Proceedings of 7th Transport Research Arena TRA 2018, April 16-19, 2018, Vienna, Austria

ST4RT – Semantic Transformations for Rail Transportation

Alessio Carenini[a], Ugo Dell'Arciprete[b], Stefanos Gogos[c], Mohammad Mehdi Pourhashem Kallehbasti[d], Matteo Rossi[d], Riccardo Santoro[e]

[a]*CEFRIEL, 2 via R. Fucini, 20133, Milan, Italy*
[b]*Hit Rail, 4 Leidseveer, 3511 SB Utrecht, The Netherlands*
[c]*UNIFE (European Rail Industry), 221 Avenue Louise, Brussels 1050, Belgium*
[d]*Politecnico di Milano,Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza L. Da Vinci 32, 20133, Milan, Italy*
[e]*TRENITALIA, 1 Piazza della Croce Rossa, 00161 Roma, Italy*

Abstract

The lack of interoperability between different transport modes is a barrier to the modal shift towards green modes of transport such as rail, on its own or as a backbone of an integrated mobility system, as well as the deployment of a Single European Transport Area. The ST4RT (Semantic Transformations for Rail Transportation) project aims at developing a demonstrator for a component that allows services relying on different standards to interoperate. This is achieved through the implementation of techniques for the automated transformation of messages across standards which are based on semantically-rich data. This paper presents some of the results achieved by the ST4RT project, and delineates its forthcoming contributions. ST4RT will contribute components to the Interoperability Framework which is at the core of the fourth Innovation Programme (IP4) of Shift2Rail.

1. Introduction

The current lack of interoperability between different transport modes limits the modal shift towards green modes of transport such as rail, on its own or as a backbone of an integrated mobility system, as well as the deployment of a Single European Transport Area. An ecosystem for common travel products or services could be created, where different operators are able to cooperate, and the complexity perceived by the travelers is masked by an interoperability framework that addresses, on the one hand, the diversity and variability of standards and protocols of different transport modes managing data coming from different environments, and on the other hand sector-level advances in interchange formats and protocols such as Telematics Applications for Passenger Services Technical Specifications for Interoperability (TAP-TSI[*]) or emerging sector-driven initiatives.

The primary objective of the ST4RT (Semantic Transformations for Rail Transportation) project[†] is to develop a demonstrator tool that will provide ontology-based transformations between different standards and protocols, resulting in enhanced semantic interoperability between disparate, heterogeneous legacy systems.

[*] http://www.era.europa.eu/Document-Register/Pages/TAP-TSI.aspx

[†] ST4RT (http://www.st4rt.eu) is a two-year project that started in November 2016, and whose total budget is 1Meuro.

Such transformation technology is essential to achieving the goals for the Interoperability Framework that is at the core of the fourth Innovation Programme (IP4) of Shift2Rail[‡]. The Interoperability Framework will provide the right tools in order to introduce seamless mobility services, foster the development of multi-modal travel services and help to overcome the obstacles currently impeding the development of market innovation and limiting a large acceptance of the semantic web for transportation. The main technical ambition of the Interoperability Framework is "the provision of the tools and technologies that will allow data exchange among different actors of the transport ecosystem, providing mechanisms to abstract data consumers from the complexity of varied data formats and non-integrated services, facilitating interoperability among systems and the creation of added value services for achieving a seamless multimodal door-to-door experience". In line with the objectives stated in IP4, such transformation technology is necessary to assure that technical interoperability can be deployed effectively and cost-efficiently by market actors in order to create service offerings that substantially improve mobility. Such transformation technology is a powerful tool that will allow us to meet the challenge to overcome the complex misalignment of eco-system services due to differences in business models and legacy systems.

The Semantic Transformation, as an element of the underlying Interoperability Framework technologies and standards as developed in IP4, will address fundamental obstacles to interoperability by overcoming incompatibility. This technology, if widely deployed, will be able to:

- overcome the fragmentation of multiple data formats and communication protocols;
- connect multi-modal providers and the services sectors;
- lower the cost of accessing data that is openly discoverable but of low quality or availability;
- maximise growth potential for the development of new products and services by removing ICT systems incompatibility, e.g. enabling market players and new entrants' capabilities, thereby reducing the cost and time-to-market for the ICT integration.

