1]
+ laneWidth :LaneWidth [0..1]
+ approach :Approach [0..1]
+ egress :Approach [0..1]
«XSDcomplexType» Approach
«XSDelement»
+ name :DescriptiveName [0..1]
+ id :ApproachNumber [0..1]
0..1
0..1
0..1
0..1
0..1
«XSDcomplexType» Approach::trainsAndBuses
«XSDelement»
+ trainsAndBuse :SpecialLane
«XSDcomplexType» Approach::computedLanes
«XSDelement»
+ computedLane :VehicleComputedLane
«XSDcomplexType» Approach::drivingLanes
«XSDelement»
+ drivingLane :VehicleReferenceLane
«XSDcomplexType» Approach::barriers
«XSDelement»
+ barrier :BarrierLane
«XSDcomplexType» Approach::crosswalks
«XSDelement»
+ crosswalk :CrosswalkLane
«XSDcomplexType» SpecialLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :SpecialLaneAttributes
+ nodeList :NodeList
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]
«XSDcomplexType» VehicleComputedLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :VehicleLaneAttributes [0..1]
+ refLaneNum :LaneNumber
+ lineOffset :DrivenLineOffset
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]
«XSDcomplexType» VehicleReferenceLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :VehicleLaneAttributes
+ nodeList :NodeList
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]
«XSDcomplexType» BarrierLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ barrierAttributes :BarrierAttributes
+ nodeList :NodeList
«XSDcomplexType» CrosswalkLane
«XSDelement»
+ laneNumber :LaneNumber
+ laneWidth :LaneWidth [0..1]
+ laneAttributes :CrosswalkLaneAttributes
+ nodeList :NodeList
+ keepOutList :NodeList [0..1]
+ connectsTo :ConnectsTo [0..1]

Figure 20 Intersection static info global data model

As explained at the beginning of this section, an extension is needed in order to incorporate the Bluetooth equipment installed in each intersection. The new *BTReader* class collects the identifier, the Bluetooth address and the information needed to determine its position in the intersection.

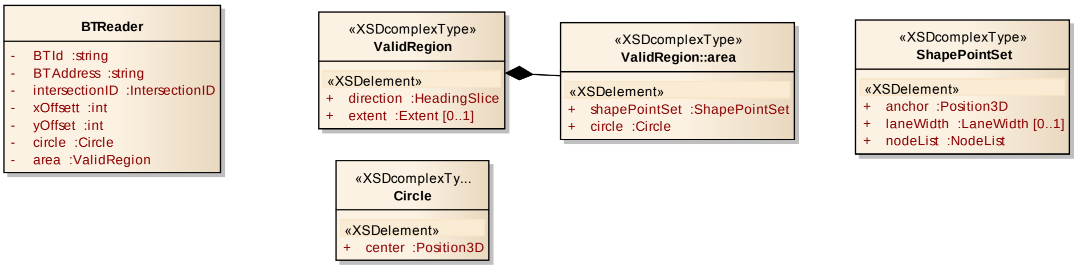


Figure 21 BT readers (BTFix receptors) static model

Urban Traffic Data && Inter-urban Traffic Data && Environmental Data && Road Traffic Prediction Data

Analogously to the previous subsection, an extension is needed in order to cover the information sent by each Bluetooth reader. This information is recorded as an event (*BTDetectionEvent*) linking the detected device with the Bluetooth device of the road infrastructure that is responsible of generating that event (i.e. the Bluetooth reader).

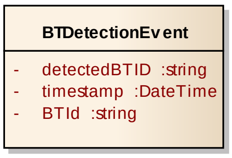


Figure 22 BT readers (BTFix receptors) dynamic model

As explained in previous subsection, the SAE J2735 has been used as starting point in order to define the dynamic information related with intersections. Concretely, these 3 messages have been identified:

- **Signal Phase and Timing Message (SPAT)**: Relates the current intersection signal light phases [5].
- **Signal Request Message (SRM):** Requests preemption or priority services [5].
- **Signal Status Messages (SSM):** Relates the internal state of the signal controller [5].

Starting with the message selected for the smart crossing use case from Madrid, the **Signal Phase and Timing Message (SPAT)** data model is shown in Figure 23.

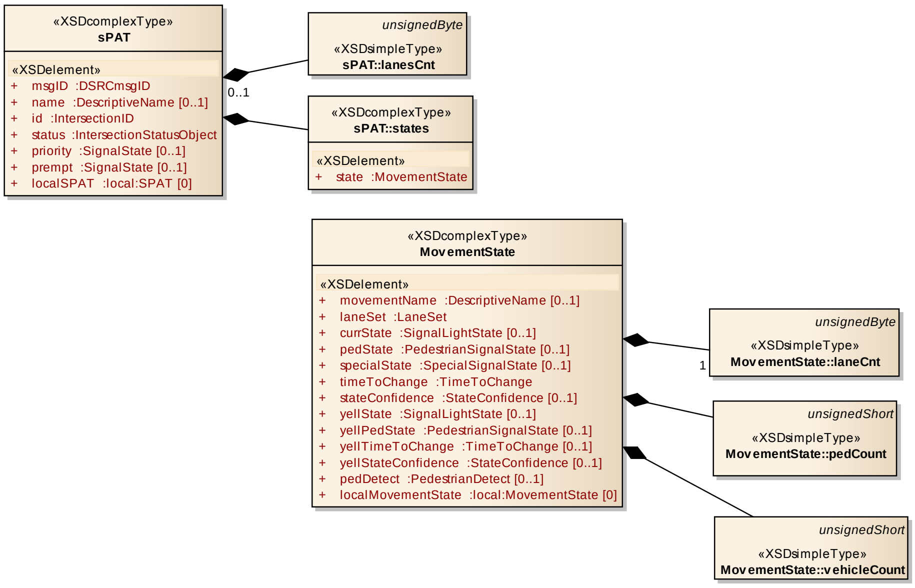

«XSDcomplexType»
sPAT
«XSDelement»
+ msgID :DSRCmsgID
+ name :DescriptiveName [0..1]
+ id :IntersectionID
+ status :IntersectionStatusObject
+ priority :SignalState [0..1]
+ prempt :SignalState [0..1]
+ localSPAT :local:SPAT [0]
0..1
unsignedByte
«XSDsimpleType»
sPAT::lanesCnt
«XSDcomplexType»
sPAT::states
«XSDelement»
+ state :MovementState
«XSDcomplexType»
MovementState
«XSDelement»
+ movementName :DescriptiveName [0..1]
+ laneSet :LaneSet
+ currState :SignalLightState [0..1]
+ pedState :PedestrianSignalState [0..1]
+ specialState :SpecialSignalState [0..1]
+ timeToChange :TimeToChange
+ stateConfidence :StateConfidence [0..1]
+ yellState :SignalLightState [0..1]
+ yellPedState :PedestrianSignalState [0..1]
+ yellTimeToChange :TimeToChange [0..1]
+ yellStateConfidence :StateConfidence [0..1]
+ pedDetect :PedestrianDetect [0..1]
+ localMovementState :local:MovementState [0]
1
unsignedByte
«XSDsimpleType»
MovementState::laneCnt
unsignedShort
«XSDsimpleType»
MovementState::pedCount
unsignedShort
«XSDsimpleType»
MovementState::vehicleCount

Figure 23 SPAT data model

SPAT is used to convey the current status of a signalized intersection. Along with the *Map Data message* (which conveys a full geometric layout of the intersection in question) the receiver of this message can determine the state of the signal phasing and when the expected next phase will occur. The SPAT message sends the current movement state of each active phase in the system as needed (values of what lights are active and values of for what durations the light is expected to continue). The state of inactive movements (typically all red) is not normally transmitted. Movements are mapped to specific lanes and approaches by use of the lane numbers present in the message. These lane numbers correspond to the specific lanes described in the MAP message for that intersection. The current signal pre-emption and priority status values (when present or active) are also sent. [5]

It is important to remark that this message has a sequence of the following relevant data: intersection identifier (*IntersectionID)*, status of the controller (*IntersectionStatusObject*), additionally the number of states to follow (*lanesCnt*), each active Movement/lane is given in turn and contains its state, seconds to the next event, etc. (*MovementsStates*) and optionally active priority (*SignalState*) and preemption (*SignalState*) state data.

The main part of this data structure consists of a sequence of *MovementsStates* for each lane in the intersection. This data frame is used to combine different information about current signal state of one or more lanes of a common type (motorized vehicle, pedestrian, train and transit lanes).

For further details see [5].

For the smart prioritization of vehicles used case in Madrid other 2 messages have been identified: ***Signal Request Message (SRM)*** to send the priority request while the bus is in the control zone and ***Signal Status Messages (SSM)*** to reply to the priority request.

Signal Request Message (SRM) is a message sent by a vehicle to the RSU in a signalized intersection. It is used for either a priority signal request or a preemption signal request depending the way the message flag is set. In either case, it identifies itself (using its VIN or another method supported by the VehicleIdent data frame), its current speed, heading and location (using the Blob of the BSM), and makes a specific request for service (Vehicle Request) as well as an anticipated time of service (a start time and end time in seconds from the present). The specific request for service is typically based on previously decoding and examining the list of supported zones for that intersection (sent in the *map data messages*). The outcome of the all pending requests to a signal can be found in the *Signal Status Message*, and may be reflected in the SPAT message contents if successful [5].

