INFORMATION TECHNOLOGIES FOR SHIFT TO RAIL

D1.1 – IT2Rail Domain Ontology Specification and Repository

Due date of deliverable: 31/08/2017

Actual submission date: 18/12/2017

Leader/Responsible of this Deliverable: Fraunhofer

Reviewed: Y

Document status		
Revision	Date	Description
1	15/09/2017	First issue
2	18/10/2017	First revision after Thales comments
3	19/10/2017	Second revision after Thales deliverable comments
4	18/12/2017	Final version after TMC approval

Project funded from the European Union's Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/05/2015 Duration: 36 months

REPORT CONTRIBUTORS

Name	Company	Details of Contribution
Robert Lehmann	Fraunhofer	
Dirk Walther	Fraunhofer	FhG internal Review
Maria Laura Trifiletti	RINA Consulting SA	Quality check

EXECUTIVE SUMMARY

This document is about the formal ontology that was developed in IT2Rail. The particular path that was followed in this project in the engineering process is described, as are the tools used and the repository that now holds the ontology for other partners and users. An insight into the transformation process and design decisions made while formalising the domain knowledge is given in an extra section.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This deliverable covers the three corresponding modules defined in Annex1 (Part A) of the IT2Rail proposal. First being the specification of the ontology (sections 3.1, 3.2 , 3.3 and 5), second the specification of the ontology repository (sections 4.2 and 6) and third the actual implementation documentation for the two.

Many of the IT2Rail partners have contributed to the results presented here. May it be in form of informal knowledge transfer or directly in form of sub-specification for certain components [1]. While all objectives defined for D1.1 IT2Rail Domain Ontology Specification and Repository could be reached, the focus of this document has shifted towards how the formal ontology evolved, what design decisions were made, what tools where used and how the formal ontology can be accessed.

The amount of time that had to be invested in formalising the domain knowledge exceeded the time that was estimated for this task. On the other hand, implementing an actual repository turned out to require less time. This balanced each other out and thus, overall, the two major tasks in this deliverable could be accomplished with the resources as planned. Links to the formal IT2Rail ontology written in OWL2 RL are provided in Section 6. The formal ontology is machine readable and processable. The ontology is also meant to be human readable, some experience in OWL2 serialisation is required, however. The tools introduced in Section 4 will help to gain insight into the structure of the ontology.

2. REFERENCED DOCUMENTS

Document	Protocol code	Title
GA		ANNEX 1 (part A) Research and Innovation action NUMBER — 636078 — IT2Rail
D2.7	WP2-DEL-004	Travel Shopping Ontology Document (FREL)
D3.7	ITR-WP3-D-THA-020-04	Booking & Ticketing Ontology document (FREL)
D4.7	WP4-DEL-007	Trip Tracker ontology document (FREL)
D5.7	ITR-WP5-D-POL-101-03	Travel Companion ontology document (FREL)
D6.7	ITR-WP6-D-POL-039-03	Business Analytics ontology document (FREL)

Table 1: Referenced IT2Rail documents

3. ONTOLOGY ENGINEERING APPROACH

A major challenge in creating a new ontology is understanding the domain whose knowledge it is meant to represent. This is where domain experts come into play. In IT2Rail, every project partner with domain knowledge about some aspect of transportation is a valuable source of knowledge. This section describes the steps that have been taken to acquire the transport domain knowledge, how the knowledge acquisition process evolved and which dead ends were encountered. Transferring domain knowledge to a formal ontology is the next, but not less important, step in the ontology engineering process. However, details on this step will be covered in Section 5.

3.1 GATHERING DOMAIN KNOWLEDGE

The very first approach to gather concepts relevant to IT2Rail (and the transport domain in general) was creating an Excel spread sheet listing terms together with a description of their meaning. While being effective in aggregating the terms, it soon turned out to be difficult to accurately express the semantic relationships between the terms in a spread sheet.

A preformatted Word document offered more convenience and clarity for gathering terms, describing their meaning and relationships to other terms. Figure 1 illustrates the structure of the entries in the word document in the latest versions. Red indicates concepts, properties are in blue and green is used for examples. An entry of this form was created for every concept deemed relevant by the domain experts.

3.21 Stop place

KEYWORD	StopPlace
DOMAIN	
SIMILAR TO	
RELATED TO	TransportInfrastructure, Vehicle, Traveler, Location, Quay, VehicleStoppingPlace, LocationCode, RouteLink, TransportMode, StopPlace, AirTransport, BusTransport, CoachTransport, RailTransport, Airport, BusStation, CoachStation, RailStation, Terminal, Topology, LocationCode, Hub, TravelEpisodeStartPoint, TravelEpisodeEndPoint, OpenProduct, Equipment
COMMENT	(draft/agreed)
RESPONSIBLE	

Is an element of the TransportInfrastructure where Vehicle(s) may stop and where Traveler(s) may board or leave Vehicle(s).

Additional Notes: A StopPlace has one or more Quay(s) and has one or more VehicleStoppingPlace(s). A StopPlace is located at a Location and has a Name and has one or more LocationCode(s). A StopPlace is connected by RouteLink(s) to one or more other StopPlace(s) at the level of VehicleStoppingPlace(s). A StopPlace may be qualified by TransportMode. StopPlace(s) can be organized in a structural hierarchy. A StopPlace can be composed of StopPlace(s). For example, London Heathrow is a StopPlace, Terminal 5 of London Heathrow is a child StopPlace of London Heathrow.