The rest of this paper presents the approach that is being developed in the ST4RT project to allow services based on different standards for the representation of data to communicate and interoperate. More precisely, Section 2 provides an overview of the approach, whereas Section 3 briefly introduces the annotation-based mechanisms that are at the core of the approach. Section 4 presents the architecture of the prototype that the project is developing, and Section 5 provides an overview of the demonstrator which will integrate the developed architecture in an existing conversion tool, in order to show how the transformation process can be applied to a practical use case. Finally, Section 6 concludes.

2. Overview of the ST4RT approach

Interoperability refers to the ability of devices or systems to participate in the coordinated execution of tasks and functions in some business process, in which data/information exchange happens in a transparent way, according to common protocols and formats. In fact, interoperability is achieved when the partners involved in the data/information exchange agree on the common information model and on the interfaces between their systems.

While interoperability can be achieved at the "syntactic" level, a full interoperability should be sought at "semantic" level – i.e., when interoperating systems are able to interpret the exchanged information automatically and meaningfully, so that exchanged messages are unambiguously defined and understood by the different parties.

A common situation in any application interoperability scenario is the following: some fundamental assumptions about the data and their meaningful use underpin integration (or, more generally, interoperability) efforts; these assumptions are held implicitly and often informally by humans and can only be controlled when some forms of "sharing" of these assumptions can be established. The fundamental obstacle to interoperability becomes the cost of achieving and maintaining this sharing effectively, which requires a form of control on the individuals that is typically exercised through organisational means. However, the scope of control, and therefore of sharing of the assumptions, usually remains 'local' to a company or to a group of cooperating companies: the result is the increased cost of setting up and maintaining the cooperation and a reduced flexibility in evolving the common solutions.

[‡] https://shift2rail.org/wp-content/uploads/2013/07/S2R-JU-GB_Decision-N-15-2015-MAAP.pdf

We address these issues through the creation of an explicit, formal, shareable, machine-readable and computable description of the semantic reference model associated with data descriptions and exchanges; the ultimate goal is to allow for a higher degree of automation of distributed processes across multiple data formats and spanning unspecified actors. These formal, explicit, machine-readable descriptions, in the form of domain ontologies, are maintained, stored and made accessible by the ontology repository provided by the Interoperability Framework and used to 'annotate' web services in the Interoperability Framework's semantic web service repository.

According to the classification proposed in (Vetere, 2005), state-of-the-art approaches to semantic interoperability can be distinguished according to two fundamental dimensions: the **data schema mapping** (*any-to-any* vs. *any-to-one*) and the **integration logic** (*centralized* vs. *decentralized*). As demonstrated in (Della Valle, 2007), the any-to-one centralized approach to semantic interoperability appears to be powerful and well-suited for managing complex and dynamic environments, like the one analysed by ST4RT, where a common shared reference ontology is available. An extended analysis of the state-of-the-art approaches to semantic interoperability is available in (ST4RT Project, 2017).

In Figure 1, we summarise the general approach for semantic interoperability that we are adopting in the ST4RT project. Two different standards A and B are semantically annotated in order to create mappings from their data models to the global reference ontology – currently represented by IT2Rail ontology, but possibly extended to a network of interrelated domain ontologies. The resulting mappings are the basis for the semantic transformations of ST4RT, so that data expressed in the two standards can be converted into their respective ontological version.