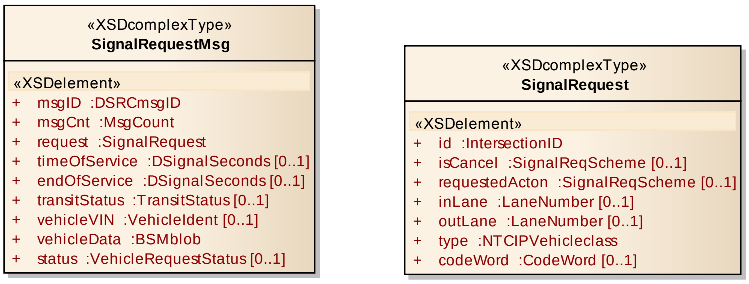

«XSDcomplexType»
SignalRequestMsg
«XSDelement»
+ msgID :DSRCmsgID
+ msgCnt :MsgCount
+ request :SignalRequest
+ timeOfService :DSignalSeconds [0..1]
+ endOfService :DSignalSeconds [0..1]
+ transitStatus :TransitStatus [0..1]
+ vehicleVIN :VehicleIdent [0..1]
+ vehicleData :BSMblob
+ status :VehicleRequestStatus [0..1]
«XSDcomplexType»
SignalRequest
«XSDelement»
+ id :IntersectionID
+ isCancel :SignalReqScheme [0..1]
+ requestedActon :SignalReqScheme [0..1]
+ inLane :LaneNumber [0..1]
+ outLane :LaneNumber [0..1]
+ type :NTCIPVehicleclass
+ codeWord :CodeWord [0..1]

Figure 24 SRM data model

As shown in Figure 24, the ***Signal Request Message*** is a sequence of requests to the intersection (*SignalRequest*) mainly containing the intersection identifier, a cancel flag, the requested action and optional lanes data such as, the time in the near future when service is requested to start (*timeOfService*), end of service (*endOfService*), additional information of transit events (*transitStatus*), vehicle identifier (*vehicleVIN*) and its current position (*vehicleData*) and status (*status*).

For further details see [5].

Signal Status Messages (SSM) is a message sent by an RSU in a signalized intersection. It is used to relate the current status of the signal and any collection of pending or active pre-emption or priority events acknowledged by the controller. The data contained in this message allow other users to determine their "ranking" for any request they have made as well as to see the currently active events. When there have been no recently received requests for service messages, this message may not be sent. The outcome of the all pending requests to a signal can be found in the Signal Status Message, and the current event may also be reflected in the SPAT message contents if successful [5].

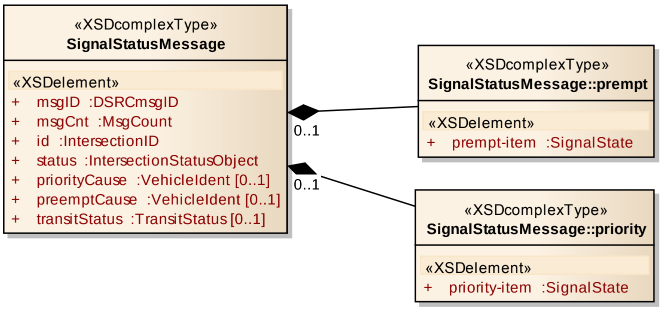

«XSDcomplexType»
SignalStatusMessage
«XSDelement»
+ msgID :DSRCmsgID
+ msgCnt :MsgCount
+ id :IntersectionID
+ status :IntersectionStatusObject
+ priorityCause :VehicleIdent [0..1]
+ preemptCause :VehicleIdent [0..1]
+ transitStatus :TransitStatus [0..1]
0..1
0..1
«XSDcomplexType»
SignalStatusMessage::prempt
«XSDelement»
+ prempt-item :SignalState
«XSDcomplexType»
SignalStatusMessage::priority
«XSDelement»
+ priority-item :SignalState

Figure 25 SSM data model

The ***SSM*** mainly includes the information related with general status of each signal controller (*status*), an optional attribute covering active priority (*priority* of *SignalState*) and preemption (*prempt* of *SignalState*) state data, optional also the identification of the vehicle asking for priority (*priorityCause*) or preemption (*preemptCause*) and, finally, additional information pertaining to transit evens (*transitStatus*).

For further details see [5].

Incident Data

No extension is necessary as In-Time simplified data model already covers all data requirements.

Urban Car Park Data

No extension is necessary as In-Time simplified data model already covers all data requirements for static and dynamic information related with car parking. However, to cover other public transport as bike sharing, car hiring or electric vehicle hiring and charging, an analogue model is proposed. In the case of bikes, the information related to bike sharing is kept in an analogous structure, with the following adaptations: *BikeSharingPoints* are included as possible *PointOfInterest;* additional datatypes *BikeUsingTariffs, BikeSharingDynamic* and *BikeBasicData* are included. For those elements being outdoors, specific attributes of parking places are removed (e.g. queue estimation, entrances/exit location, type, disabled capability).

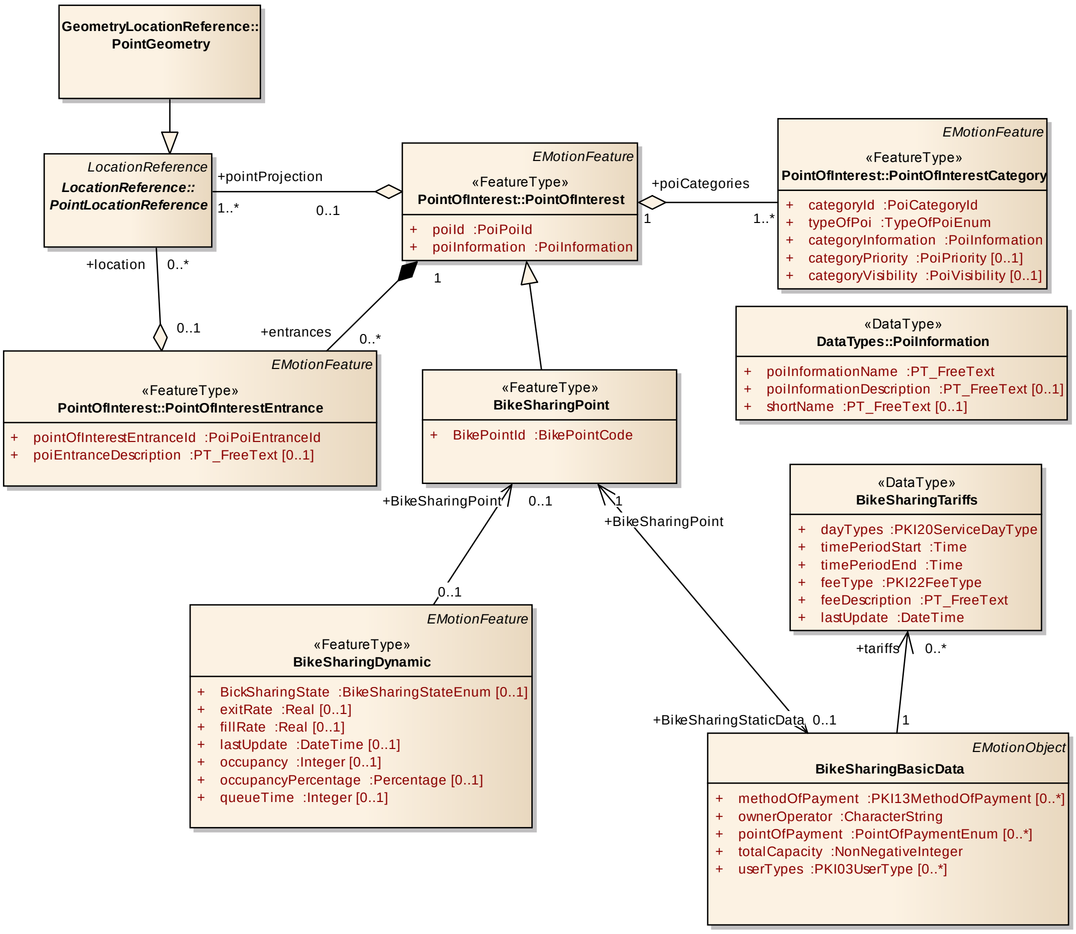

GeometryLocationReference::PointGeometry
LocationReference
LocationReference::PointLocationReference
+pointProjection
1..*
0..1
EMotionFeature
«FeatureType»
PointOfInterest::PointOfInterest
+ poiId :PoiPoiId
+ poiInformation :PoiInformation
+poiCategories
1
1..*
EMotionFeature
«FeatureType»
PointOfInterest::PointOfInterestCategory
+ categoryId :PoiCategoryId
+ typeOfPoi :TypeOfPoiEnum
+ categoryInformation :PoiInformation
+ categoryPriority :PoiPriority [0..1]
+ categoryVisibility :PoiVisibility [0..1]
+location
0..*
0..1
+entrances
0..*
1
EMotionFeature
«FeatureType»
PointOfInterest::PointOfInterestEntrance
+ pointOfInterestEntranceId :PoiPoiEntranceId
+ poiEntranceDescription :PT_FreeText [0..1]
«FeatureType»
BikeSharingPoint
+ BikePointId :BikePointCode
«DataType»
DataTypes::PoiInformation
+ poiInformationName :PT_FreeText
+ poiInformationDescription :PT_FreeText [0..1]
+ shortName :PT_FreeText [0..1]
«DataType»
BikeSharingTariffs
+ dayTypes :PKI20ServiceDayType
+ timePeriodStart :Time
+ timePeriodEnd :Time
+ feeType :PKI22FeeType
+ feeDescription :PT_FreeText
+ lastUpdate :DateTime
+BikeSharingPoint
0..1
1
+BikeSharingPoint
0..1
EMotionFeature
«FeatureType»
BikeSharingDynamic
+ BickSharingState :BikeSharingStateEnum [0..1]
+ exitRate :Real [0..1]
+ fillRate :Real [0..1]
+ lastUpdate :DateTime [0..1]
+ occupancy :Integer [0..1]
+ occupancyPercentage :Percentage [0..1]
+ queueTime :Integer [0..1]
+tariffs
0..*
+BikeSharingStaticData
0..1
1
EMotionObject
BikeSharingBasicData
+ methodOfPayment :PKI13MethodOfPayment [0..*]
+ ownerOperator :CharacterString
+ pointOfPayment :PointOfPaymentEnum [0..*]
+ totalCapacity :NonNegativeInteger
+ userTypes :PKI03UserType [0..*]

Figure 26 Dynamic Bike sharing Model

3.3 Public Transport Operation Management

The main objective of this section is to model all the information needed in order to fulfil MoveUs pilot and use cases requirements in terms of public transport operation and management. This information covers a wide range of different concepts such as the definition and planning of different lines or routes or journey tracking (vehicle of each trip, departure and arrival times, etc.).