The definition is based on IFOPT.

Figure 1: Exemplary excerpt from Word document

However, while being an extremely valuable source of knowledge, the word document was almost impossible to maintain consistent with the evolving body of knowledge acquired ever-changing in the knowledge gathering phase. Introduction of “in-between”-terms, removal or shift of relations required extensive manual updates which turned out to be rather error-prone. The word document is the source for WP2-6 Ontology deliverables and consolidates them.

3.2 SOURCE FOR THE FORMAL ONTOLOGY

The main source for the formal OWL ontology is the UML-based model that has been developed by domain experts in the Capella tool (Version 0.8.3) [2]. Capella offers a lifecycle-driven approach to system modelling. As for the ontology, the main source of knowledge are the almost sixty class diagrams that where produced to reflect certain use cases in IT2Rail. Figure 2 depicts a class diagram of one of the elements of IT2Rail - the location resolver. This and all other class diagrams or rather their elements (classes, attributes, and relations) were formalised into an OWL ontology in the engineering process.

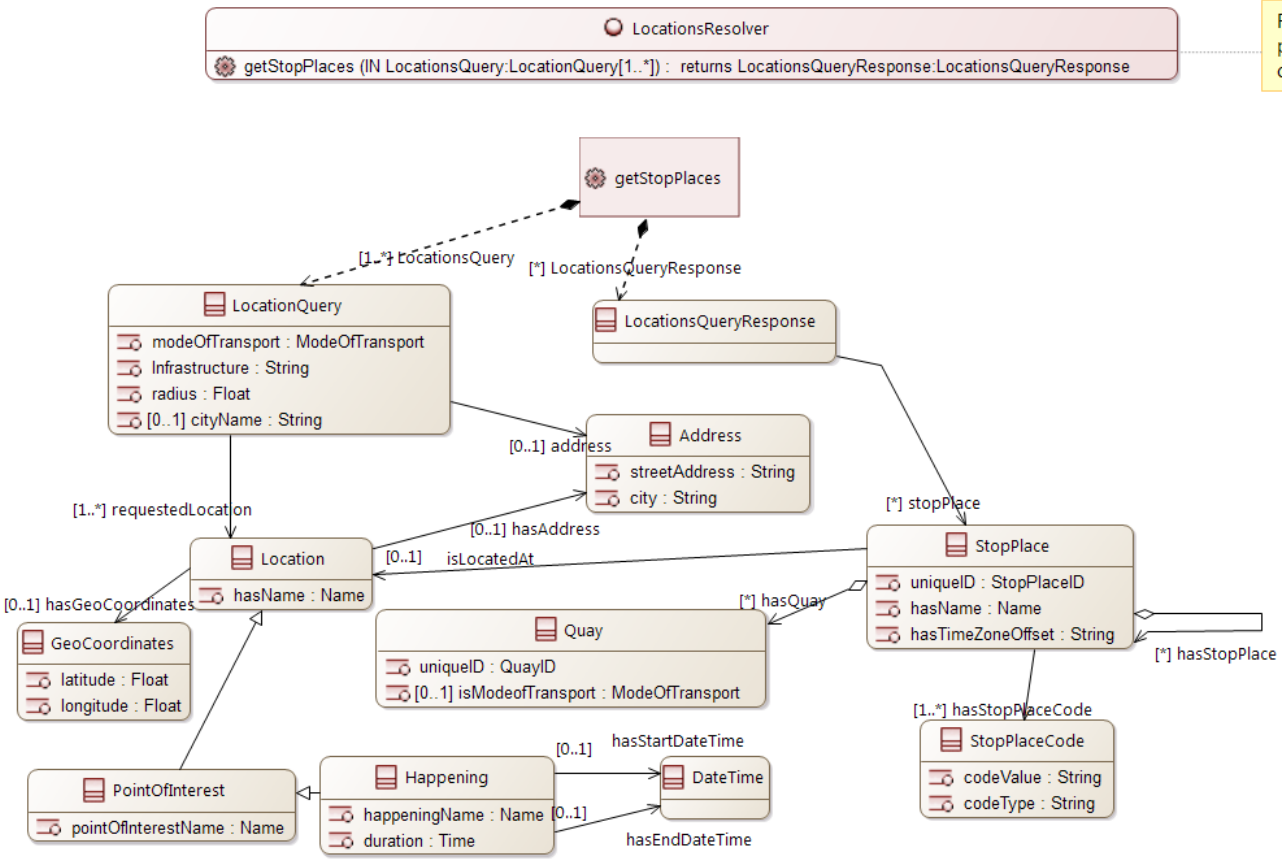

Figure 2: Example of class diagram in Capella (representing Location Resolver)

In addition to the word document and the UML domain model in Capella, various discussions with domain experts yielded aspects and details of the transportation domain that could not easily be expressed in either model (see Section 5.3 for an example). In the later phase of developing the OWL ontology, an issue tracking system (Mantis) provided by IT2Rail partner HaCon was employed to keep track of changes, bugs and ongoing discussions.

