M4.1.500 Change requests on methodology to support filtering and querying of datasets

Status:	Final
Version:	1.0
Date:	6 October 2022

Document information

Project acronym	NAPCORE
Full project title	National Access Point Coordination Organisation for Europe
Grant Agreement No.	MOVE/B4/SUB/2020-123/SI2.852232
Activity no. and title	4.1.5.1 Methodology adjustments on new requirements
Author(s)	Josef Kaltwasser (AlbrechtConsult), Ian Cornwell, Victoria Kaltwasser (AlbrechtConsult)
Co-author(s)	
Related to Milestone no.	4.1.500
External Milestone	yes

Document history

Version	Date	created/ modified by	Comments
0.1	16.08.2022	Victoria Kaltwasser, AlbrechtConsult Josef Kaltwasser, AlbrechtConsult	Initial draft
1.0	06.10.2022	Josef Kaltwasser Ian Cornwell	Final version including feedback from Ian Cornwell

Action requested

☐ To be revised by partners involved in the preparation of the document

☐ For review/ approval by the Core Alignment Team

☐ For approval by the NAPCORE Steering Committee

Abstract

This report describes the result of work item 134 (including user stories 289, 210 and 211), which should determine the methodology adjustments to support filtering and querying of datasets.

Abbreviations

Abbreviation	Meaning
EC	European Commission
CAT	Core Alignment Team
NAP	National Access Point
NAPCORE	National Access Point Coordination Organisation for Europe
SC, SCOM	Steering Committee
SCS	Steering Committee Support
SWG	Sub-working Group
WG	Working Group
WP	Working Programme

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

Table of contents

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

Introduction

The following report presents the results of the work done in sprint 2022-1 and 2022-2 on work item 134, which should determine the methodology adjustments to support filtering and querying of datasets. Recent years have identified requirements to provide other access to data sources than publication of pre-defined datasets. In particular, queries via an API that include filter criteria are frequently requested. Such an approach not only requires a conceptual notion of "filtering". It may also require a different concept of entering and navigating in a model in order to provide flexible responses to queries.

Since the results of the user stories 211 "Navigation requirements" and 289 "Analyse required Methodology changes" for work item 134 are described in this report, they are now considered to be completed.

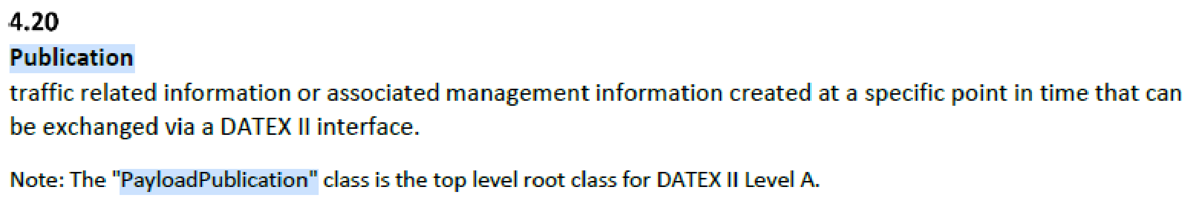
Furthermore, this report describes the results of user story 210 "Filter attributes" belonging to work item 140 "Determine methodology adjustments to support messagebrokers". User story 210 is also completed with this report, however new user stories can be created for the ongoing work item 140 considering further approaches regarding methodology adjustments to support messagebrokers.

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

DATEX II model – navigation, entry points and complex datatypes

This chapter presents the results of work done in user stories 289 and 211.

The DATEX II UML profile was created around the conceptual notion of a "publication of data". Note clause 4.20 of CEN EN 16157-1, explaining the term "Publication":



4.20
Publication
traffic related information or associated management information created at a specific point in time that can be exchanged via a DATEX II interface.

Note: The "PayloadPublication" class is the top level root class for DATEX II Level A.

This approach needs to be reconsidered. The current practice – addressed by the Note in the clause above – is that all entry points are created in the same way. There is a dedicated UML Stereotype "ModelRoot" for instances of UML "Class", that covers all metadata that should be provided together with such a DATEX II publication. In the "Common" package (CEN EN 16157-7) this Stereotype is attached to an abstract class "PayloadPublication". Common already contains one concrete instance of this, a specialisation called "GenericPublication". This class is not undisputed, because together with the Level B mechanism it allows specifying any model as Level B.

In our current data model packages, the concrete entry classes are indicated as specialisations from Common:PayloadPublication.

The three graphs of DATEX II

Once an entry point into a package is defined, the serialisation specification has a canonical graph traversal procedure to create a serial output (e.g. as XML or JSON). One reason why UML was seen as an attractive technology for data modelling was the power of graphical representation. Other standardisation domains use textual approaches based on formal languages (e.g., ASN.1 or XML Schema Definitions), and some of them – like XSD – have with growing popularity also added graphical representations. Nevertheless – based on bad experience with DATEX in the 1990ies – the choice for DATEX II was to also go for a separation of concerns, i.e., separating the abstract definition of data models with domain experts from the concrete data exchange and programming platforms. DATEX II therefore chose UML, with clearly defined mapping rules for support of potentially multiple implementation platforms.

If you open a DATEX II model with Enterprise Architect and drag all classes defined in the packages into one class diagram, you immediately see that DATEX II models are mathematical graphs, with UML Classes as nodes and other UML metaclasses as edges. You note two types of edges: Generalisations and Compositions[1]. Indeed, you could see the two subgraphs using only one type of edge each as a separate graph, the composition graph and the inheritance graph.

The inheritance graph is situated in the datatype modelling domain. It has very limited impact on the graph traversal during serialisation, but describes a way that complex types can inherit state from superclasses.

The composition graph governs the graph traversal. Serialisation starts at a model root, serialises the actual class state (potentially inherited from a superclass or explicitly specified as UML Attributes in the class itself), then it serialises the 'component classes' connected to the class via D2Relation-stereotyped UML Compositions. The use of UML in DATEX II actually doesn't specify navigability

[1] Note Bugzilla #26; the use of Compositions might have to be changed in the future anyway

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

explicitly, but it has an intrinsic notion for the D2Relations from aggregate to component for the purpose of serialisation.