3.3.1 Existing specifications

The existing In-Time simplified model (package Dynamic Public transport Information) covers the main part of MoveUs requirements as it includes the data models for journeys, service description and information related with stop points.

Each trip or journey is tracked by recording the *TargetVehicleJourney*, the information related to that vehicle (*TargetVehicleJourneyInfo*) and concepts related with the service and operator. It is important to remark that each trip is also linked with the *Line* description (specifying the direction, the line and the name) and the stop point elements. The stop point description is related to the *timetabledStopVisit*; more *timetabledStopVisit* form the Timetable. The

TargetedVehicleJourney is also associated to a *TargetedCall* with arrival and departure info and with an association to the stop point sequence description.

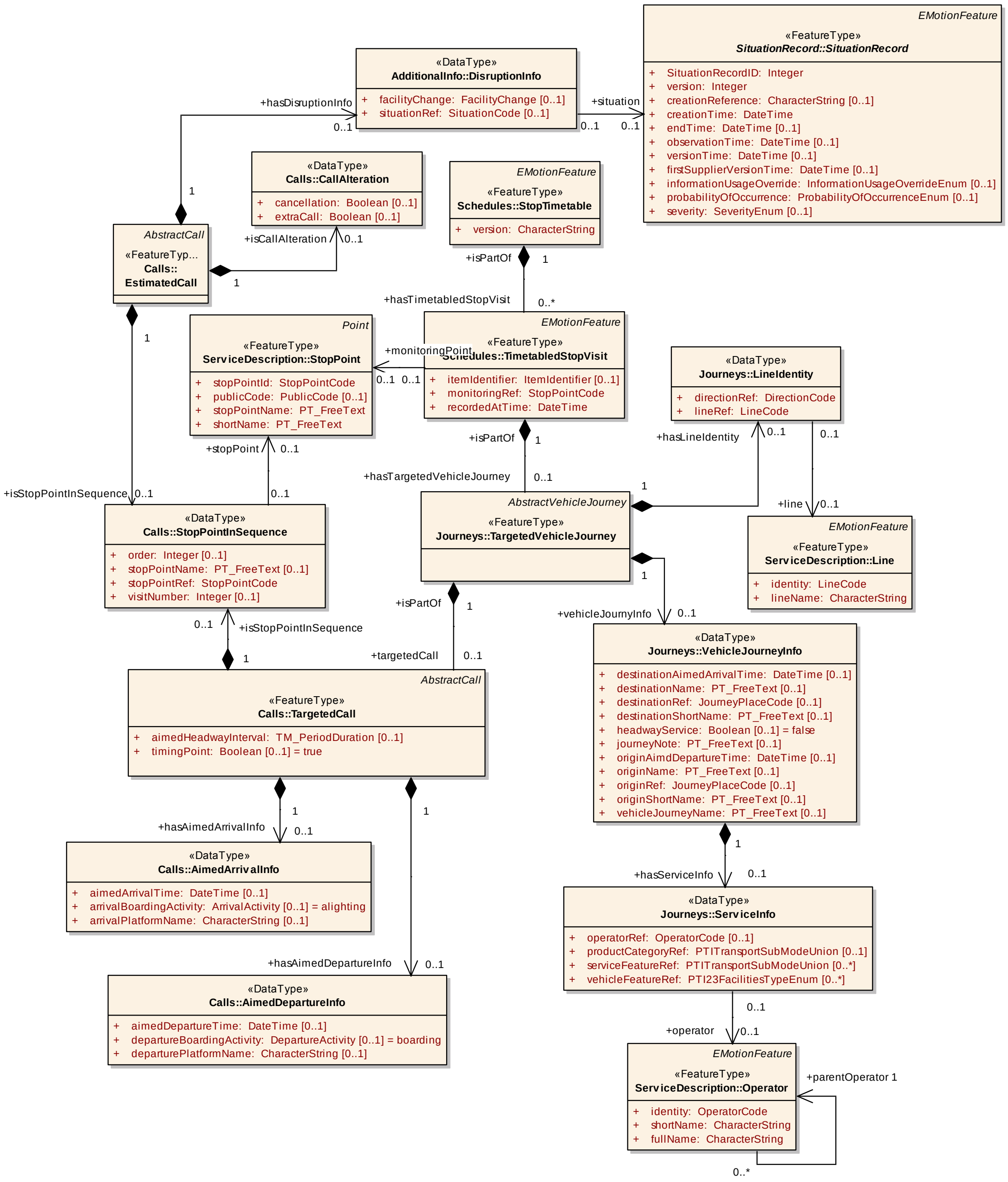

EMotionFeature
«FeatureType»
SituationRecord::SituationRecord
+ SituationRecordID: Integer
+ version: Integer
+ creationReference: CharacterString [0..1]
+ creationTime: DateTime
+ endTime: DateTime [0..1]
+ observationTime: DateTime [0..1]
+ versionTime: DateTime [0..1]
+ firstSupplierVersionTime: DateTime [0..1]
+ informationUsageOverride: InformationUsageOverrideEnum [0..1]
+ probabilityOfOccurrence: ProbabilityOfOccurrenceEnum [0..1]
+ severity: SeverityEnum [0..1]
«DataType»
AdditionalInfo::DisruptionInfo
+ facilityChange: FacilityChange [0..1]
+ situationRef: SituationCode [0..1]
+hasDisruptionInfo
0..1
+situation
0..1
0..1
1
«DataType»
Calls::CallAlteration
+ cancellation: Boolean [0..1]
+ extraCall: Boolean [0..1]
+isCallAlteration
0..1
EMotionFeature
«FeatureType»
Schedules::StopTimetable
+ version: CharacterString
+isPartOf
1
AbstractCall
«FeatureTyp...
Calls::
EstimatedCall
1
+hasTimetabledStopVisit
0..*
Point
«FeatureType»
ServiceDescription::StopPoint
+ stopPointId: StopPointCode
+ publicCode: PublicCode [0..1]
+ stopPointName: PT_FreeText
+ shortName: PT_FreeText
+monitoringPoint
0..1
0..1
EMotionFeature
«FeatureType»
Schedules::TimetabledStopVisit
+ itemIdentifier: ItemIdentifier [0..1]
+ monitoringRef: StopPointCode
+ recordedAtTime: DateTime
«DataType»
Journeys::LineIdentity
+ directionRef: DirectionCode
+ lineRef: LineCode
1
+hasLineIdentity
0..1
0..1
+stopPoint
0..1
+isPartOf
1
+hasTargetedVehicleJourney
0..1
+isStopPointInSequence
0..1
0..1
1
«DataType»
Calls::StopPointInSequence
+ order: Integer [0..1]
+ stopPointName: PT_FreeText [0..1]
+ stopPointRef: StopPointCode
+ visitNumber: Integer [0..1]
AbstractVehicleJourney
«FeatureType»
Journeys::TargetedVehicleJourney
+line
0..1
EMotionFeature
«FeatureType»
ServiceDescription::Line
+ identity: LineCode
+ lineName: CharacterString
1
+isPartOf
1
+vehicleJournyInfo
0..1
0..1
+isStopPointInSequence
1
+targetedCall
0..1
AbstractCall
«FeatureType»
Calls::TargetedCall
+ aimedHeadwayInterval: TM_PeriodDuration [0..1]
+ timingPoint: Boolean [0..1] = true
«DataType»
Journeys::VehicleJourneyInfo
+ destinationAimedArrivalTime: DateTime [0..1]
+ destinationName: PT_FreeText [0..1]
+ destinationRef: JourneyPlaceCode [0..1]
+ destinationShortName: PT_FreeText [0..1]
+ headwayService: Boolean [0..1] = false
+ journeyNote: PT_FreeText [0..1]
+ originAimdDepartureTime: DateTime [0..1]
+ originName: PT_FreeText [0..1]
+ originRef: JourneyPlaceCode [0..1]
+ originShortName: PT_FreeText [0..1]
+ vehicleJourneyName: PT_FreeText [0..1]
1
1
+hasAimedArrivalInfo
0..1
«DataType»
Calls::AimedArrivalInfo
+ aimedArrivalTime: DateTime [0..1]
+ arrivalBoardingActivity: ArrivalActivity [0..1] = alighting
+ arrivalPlatformName: CharacterString [0..1]
1
+hasServiceInfo
0..1
«DataType»
Journeys::ServiceInfo
+ operatorRef: OperatorCode [0..1]
+ productCategoryRef: PTITransportSubModeUnion [0..1]
+ serviceFeatureRef: PTITransportSubModeUnion [0..*]
+ vehicleFeatureRef: PTI23FacilitiesTypeEnum [0..*]
+hasAimedDepartureInfo
0..1
«DataType»
Calls::AimedDepartureInfo
+ aimedDepartureTime: DateTime [0..1]
+ departureBoardingActivity: DepartureActivity [0..1] = boarding
+ departurePlatformName: CharacterString [0..1]
0..1
+operator
0..1
EMotionFeature
«FeatureType»
ServiceDescription::Operator
+ identity: OperatorCode
+ shortName: CharacterString
+ fullName: CharacterString
+parentOperator 1
0..*

Figure 27 Public Transport (PT) Service Model

3.3.2 Extensions

An extension of the data model is operated for Car Pooling Management according to the requirements emerging from the Use Case definition.

The trips stored as carpooling offerings are defined by the *MV_CarPoolingTrips* feature types. Here the same feature types used for Journey Origin and

Destinations are used to identify the origin and destination of the carpooling trip (*JP:OriginDestinationRequestType*).