3.3 TECHNICAL REQUIREMENTS TOWARDS THE ONTOLOGY

A number of technical requirements existed from the beginning or they appeared as IT2Rail progressed. The foremost requirement was that the ontology language should be capable of accurately **describing the domain knowledge and technical services** in a unified manner. A **high expressivity** was required which means that arbitrary facts can be expressed and any relation and restriction can be formulated. As the developed ontology is envisioned to be used in follow-up projects, **industry established standards** that are **machine readable** and future-proof were an important requirement for the format of the ontology. These requirements are all met by the Web Ontology Language (OWL) [3].

At first, OWL2 DL [4] was chosen as it meets all the above requirements. However, the triple store that was used by other IT2Rail partners to realise certain use cases only supports OWL2 RL [5]. While OWL2 RL is a less expressive fragment of OWL2, it still allows a high degree of freedom in modelling but it required certain design decisions to be reconsidered and adapted. OWL2 RL is the language in which the IT2Rail ontology is formulated.

OWL2 can be serialised in various formats which does not have any impact on expressivity or semantics. All available formats are interchangeable, they only differ in resulting file size (plaintext vs. XML formats) and human readability. The chosen format for IT2Rail is RDF/XML for no specific reason.

4. TOOLS EMPLOYED IN THE ONTOLOGY ENGINEERING PROCESS

Two major tools where employed in the formal ontology engineering process. **WebProtégé** for collaborative working and Desktop Version of **Protégé** for complex tasks. It turned out that openly available and well-established tools should be favoured with respect to future developments in following projects. The following subsections will give a short introduction to those tools, their benefits regarding IT2Rail and how they were used and can be used in the future.

4.1 PROTÉGÉ

Protégé [6] is a commonly used tool for advanced ontology engineering in the scientific world. It offers variety of vital features to create and maintain evolving ontologies. Protégé offers support for OWL2 DL Ontologies including feature-complete axiom specification and allows to interact with common reasoners. Figure 3 shows the start-tab after opening an ontology in Protégé. Besides some ontology statistics, general ontology annotation, includes and namespaces can be viewed an edited. The Protégé GUI is highly configurable and can integrate advanced features not of immediate importance to IT2Rail. At the time of writing this document, the current version of Protégé (Version 5.2) was used to work with the IT2Rail ontology. Protégé is a free and open source tool under BSD 2-clause license available from the projects website[1].

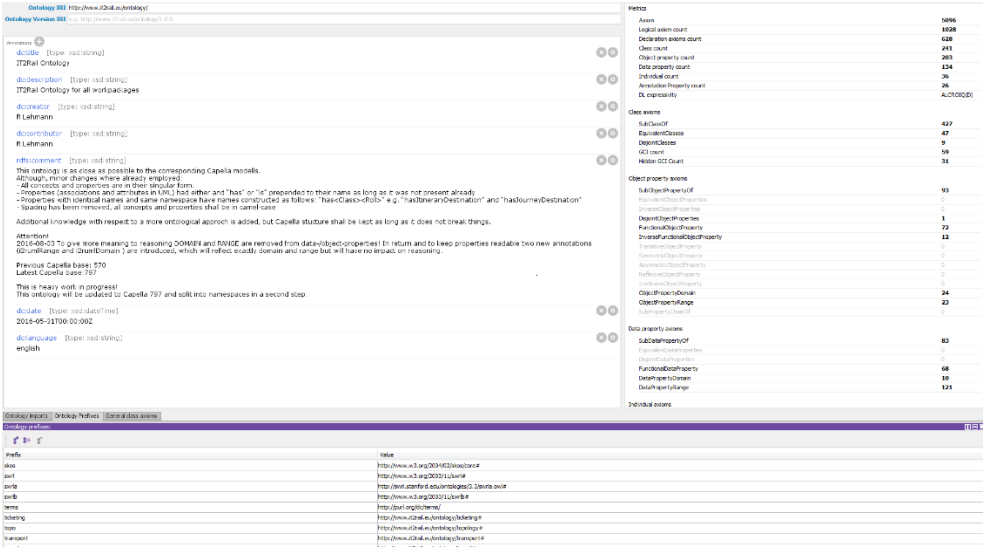

Figure 3: Protégé 5.2 general ontology GUI

Figure 4 displays the concept editor tab of Protégé. This is where concepts can be created and annotated and taxonomies and axioms can be engineered. Analogous tabs exist for object- and data-properties, datatypes and individuals. Noteworthy is the ability of Protégé to consistently refactor every element of an ontology. However, this only works for loaded or directly included

[1] https://protege.stanford.edu/

ontologies. For that reason, the ontology engineering process was carried out to a large extend using one ontology file only. The refactoring and splitting tools offered by Protégé where used to create separate files in the end.