When discussing our methodology with the people from the Alliance for Parking Data Standards, they claimed that this was a very restricted way of navigating through a model. They wanted to create a domain model for parking – very much as our community did for 16157-6 – but they did not want to stipulate entry points into the model, which different APIs would then be forced to use. They wanted at least the main classes of the model to be usable as entry points, and the navigability specification being sufficiently flexible to handle the resulting navigation requirements.

After that discussion it could be wondered why such requirements had so far not – at least not prominently - popped up in the DATEX II work. However, it actually did in a certain way. Even our domain modellers had found themselves in situations where they wanted relationships beyond what the hierarchical D2Relation-based composition graph could sensibly offer. What we invented then was what could be called the “hidden third graph” of DATEX II, using the “Reference” and “VersionedReference” stereotypes. "Hidden" because it is not realised in a way that would make it visible in a UML class diagrams. We rather hid it instead in a mix of a UML attribute (indicating one association end) and a Stereotype property (indicating the other end). This concept has been widely used to expose lifecycle management of DATEX II entities inside DATEX II messages, e.g. a version of one entity referring to a specific version of another entity, even though that entity may already have evolved to a newer version in the current message. While this is possible with the current Methodology, this third graph is unfortunately not visible as a graph in UML class diagrams.

Ideas

It should be discussed whether the methodology could be improved here, and whether it could be improved in a fully backwards compatible way. Note that the DATEX II methodology, right from its start, never limited the use of UML concepts not used in the UML profile yet. It would be possible to add other types of associations, that could provide new models using these features to govern a navigation for serialisation from more classes. A candidate could be of course all classes stereotyped as “Identifiable” or “VersionedIdentifiable” since these seem to have an ‘entity’ character anyway in most models. But other approaches – including new Stereotypes for UML Classes – could and should of course also be considered.

One option that has been discussed is to show the reference as a stereotyped (e.g. <<D2Reference>>) association (not composition or aggregation) to the referenced class, which turns out to be a UUID reference in the XML or JSON PSMs, and then the target class is explicit in the diagram rather than buried in metadata. That would also have the side-effect of making the logical dependency between classes and even between namespaces more explicit.

An even better option may be to draw a UML dependency from a reference attribute to the target class. That would arguably be a valid thing and a useful thing to do, but we’d need to explain it in the methodology and PSM mappings, because it’s a different kind of dependency to the ones created by the other two graphs. (At the namespace level, such logical dependencies are sometimes shown in dependency diagrams, albeit just by a UML comment.) Where the association idea above was to replace the attribute, the dependency would be additional to the attribute, in fact if EA supports this in the way it should then it would be from the attribute. This variation has the advantage that the PIM more intuitively resembles the platform-specific data where you don’t see the association directly.

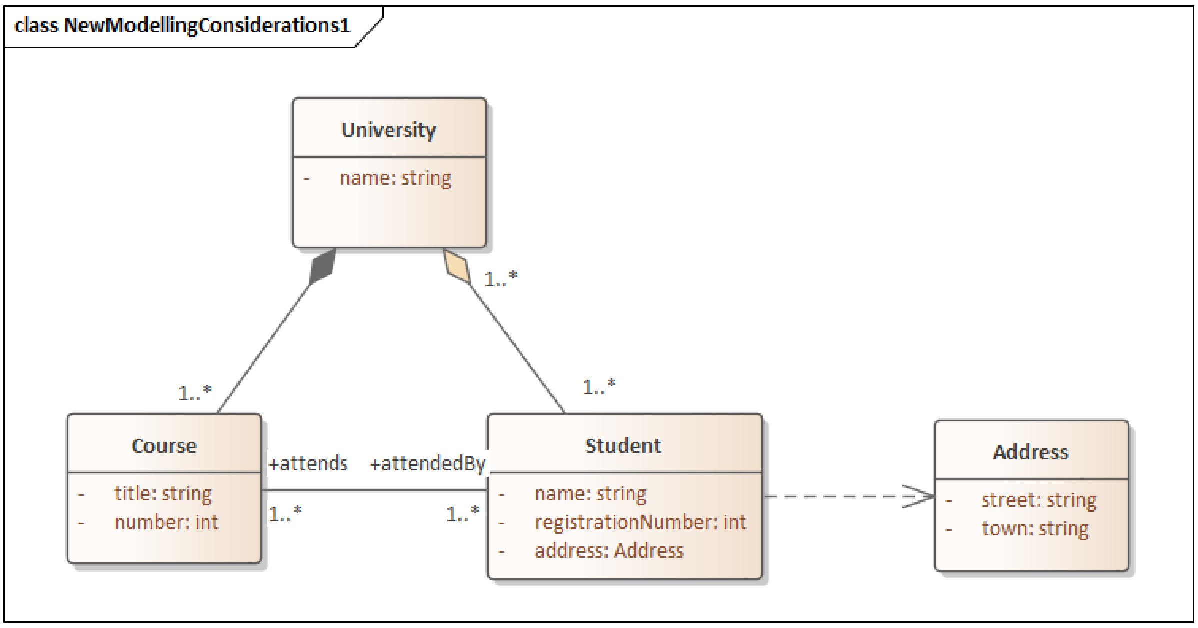
What then would be needed is also a different model traversal regime for serialisation. An API could pick one of those classes for creating the response to a request (could still be a “Publication”), and it

This project has received funding from the European Commission’s Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

could even allow parameters in the API to govern things like serialisation depth (in the sense of traversal hops allowed from the entry point to prevent endless recursion).

Example

Consider the following trivial model with a class University, having compositions to classes Student and Course, both allowing multiple occurrences. Student and Course would have an association, which by definition would have to allow multiple occurrences on both sides (describing a m:n relationship).



This diagram shows a model for university data. It was created NOT using the DATEX II Methodology. The model features in short:

- A university has one or more courses (→composition, the courses belong to the university)
- A university has one or more subscribed students. Students may subscribe to multiple universities (→aggregation)
- Students attend one or more courses, courses are attended by one or more students (→association, although an aggregation from the Course side could also be used)
- All classes have attributes with primitive data types
- The class Student also has one attribute with a composite data type: Address. It is depicted here by specifying the data type as a UML class, und using the class name as type name for the attribute. To ease understanding, we draw a Dependency relation from the using class to the defining class.

Today's DATEX II Publication would have to take a decision, let's say listing the course of a university and for each course the students. If we would want to support the receiving application in determining the course a student is attending, we might consider a "courses[1..*]" reference attribute, if the Course class is stereotyped "Identifiable".