Additionally this feature type includes the attributes that define the additional details of the offering:

- Calendar: set the days of the week and time when the offering is available
- Role: driver, passenger or unspecified
- Visibility: set if the offering is visible or not
- Roundtrip: to specify whether it is a round trip
- Validity: to specify until when the offering is valid

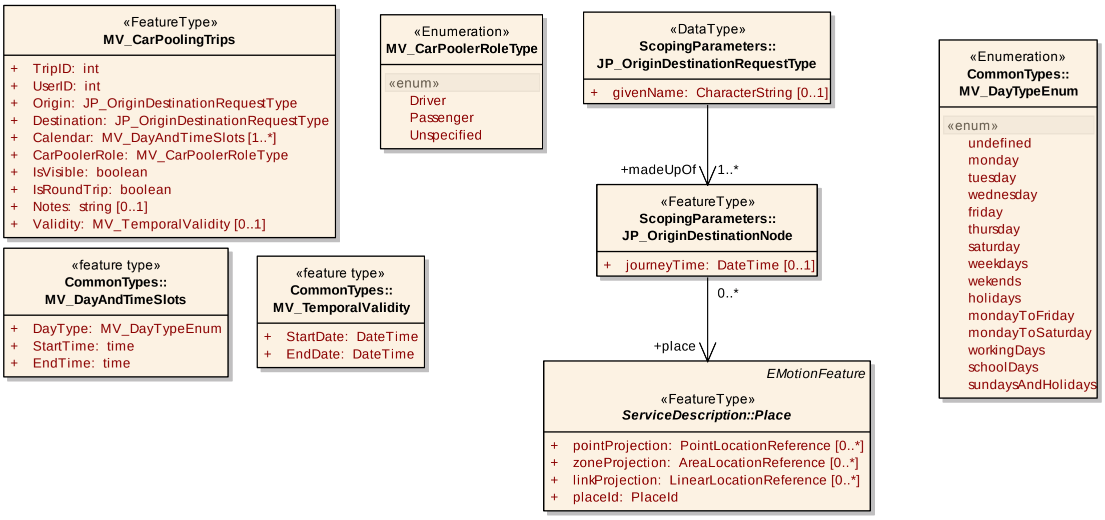


Figure 28 Carpooling Service Model

Other extension needed is related to specific management carried out in the scope of Madrid use-case (UC1):

On the one hand, a more precise information of each bus line or route is included by (1) adding *headerA* and *headerB* attributes in *Line* class in order to specify the starting and ending places, (2) summarizing the temporal information of the scheduling in the *LineTimingDescription* class (i.e. maximum and minimum frequency, temporal information of the first and last service and which days the service is available) and, finally, (3) defining the geographical stop place sequence for each line (*LineStopSequence* and *StopPlace*).

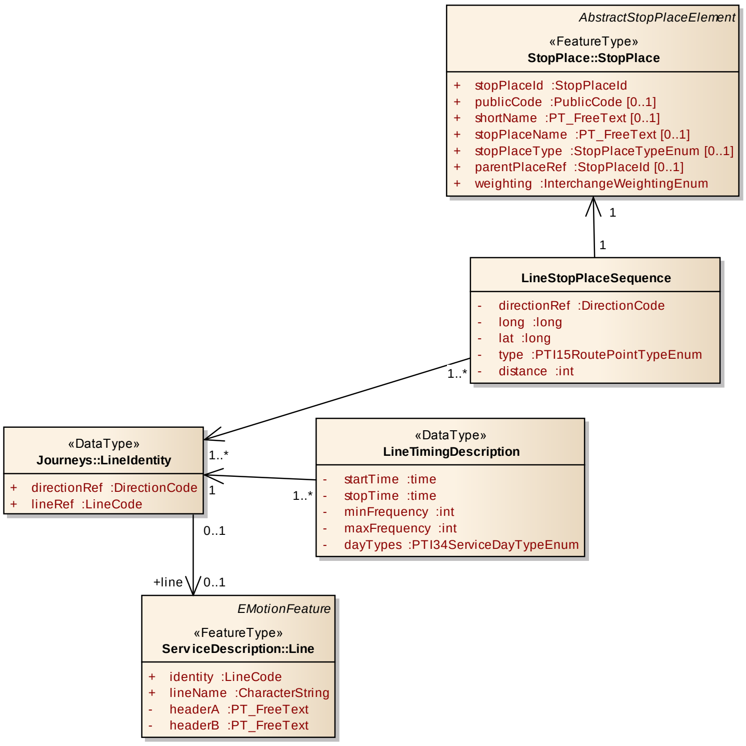


Figure 29 PT Line Management Model

On the other hand, for the correct development of Madrid use-case (UC1) it is necessary to incorporate more detailed information of each trip (Figure 30). With that objective, the following attributes have been incorporate to the already defined *VehiculeJourneyInfo* class: *dayType*, *expedition* (theoretical or real vehicle number), *directionRef* (journey direction), *trip* (theoretical or real journey number), *originDepartureTime* and *destinationArrivalTime*.

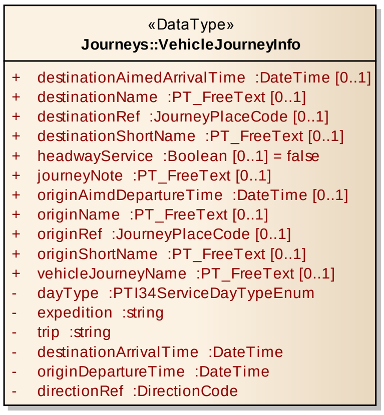

«DataType»
Journeys::VehicleJourneyInfo
+ destinationAimedArrivalTime :DateTime [0..1]
+ destinationName :PT_FreeText [0..1]
+ destinationRef :JourneyPlaceCode [0..1]
+ destinationShortName :PT_FreeText [0..1]
+ headwayService :Boolean [0..1] = false
+ journeyNote :PT_FreeText [0..1]
+ originAimdDepartureTime :DateTime [0..1]
+ originName :PT_FreeText [0..1]
+ originRef :JourneyPlaceCode [0..1]
+ originShortName :PT_FreeText [0..1]
+ vehicleJourneyName :PT_FreeText [0..1]
- dayType :PTI34ServiceDayTypeEnum
- expedition :string
- trip :string
- destinationArrivalTime :DateTime
- originDepartureTime :DateTime
- directionRef :DirectionCode

Figure 30 PT Line Management Model (detail)

3.4 Functional Traveller Journey Assistance

This functional area enables multi-modal information provision, journey planning, and on-trip trace and support.

The relevant data blocks are the following:

Private Trip Plan Data. The result of the trip planning process is used along the on-trip phase, as a reference to identify perturbations affecting estimation times and travel viability.

Road Trip Planning Data. It contains information about the road network and the traffic conditions within for use in planning trips. It shall be possible to integrate current and predicted data for different date/time combinations. It is mainly used for planning.

PT Trip Planning Data. Analogous to Road Trip Planning Data, it integrates the information about the services provided by the Public Transport operator plus the fares that will be charged; and it shall be for use in planning trips.

Travel Information Data. Real-time updated network information.

General Trip Preferences (GTP) Data. Contains the personalised data needed to support the Traveller during all his/her trips, from the trip planning, trip execution and finally, being updated once finalized.

Personal Mobility Data. Repository of historical information, where resides all the information sent by the application track capture module. This information is used in the queries associated with mobility analysis.

3.4.1 Existing specifications

Trip Planning Data (Personal/Road and Public Transport). The next plot is extracted from eMotion data model and represents a complete single-multimodal journey planning. This single-multimodal journey planning covers the following options:

- Dynamic Road Traffic Routing Information.
- Dynamic Public Transport Journey Routing.
- Dynamic Walking Planning.
- Dynamic Cycling Planning.
- Comparative Dynamic Multi Modal Journey Planning.

JP_Journey class describes a journey and it is made up by several *JP_Legs* (a leg is a structure that is used to define each single journey. A journey can be made of several legs). There are 4 leg types:

- **Timed Leg**: a leg that has specific timing points associated to a timetable e.g. Public Transport.
- **Frequency Leg**: a leg that runs at specified frequencies.
- **Continuous Leg**: a continuous leg does not have a specific timing or frequency and is suitable for legs not covered by public transport, e.g. it can be used for car or walk legs.
- **Interchange Leg**: interchange legs are typically used for walking trips for interchange purposes. They have an origin and a destination and are described with a navigation path.

Another important concept to point out from this data model is that each *JP_Leg* can include information on Tracking&mapping (*JP_LegTrack*).