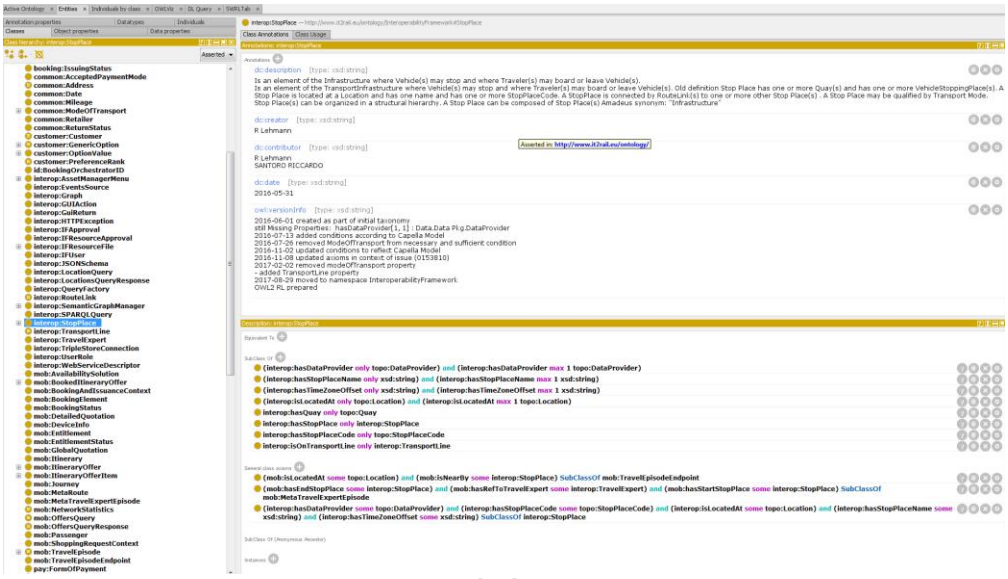

Figure 4: Protégé concept editor

As the IT2Rail ontology uses the OWL2 RL fragment, the specification of so-called General Concept Inclusions (GCIs) is an important feature to express existential restrictions on concepts. Protégé offers a convenient way to specify and keep track of relevant GCIs for a given concept as displayed for the example of the IT2Rail concept “StopPlace” in Figure 5.

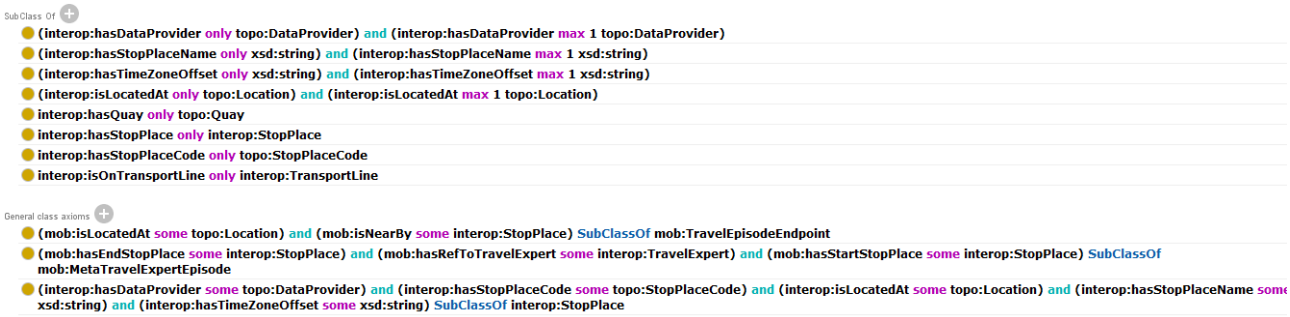

Figure 5: Protégé axiom editor (representing StopPlace)

4.2 WEBPROTÉGÉ

The browser-based version of Protégé (WebProtégé) [7] was used for collaborative development of basic ontology features including creating classes, properties, annotation, taxonomies and basic axioms. Besides collaboration, WebProtégé offers various useful features which Protégé does not, most importantly change tracking, discussion and ontology sharing. A WebProtégé instance running at the Fraunhofer IVI Institute also serves as the ontology repository; see Section 6 for details. A detailed user manual for WebProtégé can be found in [7]. WebProtégé is a free an open source tool under BSD 2-clause license. For IT2Rail the latest stable version (Version 2.6.0) was used.

Figure 6 illustrates the Interface of WebProtégé. Like the desktop version its appearance is highly customisable. Everybody can create an account on the Fraunhofer IT2Rail[2]-page and upon registration is granted permission to read (and download) or edit the ontology.