Today's service programmers won't like this. They want an API where they give the courses and get the students, and one where they give the students and get the courses.

An API providing access to an instance of this model could offer the following services:

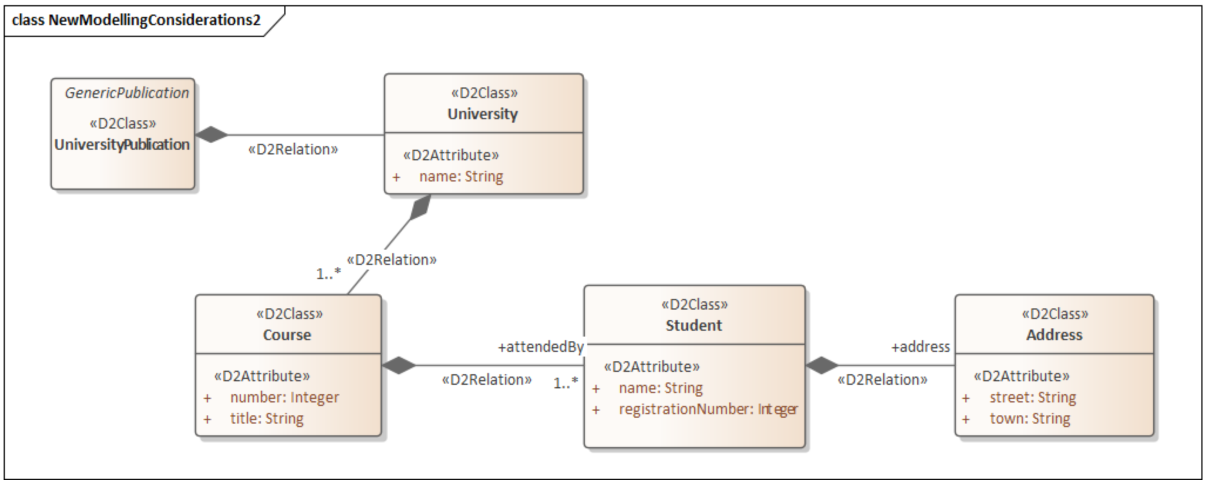
This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

1. List all courses and all students at a university.
2. For a given set of courses, list all students attending at least one of them.
3. For a given set of students, list all courses they are attending.

Question: How could DATEX II support such an API?

Stage 1

A basic DATEX II realisation of this logical model could look like this:



This example demonstrates that the DATEX II PIM is not a conceptual model but a platform independent data model: the relationship from Course to Student has become a composition where in the conceptual model it is a plain many-to-many association. When DATEX II says Student, it means "representation of a student", where that representation can be owned by a Course even if the student in the real world is not. That means that already in the PIM – although we haven't selected a PSM yet – we have some idea about limitations of the PSMs we will support, and we have used that to influence our PIM profile.

A valid instance against the schemas produced from this model would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<d2:payload xmlns:d2="http://datex2.eu/schema/3/d2Payload"
xmlns:com="http://datex2.eu/schema/3/common" xmlns:ext="http://datex2.eu/schema/3/extension"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://datex2.eu/schema/3/d2Payload DATEXII_3_D2Payload.xsd"
xsi:type="com:GenericPublication" lang="en-US" modelBaseVersion="3">
  <com:publicationTime>2006-05-04T18:13:51.0</com:publicationTime>
  <com:publicationCreator>
    <com:country>co</com:country>
    <com:nationalIdentifier>nationalIdentifier0</com:nationalIdentifier>
  </com:publicationCreator>
  <com:genericPublicationName>NewmethodologyTestInstance</com:genericPublicationName>
  <com:_genericPublicationExtension>
    <com:universityPublication>
      <ext:university>
        <ext:name>Test University</ext:name>
        <ext:course>
          <ext:number>4711</ext:number>
          <ext:title>Maths</ext:title>
          <ext:attendedBy>
            <ext:name>Jon Smith</ext:name>
```

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