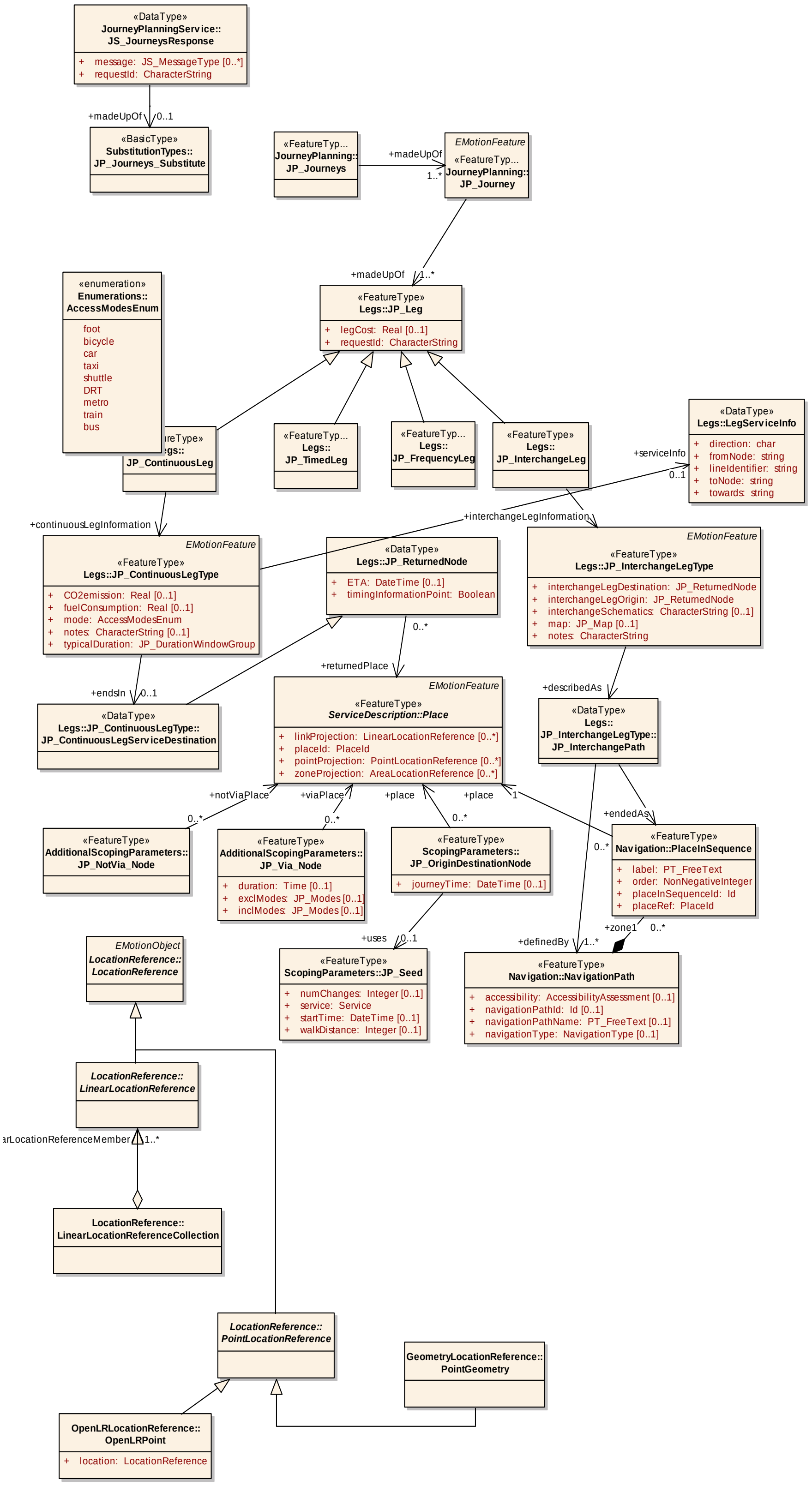


Figure 31 Journey Planning services

For deeper detail of classes and attributes needed for each leg see In-Time data model, which includes information regarding messages, service origin, etc.

Personal Mobility Data

As mentioned in the previous section, each *JP_Leg* can include information on Tracking&mapping *(JP_LegTrack*). The *TrackingAndMapping* contains data types suitable for Mapping and Tracking purposes. Objects of type *JP_Tracks* include a *JP_Map* and may include instructions. The next plot is extracted from eMotion data model.

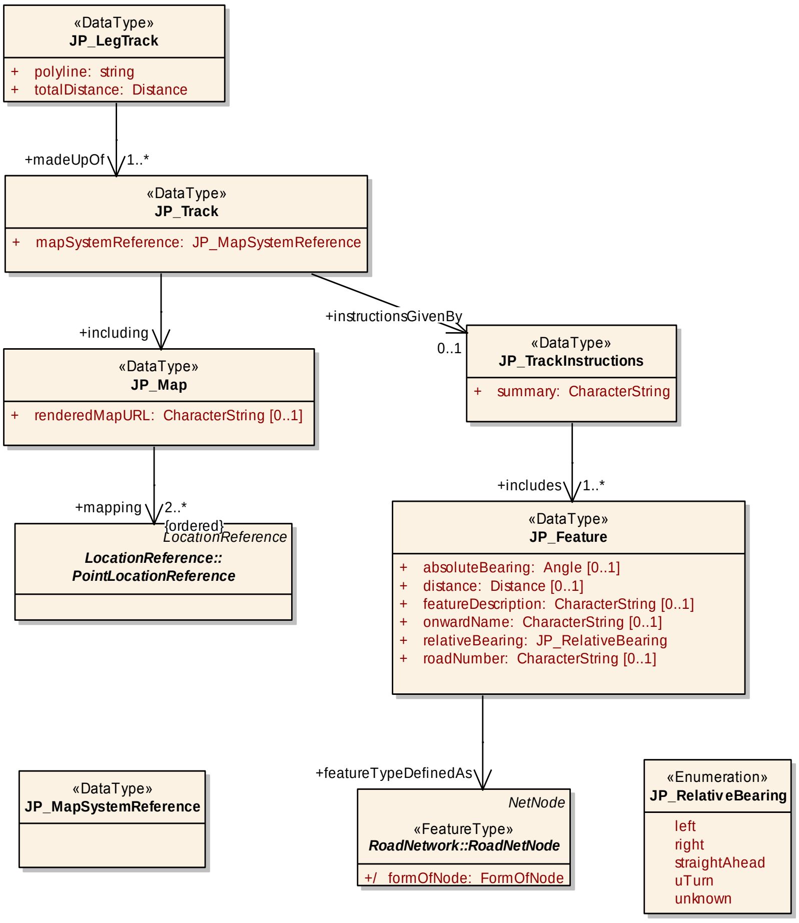


Figure 32 Journey Planning services

3.4.2 Extensions

An additional extension will be considered for MoveUs, associating an incentive (*MV_incentive,* see section 3.5) and recording the incentive associated with each journey. The incentive calculation is based on a set of rules, applicable for a specific temporal period and location, with user behaviour as parameter. Here, a journey plan (*JP_Journey*), is defined as a sequence of transport modes in a time slot.

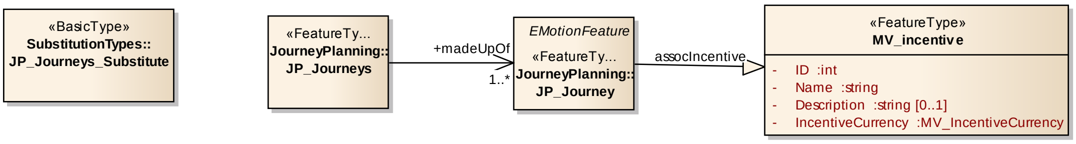


Figure 33 Link between Journeys and Incentives

Personal Mobility Data

An additional attribute *timestamp* will be added to keep the specific timing points associated to the different *JP_LegTracks* that compose the trip and allowing spatio-temporal analysis of the mobility patterns.

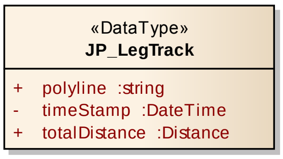


Figure 34 Trace Spatio-temporal information

3.5 Functional block Incentive Management

The component diagram depicting the organization and functional view on the Incentives Management is included as a reference.

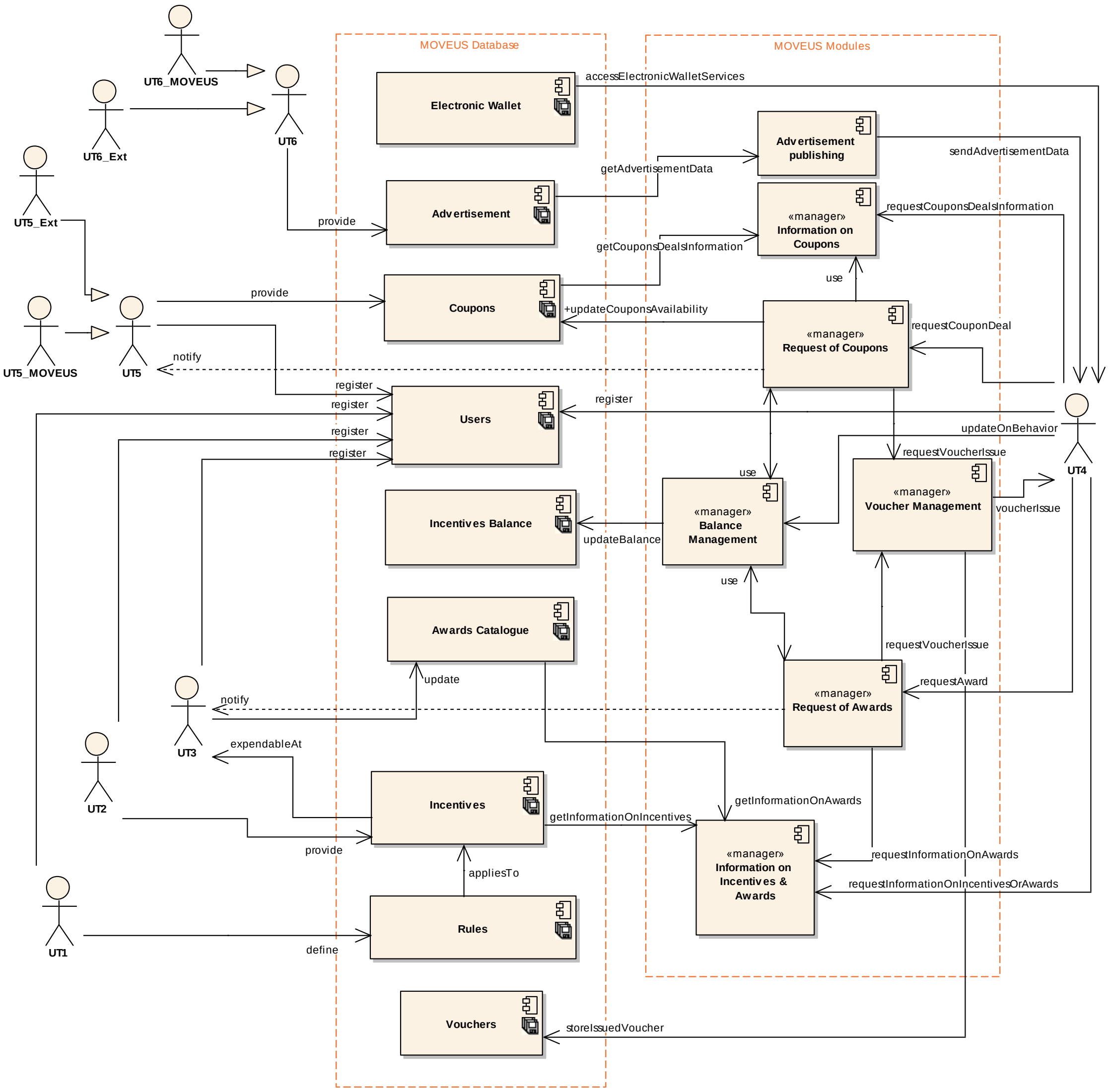


Figure 35 Incentive Management Functional View

A complete description of the Incentives model can be found in Deliverable D2.2 [6].

In chapter 3.1 of the present Deliverable, the Data model supporting the User Management according to the requirements of the Incentives model is described.