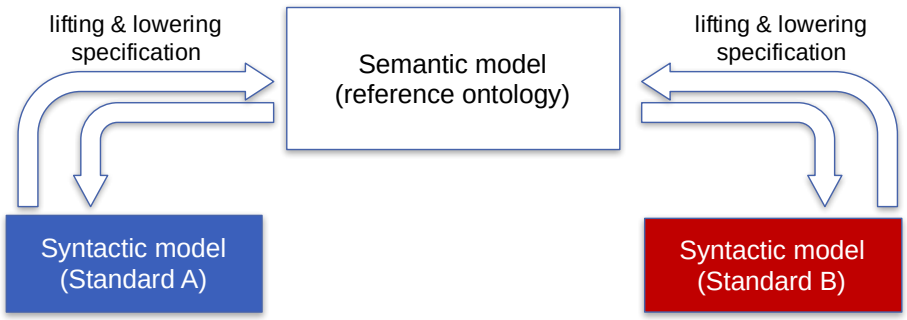

Fig. 1 ST4RT approach to semantic interoperability

Whenever two systems that adopt the different standards A and B need to exchange information, the semantic transformation takes place. As in the depicted example, a message originally expressed with regards to standard A – because it is generated by a system adopting standard A – is "lifted" to its ontological version, by means of the mapping between standard A and the reference ontology; once in its ontological counterpart, the message can then be "lowered" to a message expressed with regards to standard B, by means of its respective mapping to the reference ontology, and can be consumed by the target system. Of course, the same procedure can be followed when the message exchange is initiated by the system adopting standard B: a message expressed in standard B is first lifted to the ontological version and then lowered to a message expressed in standard A to be consumed by the target system.

Moreover, what is valid for message exchanges can be also applied to web service descriptions: a service interface exchanging data expressed in standard A can be described in a machine-readable way and its description can be lifted to the ontological level – by means of the corresponding mapping – and then lowered to a description expressed in standard B – also by means of its corresponding mapping – and so on.

The value of this semantic interoperability approach lies in the lifting to and lowering from the ontological level; this allows for a meaningful exchange between the various participating systems which, by means of their semantic annotations and mappings, agree and commit to the conceptual level of the communication, instead of the syntactic level of message exchange. This approach avoids and saves the costs associated with the creation and maintenance of several point-to-point conversions that, with the increase of the number of participant systems and their respective data models, quickly become unmanageable. On the contrary, with the adoption of semantic interoperability technologies, only the mappings to the ontological level must be maintained and aligned over time.

3. Annotation-based mapping between standards

This section presents some of the technical details of the annotation-based approach pursued by the ST4RT project to realize the mapping mechanisms described in Section 2. The overarching idea is to annotate legacy data representations to be able to match the legacy concepts to those of the reference ontology, and vice-versa. In the ST4RT approach, the assumption is that legacy data representations and the corresponding data instances are given in terms of standard technologies, such as XSD (XML Schema Definition, (Gao, 2009)) and XML (eXtensible Markup Language, (Bray, 1998)). Figure 2 shows the lifting/lowering process in the case where for both the source and target standards the data representation is defined through XSD files. Notice that the reference ontology will usually be an integration of several ontologies, each defining a separate set of concepts.