```
            <ext:registrationNumber>123456</ext:registrationNumber>
            <ext:address>
               <ext:street>This street</ext:street>
               <ext:town>Somewhere</ext:town>
            </ext:address>
         </ext:attendedBy>
      </ext:course>
      <ext:course>
         <ext:number>0815</ext:number>
         <ext:title>Computer Science</ext:title>
         <ext:attendedBy>
            <ext:name>Jon Smith</ext:name>
            <ext:registrationNumber>123456</ext:registrationNumber>
            <ext:address>
               <ext:street>This street</ext:street>
               <ext:town>Somewhere</ext:town>
            </ext:address>
         </ext:attendedBy>
         <ext:attendedBy>
            <ext:name>Mary Miller</ext:name>
            <ext:registrationNumber>234567</ext:registrationNumber>
            <ext:address>
               <ext:street>Another street</ext:street>
               <ext:town>Another Town</ext:town>
            </ext:address>
         </ext:attendedBy>
      </ext:course>
   </ext:university>
  </com:universityPublication>
 </com:_genericPublicationExtension>
</d2:payload>
```

Now a current, publications based DATEX II interface could itself not provide the API services asked for, it could only provide the full dataset. Even if the publication would work with deltas, this would only be reducing the redundancy in transmission, but the client would always apply the delta information to construct the updated, complete dataset.

Considering the three example API services described above, we can make the following observations regarding a client[2] that would have to perform processing to generate the same result as the described API services would generate:

1. *List all courses and all students at a university.*

→ All courses and students are contained in a model instance, just that all students appear with all their data in all courses (generating a lot of redundancy). Processing the two sets would require the same processing as described in the next point.

2. *For a given set of courses, list all students attending at least one of them.*

[2] It is important to emphasise this word. No matter how dumb or clever a server is, no matter how well it can support the 3 queries, it gives its result to the client according to DATEX II specifications, which leaves the client the work described here.

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

→ We would have to navigate to the courses of interest, retrieve the set of students and then merge the student sets (note that we need a mathematical concept of a set avoiding duplicates).

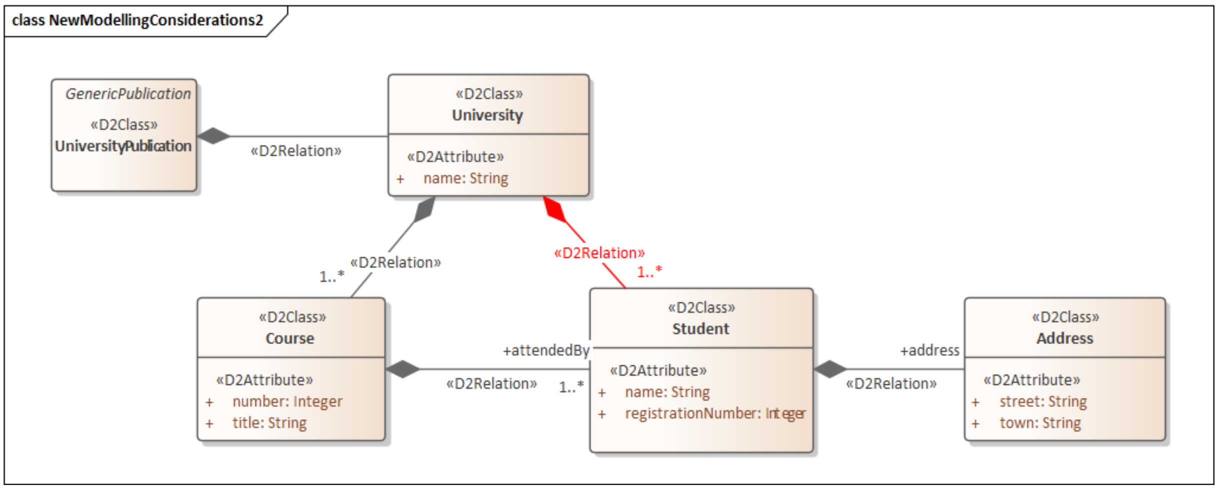
3. *For a given set of students, list all courses they are attending.*

→ We would have to iterate over a student's occurrences (e.g., found via XPath) and collect the corresponding Course instances in a set.

This is possible but implies complexity and requires resources. How much this burden is perceived as an obstacle depends on the intrinsic complexity of the receiving business logic. System of substantial complexity (traffic management centres, "big" service providers) will be able to handle this. Smaller systems like "small" service providers will complain.

Stage 2

The processing described in Stage 1 is cumbersome for a relatively easy task. The model could relieve this:



The additional D2Relation – assuming that the order property is set to "0" for Course and to "1" for Student – would now add a second sequence of Student elements in the publication. This of course increases instance size and redundancy but allows processing a list of students quickly.

The result for the three example API services would be:

1. *List all courses and all students at a university.*

→ The courses can be iterated – skipping the "attendedBy" elements – and the list of students can be retrieved by a simple iteration over a sequence of elements.

2. *For a given set of courses, list all students attending at least one of them.*

→ The same es for stage 2: We would have to navigate to the courses of interest, retrieve the set of students and then merge the student sets (note that we need a mathematical concept of a set avoiding duplicates).

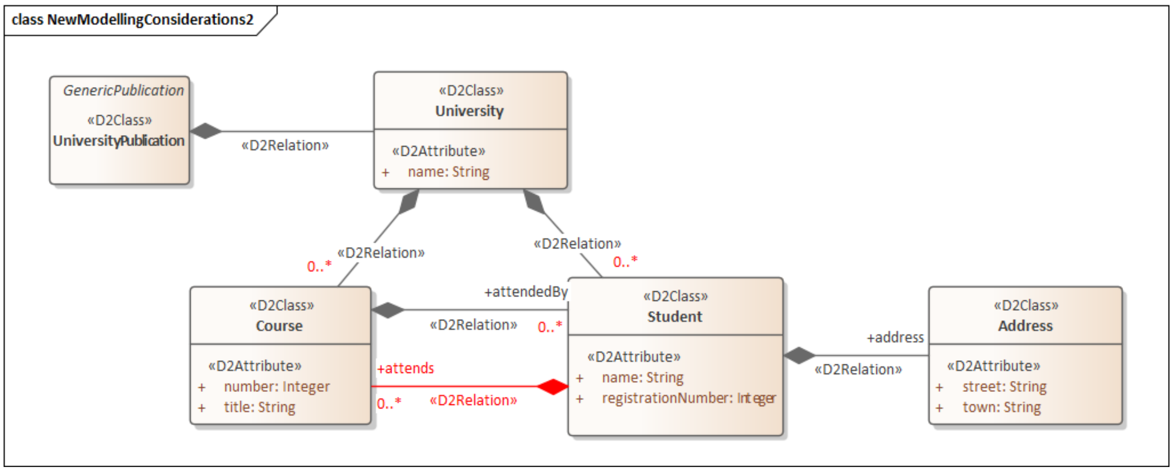
3. *For a given set of students, list all courses they are attending.*

→ The same as for stage 2 since the change does not help here. We would have to iterate over a student's occurrences (e.g., found via XPath) and collect the corresponding Course instances in a set.

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

Stage 3

To address point 3 of the example API services, another additional D2Relation is needed:



Note that the minimum multiplicities now were changed to “0” to avoid infinite recursion. This now actually allows multiple serialisation schemes, e.g., University→Course→Student versus University→Student→Course.

The following would be a valid instance for the created schemas of the above model (note I skip the unchanged ‘wrapper’):

```
<ext:university>
  <ext:name>Test University</ext:name>
  <ext:course>
    <ext:number>4711</ext:number>
    <ext:title>Maths</ext:title>
    <ext:attendedBy>
      <ext:name>Jon Smith</ext:name>
      <ext:registrationNumber>123456</ext:registrationNumber>
      <ext:address>
        <ext:street>This street</ext:street>
        <ext:town>Somewhere</ext:town>
      </ext:address>
    </ext:attendedBy>
  </ext:course>
  <ext:course>
    <ext:number>0815</ext:number>
    <ext:title>Computer Science</ext:title>
    <ext:attendedBy>
      <ext:name>Jon Smith</ext:name>
      <ext:registrationNumber>123456</ext:registrationNumber>
      <ext:address>
        <ext:street>This street</ext:street>
        <ext:town>Somewhere</ext:town>
      </ext:address>
    </ext:attendedBy>
    <ext:attendedBy>
      <ext:name>Mary Miller</ext:name>
      <ext:registrationNumber>234567</ext:registrationNumber>
      <ext:address>
        <ext:street>Another street</ext:street>
        <ext:town>Another Town</ext:town>
```

This project has received funding from the European Commission’s Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