In order to define the Data Model supporting the data storages involved in the Incentives management, according to the model definition and as depicted in the previous diagram, the following high-level view is provided:

Table 2 Incentive data blocks

Users:	Registry of users described by: • ID • User Type • Name • Etc. Plus User-specific attributes
Electronic Wallet	Information and URL of available payment services the user can be redirected to. Described by: • Description • URL • Etc.
Advertisement	Data on Advertisement. Described by: • Description • Url of the advertisement • Etc.
Coupons	Data on Coupons. Described by: • Description • Validity (Geographical, Dates) • Cost • Etc.
Incentives	Data on incentives. Described by: • Description • Type • Unit of measure • Etc.
User Balance	Amount of incentives units (credits and coins) gained by the users. Described by: • Incentive type • Total

	• Etc.
Awards Catalogue	Benefits, awards, rewards that can be obtained by giving a certain amount of credits or coins. Described by: • Description • Cost • Validity • Etc.
Rules	Data that define the measure/rule for each incentive. Described by: • Description • Validity (Dates, Geographic area etc.) • Beneficiaries (Type 4 users) • etc.
Vouchers	Contains the historical data on Issued Vouchers.

The incentives-related data model is described by focusing on three aspects that together form the overall Incentives-related package:

- Introduction of incentive currencies
- Assignment of incentives to the user
- Definition of incentives and rules
- Awards, coupons, advertisement

3.5.1 Existing specifications

The incentive schema defined within MoveUs is supported by a specific and dedicated Data Model which needs to be defined completely. No existing parts of the In-Time/Co-Cities Data Model is then re-used.

3.5.2 Extensions

3.5.2.1 Incentive currencies

The basic types of incentives can be identified with three base units of measure or currencies:

Table 3 Measure/Currency Units

Unit of measure / Currency	**Type of incentive**
CREDIT	Incentives that are calculated from Energy Efficient Behaviour

M_COIN (MoveUs Coins)	Incentives that can be spent in general at more UT3s associated to MoveUs
B_COIN (Bonded Coin)	Incentives that can be spent only at one specific UT3

In order to allow a higher flexibility, an additional *MV_IncentiveCurrency* extends the base currency by adding the same features to it:

- A name
- A monetary value
- A temporal and spatial validity

The extension allows the management of different types of incentives (e.g. incentives provided by different organizations) either within one single City or from one City to another.

MV_IncentiveCurrency becomes the unit of measure for:

- Incentives storage in the user balance,
- Award assignment,
- Coupon issuing.

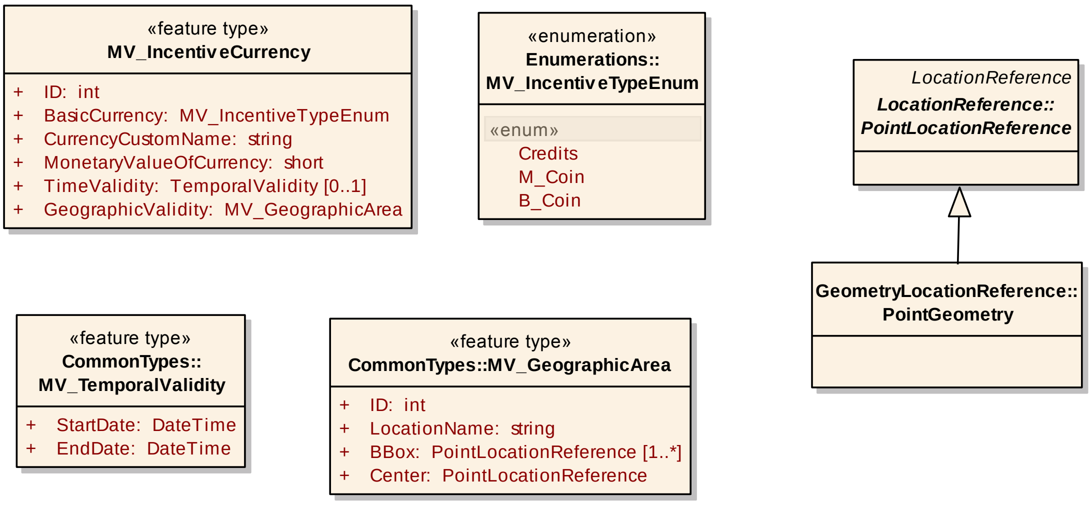


Figure 36 Incentive Currencies

3.5.2.2 Assignment of incentives

The feature type *MV_IncentiveBalance* is defined to store the amount of incentives. The assumption is that only the user type T (Traveller) can own zero or more incentives balances. Each balance features a specific incentive currency.

The Total amount of incentives is an attribute of the *MV_IncentiveBalance* class while the single transactions are described with the *MV_IncentiveTransactions* feature type.

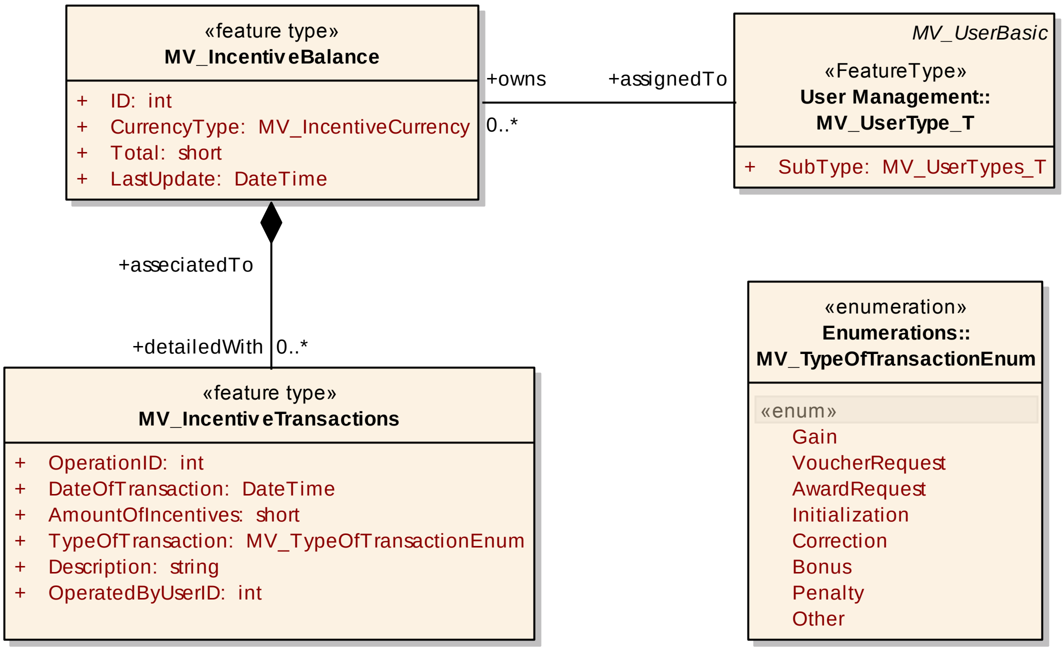


Figure 37 Incentive Transactions

3.5.2.3 Incentives and rules

An incentive is defined with a specific Incentive Currency and follows one or more rules that define how the incentives can be gained by the user.

The rule is composed by a super class with a base profile including a temporal and spatial validity. The sub-classes define the sub-rules, namely the set of attributes necessary to define how many incentives are rewarded for a specific situation or behaviour (whose relevant attributes are present in the sub-class to support the related city service functionalities). Two sub-rules are defined at the present stage:

- ***Smart mobility rule***: defines how many incentives can be gained by covering a distance with certain modes of transport and in specific timeslots.
- ***Feedback rule***: defines how many incentives can be gained by providing a number of feedbacks of a certain type.

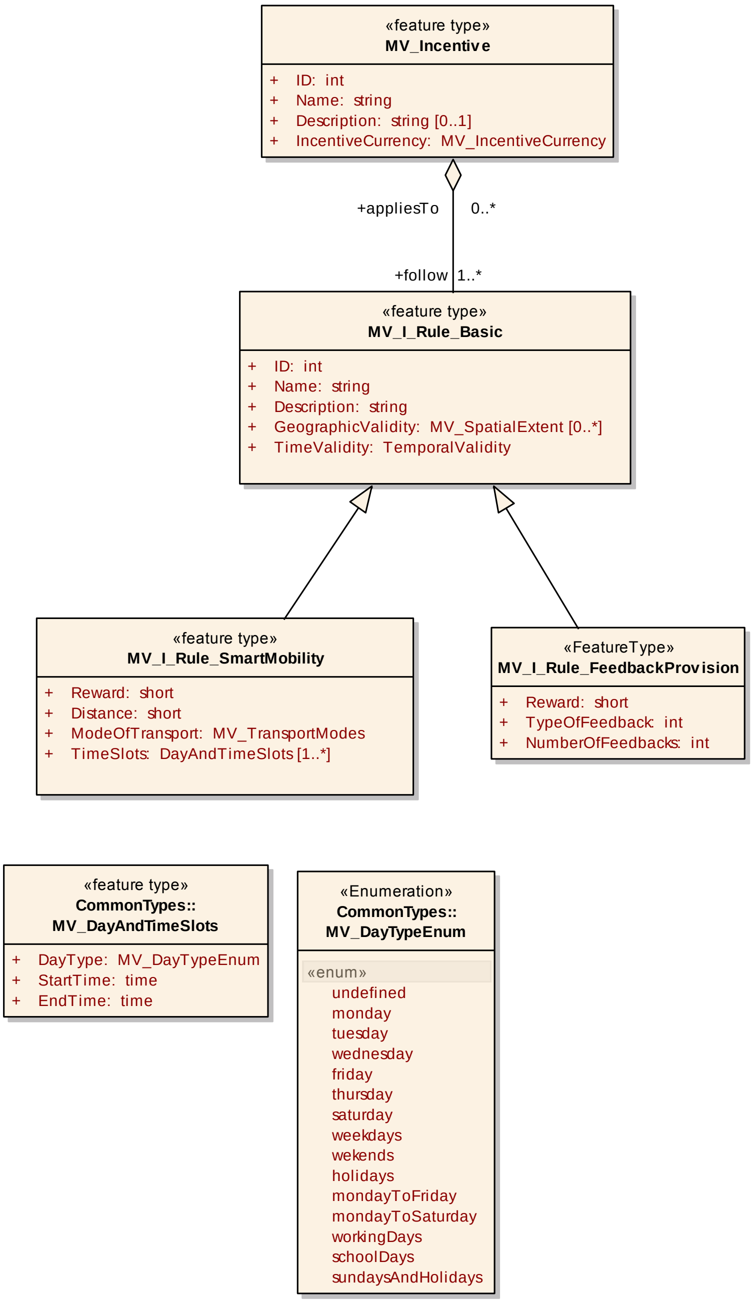


Figure 38 Incentive & Rules

3.5.2.4 Coupons, awards, vouchers, electronic wallet service and advertisement

The coupons are described by a complex feature (*MV_coupons*) that can be used to describe most aspects of the object associated to the coupon. An URL of a detail page is present to link the coupon to the organization that provides it.