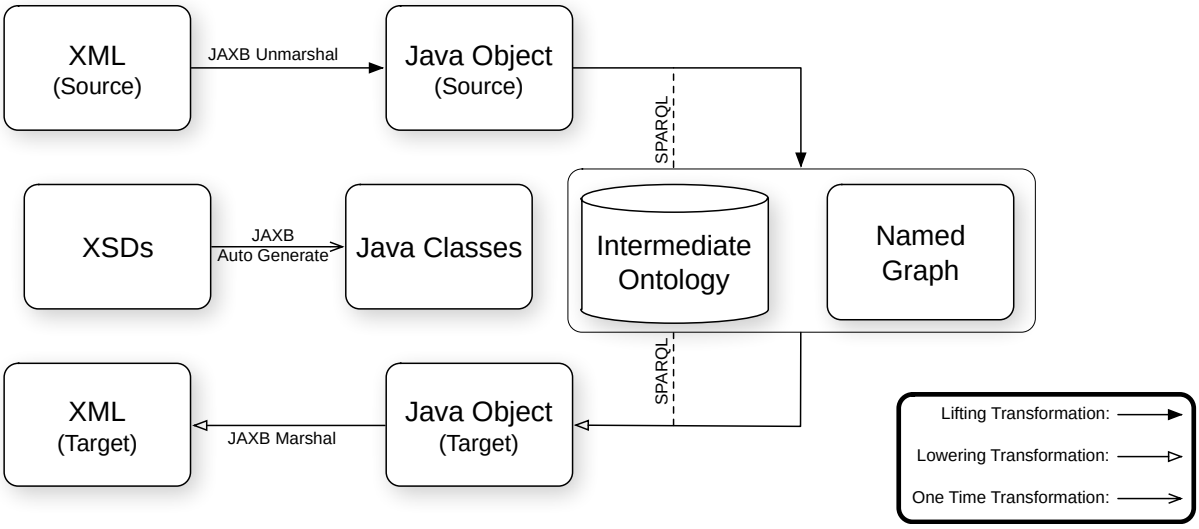

Fig. 2 Approach workflow in the case where source and target standards are defined through XSD files as in the case of FSM/TAP TSI

The starting point of the approach is that data schema definitions given in terms of XSD files are first translated into declarations in the Java programming language (Arnold, 2005) (this can be done automatically through existing frameworks such as JAXB[§]). Then, the declarations of Java classes, methods and attributes are annotated to match them with concepts in the reference ontology; for example, Java classes, which capture data types defined in XSD files, are usually matched with classes of elements in the ontology. The annotated Java code is used to produce, during the lifting phase at runtime, sets of RDF (Resource Description Framework (Lassila, 1999)) ⟨*subject*, *predicate*, *object*⟩ triples (that is, a *named graph*) conforming to the reference ontology, starting from XML files conforming to the XSD definitions of the source standard. The Java-based annotations are also used to produce, during the lowering phase, XML files obeying the XSD definition of the target standard from the named graph created during the lifting phase.

The annotations defined in the ST4RT approach are compatible with (and extend) those defined in the Pinto framework[**] for mapping Java Beans to RDF triples and vice-versa. In the rest of this section we present first the lifting, then the lowering annotations, with some examples of their usage. In general, in the ST4RT approach annotations in Java files can be associated with various elements of the Java code, and in particular classes, methods, fields, and parameters. As mentioned in Section 5, the reference case study that is currently considered in the ST4RT project involves an exchange of messages between a service supporting the Full Service Model (FSM) standard[††], and one supporting the TAP-TSI 918 XML standard. Hence, the examples presented in this section concern the lifting of FSM messages and their lowering to TAP-TSI 918 XML messages. Let us remark that the TAP TSI Regulation, as referenced in Section 1, defines messages in bit-oriented format, not suitable for the translation into declarations in Java. We therefore use for ST4RT an XML version of the same messages,

[§] https://jaxb.java.net

[**] https://github.com/stardog-union/pinto

[††] https://tsga.eu/fsm

defined in the UIC leaflet 918.1‡‡. It is likely that the XML version will be added as an optional alternative to the bit oriented one in a next revision of the TAP TSI Regulation.

Let us first consider, as a reference, the use of annotations to *lift* concepts from a legacy standard to the target integrated ontology. *@NamedGraph* is a class annotation that defines a named graph with a type, and a URI (Uniform Resource Identifier). This annotation is used to tell the conversion mechanism to what named graph the RDF triples should be added. The *@NamedSpaces* class annotation, instead, is used to identify the ontologies from which the concepts used in the mapping are taken.