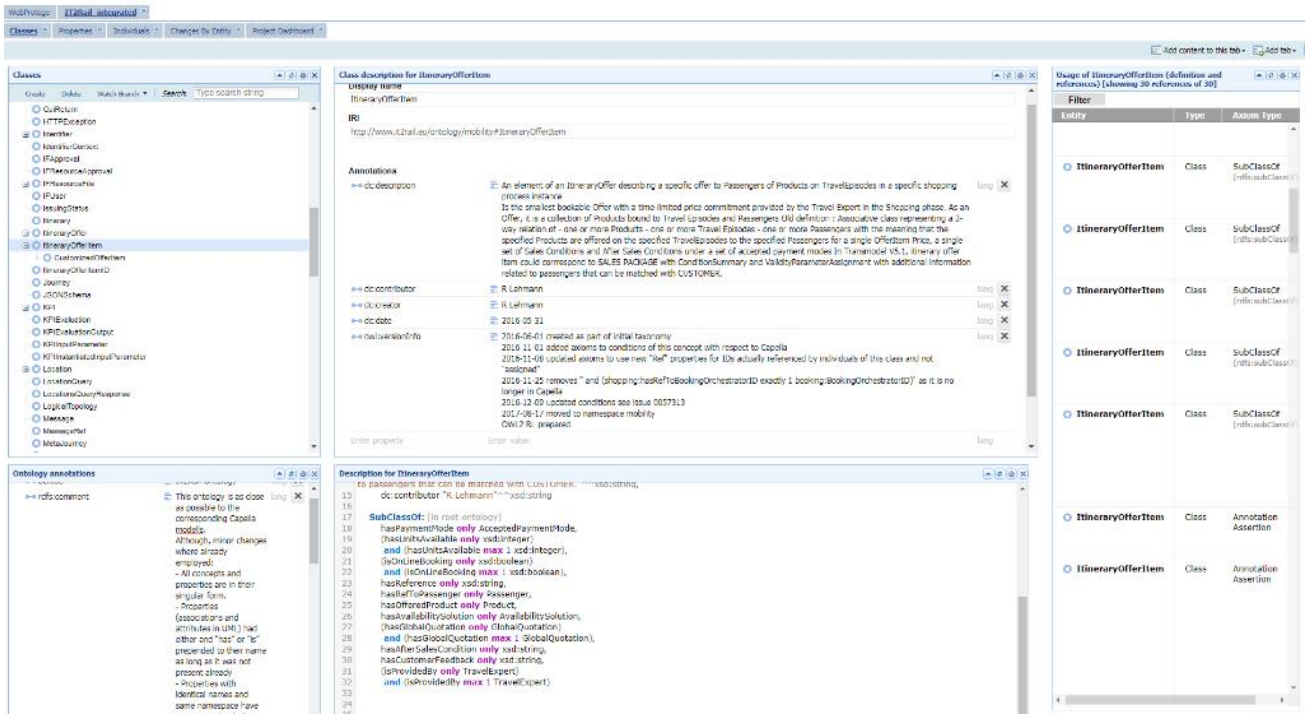

Figure 6. WebProtégé GUI

[2] http://it2rail.ivi.fraunhofer.de

5. DESCRIPTION OF THE FORMAL ONTOLOGY

In this section, we describe the structure of the ontology, the meaning of its elements, the design decisions that have been taken as well as how the UML [8] model (see Section 3.2) has been translated to OWL. Note that this document will not contain a copy of the formalised ontology itself.

5.1 UML TO OWL TRANSITION

There are some approaches in the scientific world e.g. [9] for the purpose of transforming UML to OWL. All approaches found that aim for automatic translation, however, have some prerequisites towards the actual model that is to be transformed. Ranging from special UML stereotypes that have to be used over special UML profiles to preferred modelling styles. For various reasons, this was not feasible in IT2Rail as the aim was to get domain experts to describe their view on functional aspects, not to get them to model easily translatable UML. Zedlitz et al. [10] describe a generic – manual – approach to transform UML class diagrams to OWL2. The method applied in IT2Rail is very closely related. With the major exception of enumeration and compositions transformation.

The version of the source UML is 2.5 [8] the target is an OWL 2 RL ontology specified in [3] and [5]. The basics of the UML-OWL-transformation in IT2Rail are described in this section. They appear somewhat incomplete as they only feature elements that actually appear in the IT2Rail UML class diagrams, which was decided by project members to be the primary source of information for the formal ontology.

In general, the following rules have been applied in the translation process:

- UML Classes are transformed to ontological concepts. Names in UML and OWL are identical with the exception of whitespaces being removed from some UML class names. This is straightforward and no other special exceptions exist;
- UML Attributes either translate to OWL data-properties in case they demand for plain datatypes (e.g. string, int, etc.), or to OWL object-properties in case their datatype references another IT2Rail class. Property names are as close to the UML as possible. A “has”, “is” or “isFor” is prepended to some names. All properties follow the lowerCamelCase schema;
- Associations (including aggregation and compositions) translate to object-properties. The same naming conventions as for attributes apply for associations;
- Left-hand and right-hand side of associations are translated to annotations in the ontology; see the next section for details on that decision;
- Lower and upper bounds (multiplicities) on attributes and associations translate to restrictions on concepts. (See section 5.4 for an example) In cases where it is absolutely certain that an attribute or association is only used once in the entire ontology and the

cardinality is one at most, the corresponding property in the ontology will be set to have functional characteristic.

- Directions of associations, indicated by the arrow association end and/or by name of the association (e.g. “isLocatedAt”), translate to restrictions on OWL concepts (see Section 5.4 for an example);
- There are no methods in IT2Rail UML, therefore they are not translated;
- Specialisations of classes translate to “subClassOf” in the ontology;
- Enumerations in UML are translated to concepts in the ontology. The enumerators are translated to individuals of that concept in the ontology;
- Interfaces are not translated to OWL. They basically reference the concepts that will ever appear on external service interfaces in IT2Rail;
- Folders (Packages) in UML are transformed to namespaces in the ontology. The following namespaces represent the relevant UML Packages:
    - BADataPkg
    - Common
    - Identifiers
    - IteroperabilityFramework
    - Mobility
    - Payment
    - Product
    - TCDataPkg
    - Topology
- Textual class descriptions in UML translate annotations;
- Sticky notes in UML translate to restrictions in the ontology if they contain additions knowledge not expressible in UML.

5.2 DESIGN DECISIONS

Besides the general translation rules, some design decisions had to be made. The main reasons where to alleviate some technical aspects imposed by Capella and to ease the future usage of the ontology for upcoming projects.