The *MV_Award* describe the single entry of a catalogue of awards. A single award applies to one or more incentives.

Both Awards and Coupons can be obtained and paid by means of an *Incentive Payment Type*. The feature type *MV_IncentivePaymentType* extends the concept of Incentive Currency previously introduced by adding a monetary tradeoff. This tradeoff expresses the percentage that could be applied to pay an award or a coupon partially in incentives and partially in real money.

> **Example**:
>
> An instance of *IncentiveCurrency* named "CustomCoin" is of type "B_COIN", and each coin has a value of 1 euro.
>
> An instance of *IncentivePaymentType* has the value of CustomCoin and additionally has a monetary tradeoff of the 50%.
>
> An award of value = 100Euro can be then obtained with 100 CustomCoin or with 50 CustomCoin + 50 Euros.

MV_Advertisement is a basic feature type supporting the Advertisement object.

MV_Vouchers is used to store the information of the Vouchers that have been issued and can be used for historical purposes or for retrieving the details of the Voucher (for example at the moment of use of the voucher).

The Electronic Payment Service Registry (*MV_EPS_Registry*) is defined for the Electronic Wallet Service according to the definition of this service in the Use Case description. Having the characteristics of the Registry entity defined in **Error! Reference source not found.**. The *MV_EPS_Registry* is defined as a super class of it. See **Error! Reference source not found.** for more details on the registry features.

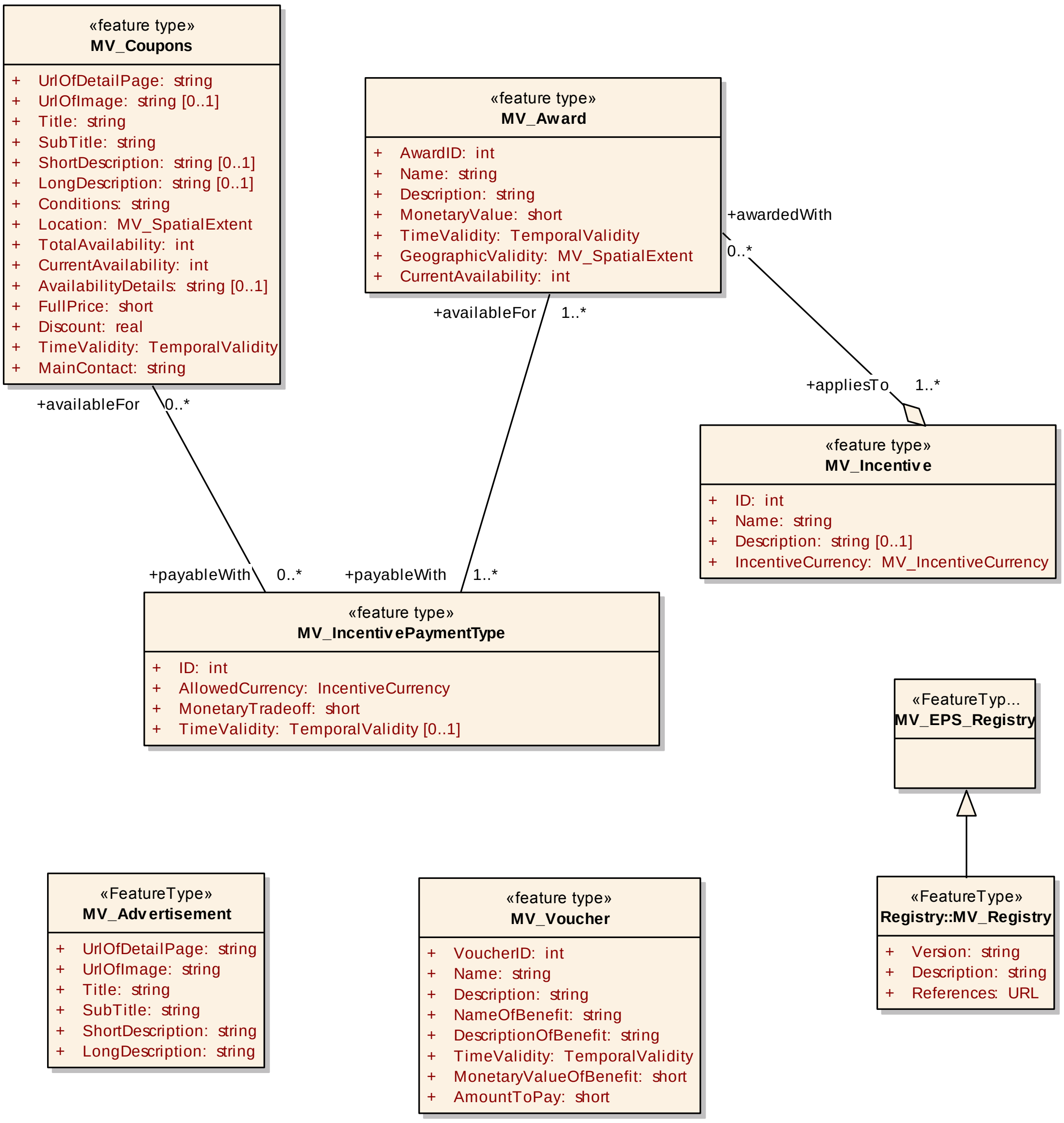


Figure 39 Coupons & Awards

3.5.2.5 Coordination with User Management functional block

As described in section 3.1 the definitions of the Incentives-related Data Model are fully harmonized with those of the User Management and therefore the feature types defined in the present functional module have to be understood and completed with those of Functional block 1.

3.6 Functional block CF/EC Estimation

Energy Consumption and Carbon Footprint issues will be supported by an specific algorithm to be used by a computational engine in order to output 'global' Energy/CO2 computed values (per user, per routing, per means of transportation etc.), based on input information concerning the Energy/CO2 label set and information related to mobility options chosen by users in real time. The assessment methodology, methods and underlying data needs are currently under definition in WP4.

3.6.1 Existing specifications

No existing parts of the In-Time/Co-Cities Data Model are then re-used. The energy efficiency schema defined in WP4 will be supported by a specific and dedicated Data Model which needs to be defined from scratch.

3.6.2 Extensions

Expected extensions of the common MoveUs data model are expected according to the WP4 working progress.

In advance, some of the key concepts to be managed are outlined at the table below.

Table 4 Energy Efficiency concepts

KPI	Set of Key Performance Indicators related with energy efficiency and carbon footprint in the transportation domain. Described by: • Description • Calculation • Relation to transportation modality • Etc.
Energy labels	Set of translations of energy efficiency values for users. Described by: • Description • Equivalence to user meaningful values, e.g. cost,
Energy affecting parameters	Set of parameters that are affected by KPIs in the different living labs. Described by: • Description • KPIs related to this parameter • Positive or negative effect for energy consumption and carbon footprint • Etc.

3.7 Functional block Feedback

MOVUS proposes a co-operative mobility concept, seen as the interconnection of users, vehicles and infrastructure that enables a full sharing of information between the different actors. Here, is a key issue the final user involvement. Specifically, the feedback provided by users enables cities and transportation operators to adapt their plans and offer in a more efficient way.

This feedback can be achieved in two different ways: passive, as automated application process, or active, answering the own user questions related to the quality of the own MoveUs Application information or the public transport environment (e.g. feedback traffic and transportation related events and metrics).

3.7.1 Existing specifications

The existing Co-Cities Data model defines a package of feedback-related feature types that are used in MoveUs.

Among the set of feedback services specified in Co-Cities, two domains are specifically considered for MoveUs:

1. Journey Planning-related feedbacks
2. Traffic-related feedbacks

The super-class *Feedback* features the base feedback attributes. This includes a Trust Level that can be used to differentiate (namely, assign different reliability levels) the subject who formulated the feedback.

The sub-classes of *Feedback* define the specific data types of the feedback information:

- Traffic feedback (new data about a traffic event): this is a class with an attribute of type *TrafficElement* (see section 3.2).
- Traffic quality feedback (quality of information of a given traffic event): a set of boolean values indicating if the specific information given about a traffic event is correct or not.
- Journey Planning quality feedback (quality of information of a given journey): a set of Boolean values indicating if the specific information given for each leg of a journey is correct or not.