```
            </ext:address>
         </ext:attendedBy>
      </ext:course>
      <ext:student>
         <ext:name>Jon Smith</ext:name>
         <ext:registrationNumber>123456</ext:registrationNumber>
         <ext:address>
            <ext:street>This street</ext:street>
            <ext:town>Somewhere</ext:town>
         </ext:address>
         <ext:attends>
            <ext:number>0815</ext:number>
            <ext:title>Computer Science</ext:title>
         </ext:attends>
         <ext:attends>
            <ext:number>4711</ext:number>
            <ext:title>Maths</ext:title>
         </ext:attends>
      </ext:student>
      <ext:student>
         <ext:name>Mary Miller</ext:name>
         <ext:registrationNumber>234567</ext:registrationNumber>
         <ext:address>
            <ext:street>Another street</ext:street>
            <ext:town>Another Town</ext:town>
         </ext:address>
         <ext:attends>
            <ext:number>0815</ext:number>
            <ext:title>Computer Science</ext:title>
         </ext:attends>
      </ext:student>
   </ext:university>
```

The result for the three example API services would be:

1. *List all courses and all students at a university.*

→ The courses can be iterated – skipping the "attendedBy" elements – and the list of students is also a simple iteration, skipping the attends elements.

2. *For a given set of courses, list all students attending at least one of them.*

→ The previous solution is still valid: navigate to the courses of interest, then retrieve the set of students, merge the student sets (without duplication);
now, as an alternative: iterate over the sequence of students and check whether they attend at least one of the queried courses

3. *For a given set of students, list all courses they are attending.*

→ Now solution 2. can be mirrored.

But independently from the easier processing of this complete dataset, we could also imagine an API that offers the three different services. The following depicts the different results:

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

API service 1: courses and students

```
<ext:university>
  <ext:name>Test University</ext:name>
  <ext:course>
    <ext:number>4711</ext:number>
    <ext:title>Maths</ext:title>
  </ext:course>
  <ext:course>
    <ext:number>0815</ext:number>
    <ext:title>Computer Science</ext:title>
  </ext:course>
  <ext:student>
    <ext:name>Jon Smith</ext:name>
    <ext:registrationNumber>123456</ext:registrationNumber>
    <ext:address>
      <ext:street>This street</ext:street>
      <ext:town>Somewhere</ext:town>
    </ext:address>
  </ext:student>
  <ext:student>
    <ext:name>Mary Miller</ext:name>
    <ext:registrationNumber>234567</ext:registrationNumber>
    <ext:address>
      <ext:street>Another street</ext:street>
      <ext:town>Another Town</ext:town>
    </ext:address>
  </ext:student>
</ext:university>
```

API service 2: all students for a set of course, invoked with courseSet={0815}

```
<ext:university>
  <ext:name>Test University</ext:name>
  <ext:course>
    <ext:number>0815</ext:number>
    <ext:title>Computer Science</ext:title>
    <ext:attendedBy>
      <ext:name>Jon Smith</ext:name>
      <ext:registrationNumber>123456</ext:registrationNumber>
      <ext:address>
        <ext:street>This street</ext:street>
        <ext:town>Somewhere</ext:town>
      </ext:address>
    </ext:attendedBy>
    <ext:attendedBy>
      <ext:name>Mary Miller</ext:name>
      <ext:registrationNumber>234567</ext:registrationNumber>
      <ext:address>
        <ext:street>Another street</ext:street>
        <ext:town>Another Town</ext:town>
      </ext:address>
    </ext:attendedBy>
  </ext:course>
</ext:university>
```

This project has received funding from the European Commission’s Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

API service 3: all courses for a set of students, invoked with studentSet={"Jon Smith"}