- Domain and Range are not used in the ontology. Instead and to keep it easy and understandable to non-OWL experts, two annotation properties where introduced:

"i2rumlRange" and "i2rumlDomain". These properties are of type xsd:string and represent exactly the OWL Domain and Range that are imposed by UML associations on the left- and right-hand side. The annotation properties are meant to keep the original structure of UML without having any impact on classification;

Collections are not translated into OWL directly but get resolved. This happens by modifying the cardinalities of the associations/properties; for example, see Figure 7. The association between "Token" and "PayloadCollection" is of cardinality "exactly 1". This however is translated in the ontology too "Token" has "at least" one "Payload". In Protégé is expressed as follows:

```
Class: Token

   SubClassOf:
      hasPayload only Payload

      … hasPayload some Payload …
   SubCLassOf:
      Token
```

Table 2: Collection as expressed in the IT2Rail Ontology

- All own IDs in UML classes have been removed from concepts since individuals are defined to be globally unique in OWL. For instance, the class "Token" will not have a property "TokenID" since every individual classified as token will already be identified by a globally unique IRI;

  Compositions were enforced in in way that the dependent participant in the relation will always have an axiom indicating that an inverse relation to the antecedent must exists; for example, see Figure 7. The composition – existential dependency – between Token and Payload is enforced by:

```
Class: Payload

      Inverse hasPayload some Token
   SubClassOf:
      Payload
```

Table 3: Composition as expressed in the IT2Rail Ontology

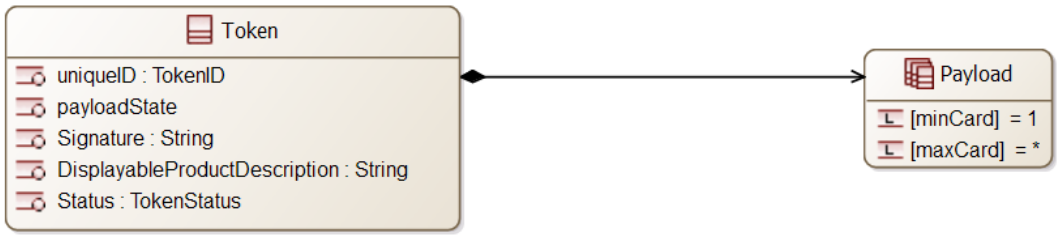

Figure 7: Example from IT2Rail UML for collection and composition

- Classes/concepts that would only "contain" single purpose or single use individuals/objects where resolved to DataProperties. This particular applies to time individuals;
- In case of explicit UML multiplicities lager than one (and smaller than a reasonable value), multiple data/object properties with almost identical names where introduced e.g.: UML [1..2] will produce "someObjectProperty1" and "someObjectProperty2" which are sub-properties of "someObjectProperty". This is due to limitations in OWL2 RL;
- Properties (data/object) belong to the namespace of the domain concept. In case the domain is union of concepts from separate namespaces the property is replicated in each namespace and a super property is introduced in root namespace (topmost ontology). There are two reasons behind that design decision. First, since the ontology is split by namespaces into separate files, whoever uses parts of the IT2Rail ontology in the future will be enabled to stay in one ontology/namespace for many use cases with no need to even consider additional files/namespaces by referencing properties their properties. Second, applying changes – in particular refactoring – to concepts/properties in the defining namespace/ontology will not immediately produce errors in ontologies referencing that element.

In addition, the IT2Rail ontology uses annotations on concepts and properties to document them. In some cases, empty annotations are used as placeholders to indicate that no information was provided by the UML model. This usually concerns informal descriptions of concepts and properties. The following annotations have been used:

- dc:description: reflecting the human readable informal description of the concept from text document (see Section 3.1) or UML;
- dc:creator: reflects the name of the person who created the element in the formal ontology;
- dc:contributor: reflects the person(s) who contributed to the element in the formal ontology;
- dc:date: reflects the date the element was introduced;
- owl:versionInfo: reflects the version history of the element;

- owl:deprecated: indicates that the element was used in a former version of the formal ontology but is not yet removed due to compatibility reasons;
- rdfs:comment: contains additions comments of the contributors;
- an:i2rumlDomain: union of concepts that form the domain of an date/object property;
- an:i2rumlRange: union of concepts that form the range of an object property.

5.3 ADDITIONAL KNOWLEDGE IN THE FORMAL ONTOLOGY

Besides the UML model, the most important source of information for the IT2Rail ontology is the knowledge of the domain experts. In some cases, however, this knowledge could not be described in UML due to its limited expressive power. There are various examples in the formal ontology. The following example shall depict that matter on – very much abstracted – domain expert knowledge (left) that cannot easily be described in UML class diagrams but requires a representation in the ontology (right in Manchester notation).

"A rail station is something where something else starts or ends that is operated by a third something that is in some kind of rail transport mode"	Class: RailStation EquivalentTo: (inverse isEndingAt some (inverse isOperating some ((isInModeOfTransport some RailTransport) and (isInModeOfTransport only RailTransport)))) or (inverse isStartingAt some (inverse isOperating some ((isInModeOfTransport some RailTransport) and (isInModeOfTransport only RailTransport))))

Table 4: Example OWL-axiom

5.4 EXAMPLE TRANSFORMATION

Figure 8 displays an example of an IT2Rail UML class diagram. It is an excerpt of a much larger diagram but suffices to demonstrate how UML is translated to OWL in IT2Rail. For the sake of

implicity and readability some attributes are removed that are in the UML model. This concerns attributes from most relations that exist in IT2Rail but not in the example figure.