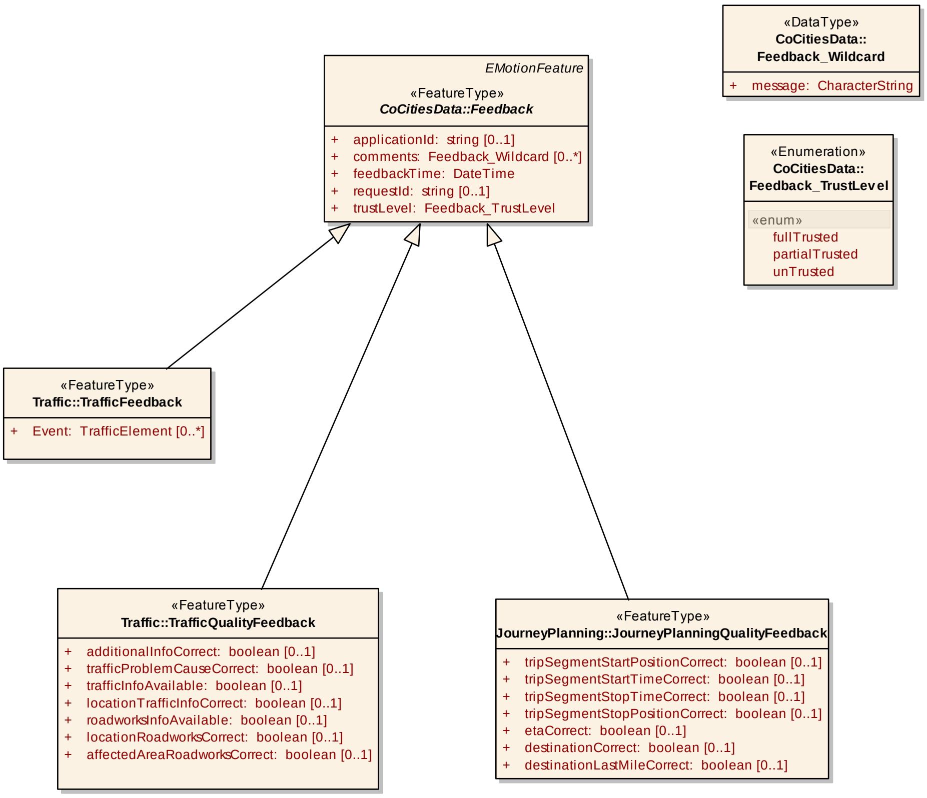

«DataType»
CoCitiesData::
Feedback_Wildcard
+ message: CharacterString
EMotionFeature
«FeatureType»
CoCitiesData::Feedback
+ applicationId: string [0..1]
+ comments: Feedback_Wildcard [0..*]
+ feedbackTime: DateTime
+ requestId: string [0..1]
+ trustLevel: Feedback_TrustLevel
«Enumeration»
CoCitiesData::
Feedback_TrustLevel
«enum»
fullTrusted
partialTrusted
unTrusted
«FeatureType»
Traffic::TrafficFeedback
+ Event: TrafficElement [0..*]
«FeatureType»
Traffic::TrafficQualityFeedback
+ additionalInfoCorrect: boolean [0..1]
+ trafficProblemCauseCorrect: boolean [0..1]
+ trafficInfoAvailable: boolean [0..1]
+ locationTrafficInfoCorrect: boolean [0..1]
+ roadworksInfoAvailable: boolean [0..1]
+ locationRoadworksCorrect: boolean [0..1]
+ affectedAreaRoadworksCorrect: boolean [0..1]
«FeatureType»
JourneyPlanning::JourneyPlanningQualityFeedback
+ tripSegmentStartPositionCorrect: boolean [0..1]
+ tripSegmentStartTimeCorrect: boolean [0..1]
+ tripSegmentStopTimeCorrect: boolean [0..1]
+ tripSegmentStopPositionCorrect: boolean [0..1]
+ etaCorrect: boolean [0..1]
+ destinationCorrect: boolean [0..1]
+ destinationLastMileCorrect: boolean [0..1]

Figure 40 Feedback Model

3.7.2 Extensions

No extension is foreseen at the moment for the MoveUs Feedback functional block compared to the Data features defined by the Co-Cities model.

3.8 Functional Block Registry

The registry of Metadata in MoveUs is used to direct the dynamic functionalities of the City Services.

3.8.1 Existing specifications

The existing eMotion specifications are used in part for the service (API) description.

3.8.2 Extensions

A simplified description profile for MoveUs is introduced.

An optional eMOTIONServiceDesc association links the registry to a more complete WSDL-based service description.

MV_Registry is a superclass of local registry entities (*MV_LocalRegistry*) that have a spatial extent (mandatory) and a temporal extent (non-mandatory).

The service descriptions (MoveUs-specific or eMotion-based) are associated to *MV_LocalRegistry*.

The access criteria, necessary for differentiating the functionalities in the city services are defined by considering the user type, defined as the union of the different (sub-)user types and also the additional role ID (see 3.1.2 for more details on these data types). With the implementation of the services a differentiation can then be operated by considering generic (*MV_UserType_MV*) or specific (types "I", "T" or "D") user profiles and by combining these with the additional User role if necessary. The possibilities can be:

- Anonymous access (generic MoveUs user type and commonly agreed user role)
- Access regulated by User Type only (using a common or known user role)
- Access regulated by User Role only (using the generic MoveUs user type)
- Combination of the previous two options

For each instance of access criteria a set of pair **key-values** is defined as a generic method for activating, de-activating or differencing the functionalities and features of the city services.

More information on the dynamic behaviour of the City Services can be found in Deliverable D3.3.

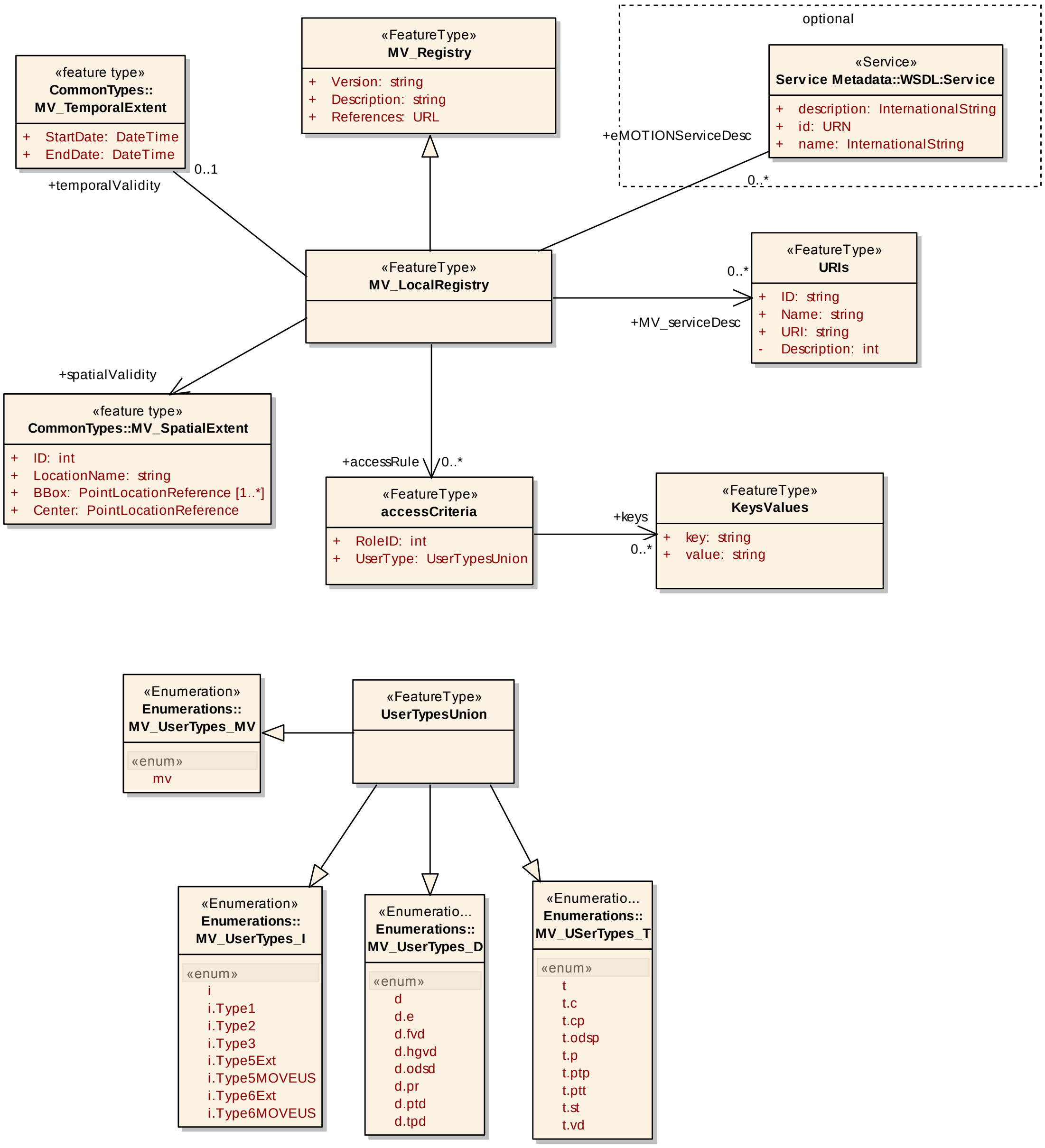


Figure 41 Registry Model

Additionally, in order to ensure the application tailoring and customization according to each city peculiarities (e.g. premium services as green route access available for profiles tagged as "eco-friendly"), official languages and identity (e.g. logo, app style). These characteristics will be coded by means of different key-values.

4 Conclusions

The outcomes of Task 3.1 remark the existing work in the field of mobility services and in particular the availability of information models, which cover much of the relevant static and dynamic elements identified as relevant. A significant portion of the concepts needed for the storage of information and provision of services in MoveUs project have been previously considered in the projects taken as reference.

However, a deep analysis of the use cases has determined necessary to define adaptations and specific packages for the most innovative project goals, i.e.: incentive management, energy efficiency, services customization and of course, those aspects related to intelligent traffic management.

Furthermore, the parallel execution of the definition activities for the information model (T3.1), system architecture (T3.2), service specifications (T3.3) and the underlying algorithms (e.g. calculation of energy efficiency and trip planner) have motivated an iterative working approach during this phase of the project.

While this document constitutes a mature version of the MoveUs data model, integrating the specific and complex aspects before mentioned, an update will be necessarily required by the end of the 1st project year (M12), to make sure that the final T3.1 result is fully aligned with the final outcome of the two parallel tasks mentioned (T3.2 and T3.3), which will be already available at that time.

5 References

[1] eMotion Project. http://www.emotion-project.eu/

[2] In-Time Project. http://www.emotion-project.eu/

[3] Co-cities Project. http://www.co-cities.eu/

[4] FRAME Framework. http://www.frame-online.net

[5] SAE J2735 DEDICATED SHORT RANGE COMMUNICATIONS (DSRC) (Rev 29)(2008)

[6] MoveUs deliverable D2.2 - Use cases, incentives-based model concept and common specifications for the pilots