```
<ext:university>
  <ext:name>Test University</ext:name>
  <ext:student>
    <ext:name>Jon Smith</ext:name>
    <ext:registrationNumber>123456</ext:registrationNumber>
    <ext:address>
      <ext:street>This street</ext:street>
      <ext:town>Somewhere</ext:town>
    </ext:address>
    <ext:attends>
      <ext:number>0815</ext:number>
      <ext:title>Computer Science</ext:title>
    </ext:attends>
    <ext:attends>
      <ext:number>4711</ext:number>
      <ext:title>Maths</ext:title>
    </ext:attends>
  </ext:student>
</ext:university>
```

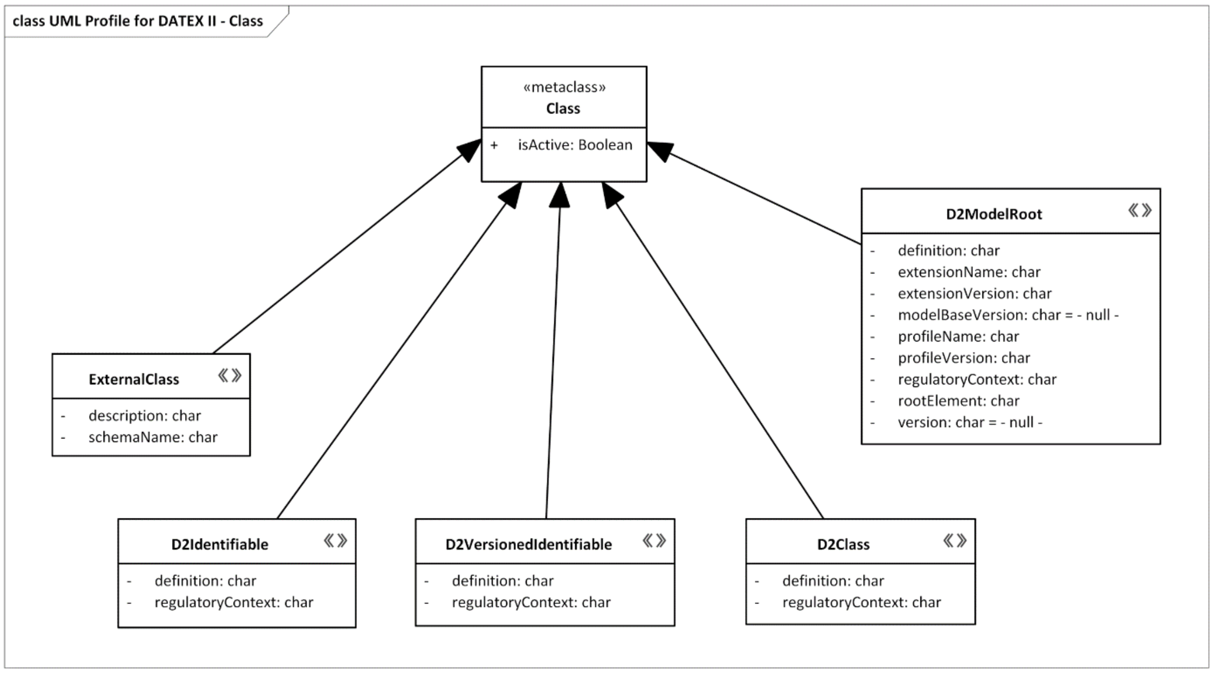
Note that the three instances depicted above all validate against the schema created by the tool.

An alternative to adding relationships explicitly to the model to produce the stage 3 model is to define generic rules about derived relationships that could be selected or omitted in implementation profiles. The text so far already recognises that we've chosen to express something beyond a conceptual model, but we could relax that in very well-defined ways. This could surely be contentious and has some issues to overcome but might be worth considering further. We would have 2 rules in mind, generalising from this University/Course/Student example (i) allow reverse directions to be implied. If your model has Foo aggregating Bars then the concept of the Foo(s) of a Bar must make sense and you may want to express a Bar with its Foo(s) (like a Student with its Courses) (ii) allow omission of intermediate classes. If your model has Foo aggregating Bar which in turn aggregates Baz, then the concept of Bazes of a Foo must make sense, and you may want to express a Bar with its Bazes (like a University with its Students, omitting the intermediate Course). We may think it is better to model that explicitly like in the Stage 3 model, or we may think we can stick to the Stage 1 model and let a profile declare that it uses the reverse association feature and the omission of intermediates feature. The case of many-to-many relationships like Student-Course is a complexity for reverse direction relationships no matter the approach.

Are all D2Class instances equal?

Further to investigating serialisation schemes and the use of the D2Relation concept, it is also relevant to see how UML classes in general – and the D2Class concept in particular – are actually used in models. The basic concepts available in the DATEX II v3 Methodology are:

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223



ExternalClass is the class variation of the general concept for all metaclasses to allow external models being referred, so we can simply discard this here.

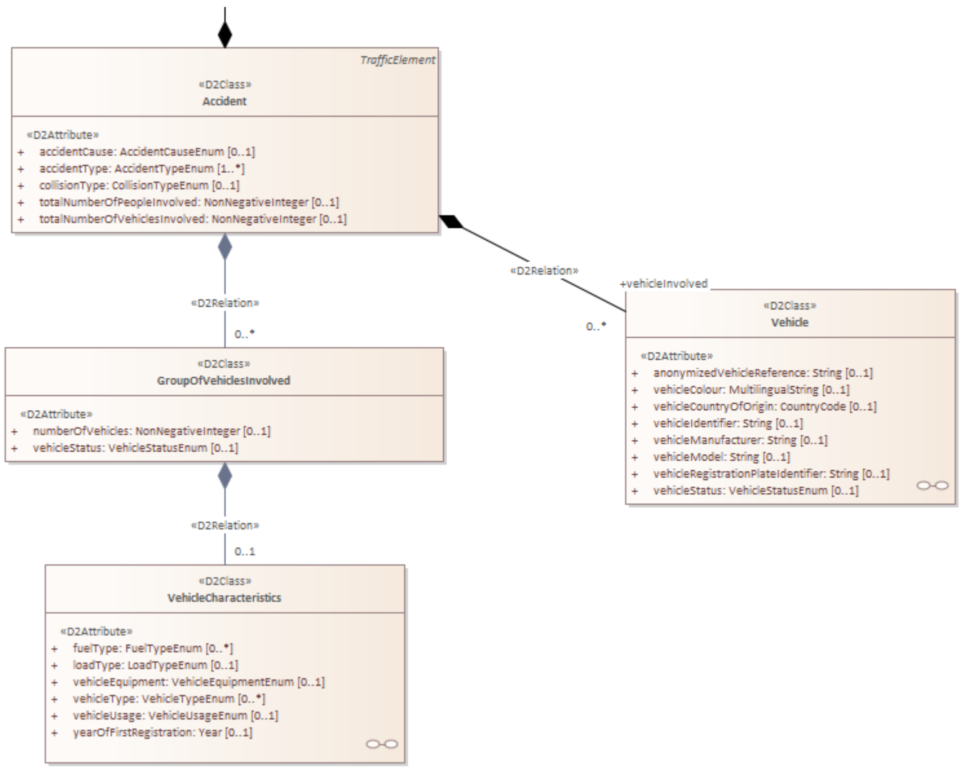
D2ModelRoot is the concept of publication entry point described earlier. It is actually implemented by the abstract base class "PayloadPublication" in the "Common" package (16157-7), which in itself – together with "LocationReferencing" (16157-2) – is the basis of all other publication models, so the stereotype is actually not used anywhere else, except for some Level C models created outside the DATEX II community.

D2Identifiable, and **D2VersionedIdentifiable** are probably the closest in the concept of indicating “entities” or “objects”, since they at least convey the concept of an “id” attribute, providing identity. The “versioned” version only adds an element of versioning status changes. Both carry exactly the same properties as D2Class, they are just hints for the code generator to provide them with mechanisms for giving identity, and also versioning in the latter case.

D2Class is the main use of UML classes in the DATEX II v3 Methodology. What is interesting to note is that current models seem to use the concept “D2Identifiable” rather sparingly. This is understandable from the perspective of a publisher since this is the basis for referencing via attributes. Hence, the publisher must create and manage referenceable, unique "id" attributes for each instance of such a class. Beyond that, there is no further distinction in how to use UML classes via the D2Class concept. But – as depicted in the next diagram taken from the Situation package (16157-3) – there seem to be different levels of concepts using this stereotype.

While some uses of D2Class, e.g., "Vehicle", would rather look like some kind of entity – a bit like "Course" and "Student" in our examples - others like "VehicleCharacteristics" seem simply to be a composite data type. So indeed, DATEX II v3 seems to have an entity concept only for ‘heavyweight’ entities. Relations to ‘lightweight’ entities and composite data types share the same concept (D2Relation with a D2Class as component). We might consider making that distinction clearer in the future, e.g., by denoting composite data types as classes (maybe in future with a dedicated stereotype D2CompositeDataType, however this is not needed if entities are distinguished by some other stereotype) with the class name used as attribute type and the classes depicted and connected via a dependency relation, as is done e.g. in TPEG standards.

This project has received funding from the European Commission’s Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223


TrafficElement
«D2Class»
Accident
«D2Attribute»
+ accidentCause: AccidentCauseEnum [0..1]
+ accidentType: AccidentTypeEnum [1..*]
+ collisionType: CollisionTypeEnum [0..1]
+ totalNumberOfPeopleInvolved: NonNegativeInteger [0..1]
+ totalNumberOfVehiclesInvolved: NonNegativeInteger [0..1]
«D2Relation»
0..*
«D2Class»
GroupOfVehiclesInvolved
«D2Attribute»
+ numberOfVehicles: NonNegativeInteger [0..1]
+ vehicleStatus: VehicleStatusEnum [0..1]
«D2Relation»
0..1
«D2Class»
VehicleCharacteristics
«D2Attribute»
+ fuelType: FuelTypeEnum [0..*]
+ loadType: LoadTypeEnum [0..1]
+ vehicleEquipment: VehicleEquipmentEnum [0..1]
+ vehicleType: VehicleTypeEnum [0..*]
+ vehicleUsage: VehicleUsageEnum [0..1]
+ yearOfFirstRegistration: Year [0..1]
«D2Relation»
+vehicleInvolved
0..*
«D2Class»
Vehicle
«D2Attribute»
+ anonymizedVehicleReference: String [0..1]
+ vehicleColour: MultilingualString [0..1]
+ vehicleCountryOfOrigin: CountryCode [0..1]
+ vehicleIdentifier: String [0..1]
+ vehicleManufacturer: String [0..1]
+ vehicleModel: String [0..1]
+ vehicleRegistrationPlateIdentifier: String [0..1]
+ vehicleStatus: VehicleStatusEnum [0..1]

Analysis

The current Level A models use the D2Relation concept (i.e., the fact that we only have composition as relation type) in a way that is obviously inspired by the conceptual notion of a "publication". However, this is not described anywhere, maybe because it has been perceived at the time DATEX II was developed as so obvious that there is no need for description. It may be described as the "Publication Paradigm" roughly as follows:

A publication covers a set of information Items in the governance of one publisher. The publisher determines elements of scope, in particular the geographic or network related scope. He selects the suitable domain model (= DATEX II content standard part). Ideally, he uses a suitable profile of this standard, extended by a Level B extension where necessary. If we limit this description to snapshot data: he creates – based on his updating policy either periodically or on-occurrence – a valid instance of the profile model containing the current state of all currently existing information items.

The model describes the entry points (typically one or few), and the use of the D2Relation in the model creates a canonical serialisation scheme. The modeller has to decide on this scheme, as is depicted in the *Stage 1* section of this document. As shown in the example, this choice has significant impact on client-side processing complexity. This can in the end lead to clients rejecting the publication because the complexity to structure the data in a way digestible to their business logic may seem inappropriate for the client-side setup.

As shown in *Stage 2* and *Stage 3*, additional model elements could be provided that would enlarge the applicability of the received data on client side by reducing client-side processing complexity. It is worth noting that this has so far all been based on the existing (DATEX II v3) Methodology, i.e., without adding additional elements to the UML profile. It should be also noted though, that existing multiplicities had to be changed in these steps in order to allow alternative coding strategies. This would imply that documentation of such models would require more elements about actually applying a model, which so far was more or less canonical. Furthermore, it might be better to abandon the current practice of displaying all relations in diagrams, but rather create different

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

diagrams that describe different serialisation schemes. Since this approach allows for different applications of one model, APIs based on such a model could indeed provide different publication results for different services, even with the identical state of the information resource underneath, and even based on the same schema created by the DATEX II v3 tool, as shown in the example.

It is of course worthwhile to consider providing different UML elements, not governed by the DATEX II v3 Methodology yet, to achieve the same effect. Pros and Cons of these two alternative approaches will have to be analysed in depth.

It should also be noted that both approaches can be applied in a backwards compatible way, in the sense that all DATEX II v3 compliant models remain compliant in v4.

How far current models could be extended and improved by this approach – and whether indeed real APIs could be created based on this – is still to be tested with real models.

Beyond the navigability in the composition graph, it also seems wise to use a concept for complex data types that is clearly distinguishable from entity-type classes and their relationships. Allowing total flexibility in directions of navigation and entry points would avoid needing any distinction in our UML profile but would create the possibility of fragments that wouldn't have enough context to make sense. The complex types that hold properties of entities will still be useful in formulating queries, but they won't be useful enough as results on their own. In OO design terms, they are value objects. Distinguishing entities from value objects is a well-known principle, e.g. of Domain Driven Design related principles in CORBA in the 90s. Examples exist also in other ITS standard – e.g., in the ISO standards for TPEG - so such an approach would also create more alignment on Methodology level. This would require refactoring of the Methodology but could also be introduced in a backwards compatible way.

Proposals for 2023 work program

1. Test suitability of the proposed modelling options for DATEX II v3 Methodology by refactoring existing models and defining APIs – analyse which type of serialisation parameters would be need and how they would need to be exposed / documented
2. Analyse options for using new UML stereotypes in DATEX II v4 Methodology to better depict such API options and corresponding model navigation / serialisation patterns
3. Propose a backwards compatible proposal for better handling of “value objects” (complex datatypes)

This project has received funding from the European Commission’s Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

Filter attributes

Why consider Filtering in DATEX II Content Methodology?

This chapter presents the results of work done in user story 210 (*Filter attributes*) in work item 140 (*Determine methodology adjustments to support messagebrokers*). When the DATEX II Content Methodology had been developed in the early Millenium years, the main targeted data exchange technologies where request/response-based: either plain http or SOAP on top of http. The basic notion was that of an information resource, whose state (snapshot) or recent state changes (delta) where published. While these technologies on the communication layer are intrinsically "pull" oriented – with the publisher taking the server role – it is not uncommon to switch the client and server role sometimes, depending on concrete use case requirements. With the publisher taking the client role in communication, it can instantiate communication as soon as new data is available, hence reducing latency of the information flow.

Recent years have seen new exchange technologies becoming more and more successful based on the message broker exchange pattern (e.g. AMQP and MQTT). The way information is exchanged here is based on the conceptual notion of a topic, where interested parties could subscribe to receive news on that topic. The concept is intrinsically "push" oriented. One or more potential sources of news push them to the corresponding topic, the messagebroker transfers them to subscribed clients. At least some of these technologies offer the concept of a filter, so clients could decide whether they want to receive all or just specific news published on a topic they are subscribed to. So, while work item 140 *(Determine methodology adjustments to support messagebrokers*) will analyse all aspects of a potential support of this data exchange approach, user story 210 focussed on analysing the concept of *Filter attributes*, and whether and how DATEX II Content Methodology could or should support such filters.

Ideas

The two topics identified by the teams as being relevant for the user story "Filter attributes" are tiling (= using geographic tiles as broker topics) and the attribute-based concept of Apache filters (as they are available for AMQP message brokers). This might - at least potentially - create needs for features in Methodology, e.g., properties in the UML Profile to indicate attributes that could/should be used for filtering. The ideas are currently pursued by the backend variant of hybrid C-ITS specification developed in C-Roads.