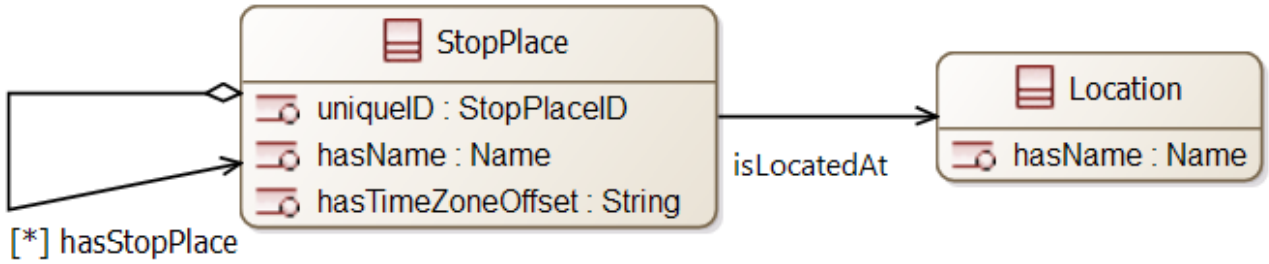

Figure 8: Excerpt from IT2Rail Diagram

Transformation below includes the annotation for class “StopPlace”. “…” indicates that an annotation has been shortened in this example. An annotation like that is present for every entry but would exceed the scope of this document. The Transformation in Manchester [11] like syntax. Note that Manchester syntax is the most compact format but officially does not support certain constructs required for an OWL2 RL ontology.

```
Class: interop:StopPlace

    Annotations:
        owl:versionInfo "2016-06-01 created as part of initial …"^^xsd:string,
        dc:creator "R Lehmann"^^xsd:string,
        dc:date "2016-05-31"^^xsd:string,
        dc:contributor "R Lehmann SANTORO RICCARDO"^^xsd:string,
        dc:description "Is an element of the …"^^xsd:string

    SubClassOf:
        (interop:isLocatedAt only topo:Location)
         and (interop:isLocatedAt max 1 topo:Location),
        (interop:hasStopPlaceName only xsd:string)
         and (interop:hasStopPlaceName max 1 xsd:string),
        (interop:hasTimeZoneOffset only xsd:string)
         and (interop:hasTimeZoneOffset max 1 xsd:string),
        interop:hasStopPlace only interop:StopPlace,

         (interop:isLocatedAt some topo:Location) and
         (interop:hasStopPlaceName some xsd:string) and
```

```
          (interop:hasTimeZoneOffset some xsd:string)
    SubClassOf:
        interop:StopPlace

Class: topo:Location

    SubClassOf:
        (topo:hasLocationName only xsd:string)
         and (topo:hasLocationName max 1 xsd:string)

         topo:hasLocationName some xsd:string
    SubClassOf:
        topo:Location

ObjectProperty: :isLocatedAt

ObjectProperty: interop:isLocatedAt

    Annotations:
        an:i2rumlDomain "StopPlace"^^xsd:string,
        an:i2rumlRange "Location"^^xsd:string,

    SubPropertyOf:
        :isLocatedAt

    Characteristics:
        Functional

ObjectProperty: :hasStopPlace

ObjectProperty: interop:hasStopPlace

    Annotations:
        an:i2rumlRange "StopPlace"^^xsd:string,
        an:i2rumlDomain "StopPlace"^^xsd:string

    SubPropertyOf:
        :hasStopPlace

DataProperty: interop:hasTimeZoneOffset

    Annotations:
        an:i2rumlDomain "StopPlace"^^xsd:string,
```

```
    Range:
        xsd:string

DataProperty: :hasName

DataProperty: interop:hasStopPlaceName

    Annotations:
        an:i2rumlDomain "StopPlace"^^xsd:string,

    Range:
        xsd:string

    SubPropertyOf:
        :hasName

DataProperty: topo:hasLocationName

    Annotations:
        an:i2rumlDomain " Location"^^xsd:string,

    Range:
        xsd:string

    SubPropertyOf:
        :hasName
```

Table 5: Example IT2Rail Ontology in Manchester-Syntax

The same transformation in another format – turtle [12].

```
@Prefix ontology: <http://www.it2rail.eu/ontology/> .
@Prefix: interop: <http://www.it2rail.eu/ontology/InteroperabilityFramework#> .
@Prefix: owl: <http://www.w3.org/2002/07/owl#> .
@Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@Prefix: dc: http://purl.org/dc/elements/1.1/ .

### http://www.it2rail.eu/ontology/InteroperabilityFramework#StopPlace
interop:StopPlace rdf:type owl:Class ;
   rdfs:subClassOf
     [ owl:intersectionOf ( [ rdf:type owl:Restriction ;
        owl:onProperty interop:isLocatedAt ;
```

```
        owl:allValuesFrom topo:Location
        ]
        [ rdf:type owl:Restriction ;
        owl:onProperty interop:isLocatedAt ;
        owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
        owl:onClass topo:Location
        ]
      ) ;
    rdf:type owl:Class
    ] ,
    [ owl:intersectionOf ( [ rdf:type owl:Restriction ;
        owl:onProperty interop:hasStopPlaceName ;
        owl:allValuesFrom xsd:string
        ]
        [ rdf:type owl:Restriction ;
        owl:onProperty interop:hasStopPlaceName ;
        owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
        owl:onDataRange xsd:string
        ]
      ) ;
    rdf:type owl:Class
    ] ,
    [ owl:intersectionOf ( [ rdf:type owl:Restriction ;
        owl:onProperty interop:hasTimeZoneOffset ;
        owl:allValuesFrom xsd:string
        ]
        [ rdf:type owl:Restriction ;
        owl:onProperty interop:hasTimeZoneOffset ;
        owl:maxQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
        owl:onDataRange xsd:string
        ]
      ) ;
    rdf:type owl:Class
    ] ,
    [ rdf:type owl:Restriction ;
    owl:onProperty interop:hasStopPlace ;
    owl:allValuesFrom interop:StopPlace
    ] ,
    [ rdf:type owl:Restriction ;
    owl:onProperty interop:hasStopPlaceCode ;
    owl:allValuesFrom topo:StopPlaceCode
    ];
  dc:contributor """R Lehmann, SANTORO RICCARDO"""^^xsd:string ;
```

```
   dc:creator "R Lehmann"^^xsd:string ;
   dc:date "2016-05-31"^^xsd:string ;
   dc:description """Is an element of …"""^^xsd:string ;
   owl:versionInfo """2016-06-01 created as part of initial …"""^^xsd:string .

### http://www.it2rail.eu/ontology/InteroperabilityFramework#isLocatedAt
interop:isLocatedAt rdf:type owl:ObjectProperty ;
   rdfs:subPropertyOf ontology:isLocatedAt ;
   rdf:type owl:FunctionalProperty ;
   an:i2rumlDomain "StopPlace"^^xsd:string ;
   an:i2rumlRange "Location"^^xsd:string ;

### http://www.it2rail.eu/ontology/isLocatedAt
ontology:isLocatedAt rdf:type owl:ObjectProperty ;

### http://www.it2rail.eu/ontology/topology#hasLocationName
topo:hasLocationName rdf:type owl:DatatypeProperty ;
   rdfs:subPropertyOf ontology:hasName ;
   rdfs:range xsd:string ;

### http://www.it2rail.eu/ontology/InteroperabilityFramework#hasStopPlaceName
interop:hasStopPlaceName rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf ontology:hasName ;
    rdfs:range xsd:string ;

### http://www.it2rail.eu/ontology/hasName
ontology:hasName rdf:type owl:DatatypeProperty ;

### http://www.it2rail.eu/ontology/InteroperabilityFramework#hasStopPlace
interop:hasStopPlace rdf:type owl:ObjectProperty ;
   rdfs:subPropertyOf ontology:hasStopPlace ;
   an:i2rumlDomain "StopPlace"^^xsd:string ;
   an:i2rumlRange "StopPlace"^^xsd:string ;

###  http://www.it2rail.eu/ontology/InteroperabilityFramework#hasTimeZoneOffset
interop:hasTimeZoneOffset rdf:type owl:DatatypeProperty ;
   rdfs:range xsd:string ;
```

Table 6: Example IT2Rail Ontology in Turtle-Syntax

6. ONTOLOGY REPOSITORY

The IT2Rail ontology is available from the Fraunhofer Institute for Transportation and Infrastructure Systems Website.[3] Registration is free and open for anyone, an email to it2rail@ivi.fraunhofer.de is required though to be granted access to the ontology files.

The ontology is available in different formats which can be translated into each other with no loss of information. Furthermore, the ontology is available as “one file” and as split ontology. The latter is split by namespace and additionally defines a root ontology as umbrella including the ontology annotations.

7. REFERENCES

[1] CEFRIEL, "Travel Expert Model v6.1," 2017.

[2] PolarSys, “Capella Solution,” [Online]. Available: https://www.polarsys.org/solutions/capella. [Accessed 01 07 2017].

[3] W3C OWL Working Group, “OWL 2 Web Ontology Language,” 11 Dec 2012. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/. [Accessed 01 07 2017].

[4] W3C, “OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition),” 11 Dec 2012. [Online]. Available: https://www.w3.org/TR/owl2-syntax/. [Accessed 01 07 2017].

[5] W3C, “OWL 2 Web Ontology Language Profiles (Second Edition),” 11 Dec 2012. [Online]. Available: https://www.w3.org/TR/owl2-profiles/#OWL_2_RL. [Accessed 01 07 2017].

[6] M. Musen, *The Protégé project: A look back and a look forward,* AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 2015.

[7] T. Tudorache, M. Horridge and J. Vendetti, "WebProtégé Users Guide," 15 Nov 2015. [Online]. Available: https://protegewiki.stanford.edu/wiki/WebProtegeUsersGuide. [Accessed 01.09.2017].

[3] https://it2rail.ivi.fraunhofer.de

[8] Object Management Group, *OMG Unified Modeling Language (UML),* 2015.

[9] D. d. Almeida Ferreira and A. M. R. d. Silva, “UML to OWL Mapping Overview An analysis of the translation process and supporting tools,” in *7th Conference of Portuguese Association of Information Systems.*.

[10] J. Zedlitz, J. Jörke and N. Luttenberger, “From UML to OWL 2,” in *Knowledge Technology. Communications in Computer and Information Science*, Berlin, Heidelberg, 2012.

[11] W3C, "OWL 2 Web Ontology Language Manchester Syntax (Second Edition)," 2012.

[12] W3C, "RDF 1.1 Turtle - Terse RDF Triple Language".