In the following, AMQP v1.0 was considered - not because it is yet agreed that we will develop an exchange specification for that, but because as the current version (also an ISO/IEC standard) and having legacy support for v0.9 filters, it seems reasonably indicative of the kinds of exchange features we might support. In AMQP a filter is effectively a Boolean-typed function filter(message), and it can't alter the content of a message, it only determines whether a whole message should pass. The details of specific filters are outside the standard and defined in a registry. The two relevant registered kinds both come down to matching values in the properties sections of a message, which are different to the application-data section where the DATEX II payload would appear. So, in an implementation of a service, you have to know what attributes you would allow filtering on and repeat them in a property. The key question for this user story 210 becomes: does the range of good possibilities for filter attributes belong with a content model or only with specific publication services? If we thought the latter, we need take no action in the content methodology. An AMQP exchange specification at the general level would presumably allow any DATEX II content to be placed into the filterable properties and then for specific implementations or profiles one would decide and specify what values are going to be set in filterable properties. If we thought the former,

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

we could potentially increase the efficiency of achieving interoperability by denoting certain attributes as (say) <<filterable>>, and an AMQP exchange specification would say that only these can (or all these must?) be placed into filterable properties in the AMQP message. However, it should be discussed if we should constrain the range of filterable attributes in the content model specifications. Consider the Common namespace for example. One could probably imagine some scenario where almost any attribute could be a useful filter attribute. So, this consideration has not led to propose any new methodology features to support these kinds of filters - we can leave that to more specific service specifications. Query APIs that can select the level of data to be returned are another matter, discussed in the previous section, where we may benefit from new methodology features, but AMQP does not provide that kind of facility.

It could be discussed if we need a split into 'core Methodology' and 'extended Methodology' (which includes elements for harmonisation like profiles, etc. - where harmonised filtering principles could be placed). We could define something like <<filterable>> with the expectation that it should never be used in the base models, but it could be used in profiles.

Another point to be considered is the question whether today's perception of a DATEX II publication will fit into a message broker paradigm. The concept of topics in typical messagebroker setups typically has finer granularity than the usual shaping of publications in DATEX II interfaces. Looking into the various well-established DATEX II content models like situation, VMS or elaborated data, you typically find a hierarchy publication→container→objects. Even if we would choose one topic for the content of a publication, a typical messagebroker setup would probably at least make the single objects 'messages' published on this topic.

We have received reports that DATEX II users had transported DATEX II via message queues in one service where whole SituationPublications were transported, whereas in another service transport VmsUnit objects were transported individually, without any enclosing table publication. This observation relates this user story to what has been discussed regarding entry points in the scope of user story 211 (*Navigation requirements*) discussed in the previous section.

A message queue PSM should define what entry points are permitted. The case mentioned above suggests that we should not limit this to <<DATEX II ModelRoot>> objects. The PSM could allow any DATEX II object (in which case no methodology support would be needed). Or the PSM could allow a specific set of objects denoted by a stereotype – perhaps a new one (which would need a methodology revision) or perhaps (as discussed in the previous section) we could use existing ones like <<D2Identifiable>> and <<D2VersionedIdentifiable>>. However, VmsUnit (from v2.X) is not either of these stereotypes, suggesting that idea as such is not sufficient. Nevertheless, with the concept of data values discussed in the previous section, it might be suitable to allow all (remaining) entity classes and juts discard the value object classes.

Note that the concept of allocating topics according to a geographic tiling system – which would make sense for single objects messages – would need a specification for matching the concepts of Part 2 (Location Referencing) to rectangular tiles. It may be simple for (most) point locations, but nontrivial for linear and area locations.

Analysis

It seems that using the messagebroker concept for the typical DATEX II publication of today is possible (maybe to organise technical aspects like scalability and subscription management) but does explore the full potential of today's messagebroker applications. To do so would mean to allow publishing single instances of classes found in a model. This therefore relates very much to the discussion regarding entry points for APIs, together with the corresponding model navigation and

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223

serialisation aspects, described in more depth in the previous section. If we had this notion of entering a model in various places and serialising objects from there, we could use this not only for query APIs but also for publishing these objects on corresponding messagebroker topics. We would certainly still need some kind of 'information management wrapper'- even around a single, potentially small object – but whether that is something new or just a refactoring of the current structures is to be seen. Nevertheless, the concept of publishing objects via messagebroker topics should be analysed together with further work on APIs. We do assume that these developments require more guidelines on how to use things produced by the DATEX II core Methodology, so the concept of an additional extended Methodology should be pursued further.

Proposals for 2023 work program

4. Analyse API aspects of model entry point, serialisation specification and filter attributes not only from a query API perspective, but also from the perspective of a messagebroker API.
5. Consider complementing the current Content Methodology (to be called "Core Methodology") with and "Extended Methodology", where guidelines for proper using of the increasing flexibility of DATEX II v4 will be captured.
6. If messagebroker APIs based on single entity messages are pursued further, consider specifying matching strategies foe DATEX II location references to rectangular geographical tiles.

This project has received funding from the European Commission's Directorate General for Transport and Mobility under Grant Agreement no. MOVE/B4/SUB/2020-123/SI2.85223