Java classes are mapped to entities in the corresponding named graph using the class annotation *@RdfsClass*. For example, the following annotation maps the *ContactInformation* concept from the FSM standard (for which the JAXB framework creates a Java class with the same name) to the *Passenger* class of the ontology.

```
@RdfsClass("customer:Passenger")
public class ContactInformation extends FSMID
```

@RdfProperty, instead, is a field annotation, where the annotated field is mapped to the object (i.e., the third element) of an RDF triple, and the subject (i.e., the first element of the triple) is the entity related to the Java class in which the field appears.
The *@RdfProperty* annotation is used when there is a direct mapping between a value in the incoming message and a property (i.e., a predicate) in the ontology (i.e., when the incoming value must appear *as is* in the named graph). However, it can happen that the values to be stored in the named graph must be built from the incoming message by piecing together information from the incoming message, but also from the ontology. This is achieved through queries in the SPARQL (Harris, 2013) language, to be performed on the ontology. More precisely, *@Sparql* is a field annotation where, similarly to the *@RdfProperty* annotation, the mapping target is the object of an RDF triple, but the value is the result of a SPARQL query. A *@Sparql* annotation has attributes *name*, *inputs*, *outputs*, and *type* (some of them are optional); it can appear also as a method annotation for lifting a single attribute, as a parameter annotation for lowering a single attribute and as a class annotation for lifting/lowering multiple attributes (as explained below). Figure 3 shows an example of *@Sparql* annotation applied to attribute *country*; the result of the query is mapped to the object of an RDF triple whose subject and property are, respectively, an instance of *customer:Passenger* (since *ContactInformation* is the enclosing class) and *st4rt:isInCountry_Alpha2*.

```
@RdfsClass(
  "customer:Passenger"
)
public class ContactInformation
  extends FSMID {
[...]

@Sparql(
  name = "getAlpha2ByName",
  inputs = {"country"}
)
@RdfProperty(
  "st4rt:isInCountry_Alpha2"
)
@XmlElement(name = "Country")
protected String country;

[...]
}
```

```
<query>
  <name> getAlpha2ByName </name>
    <inputs>
      <input> input1 </input>
    </inputs>
    <comment>
      Input: Country name (??input1)
      Output: ??input1's Alpha2 code
    </comment>
  <sparqlquery><![CDATA[
PREFIX st4rt: <http://www.semanticweb.org/Mehdi/ontologies/2017/2/st4rt#>
PREFIX countries: <http://www.bpiresearch.com/BPMO/2004/03/03/cdl/Countries#>
SELECT ?alpha2
  WHERE {
    ?subject countries:countryCodeISO3166Alpha2 ?alpha2 ;
    countries:countryNameISO3166Short ??input1 }
  ]]></sparqlquery>
</query>
```

Fig. 3 Lifting @Sparql example and corresponding SPARQL query

The *@Sparql* annotation indicates that the value must be fetched through the *getAlpha2ByName* SPARQL query, whose definition is shown in the right-hand side of Figure 3. Notice that when the input parameter of the query (i.e, *input1*) is used in the query, its name is prefixed by *??*. When the query is run for attribute *country*, parameter

‡‡ http://www.shop-etf.com/

??input1 must be replaced by the value of the *country* attribute before the SPARQL query is run. For example, if "Italy" is the value of *country* attribute, the following RDF will be produced, as "IT" is the result of the query:

:contactInformation st4rt:isInCountry_Alpha2 "IT"

The annotations presented above are also used in the *lowering* phase, to create instances of concepts according to the target standard. In this case, the annotations are associated with so-called "setter" methods, which are used by the JAXB framework to create instances of XML files from Java objects.

The annotations introduced so far are used to lift/lower one single concept at a time. The ST4RT approach allows also for the possibility to lift/lower several concepts at the same time, to avoid having to run separate SPARQL queries to retrieve values when only one would be enough – which could be beneficial from the point of view of the efficiency of the mapping process. In this case, the *@Sparql* annotation should be applied to a whole class, rather than a single attribute (as was the case in the example of Figure 3). Figure 4 shows an example of a *@Sparql* annotation that lowers attributes *gender* and *country* together using the *LoweringContactInformation* query.

```
@Sparql(
  name =
   "LoweringContactInformation",
  outputs = {"gender", "country"},
  type = QueryType.Lowering)}
)
public class ContactInformation
  extends FSMID
```

```
<query>
  <name> LoweringContactInformation </name>
  <comment>
    Lowering query for ContactInformation attributes.
  </comment>
  <outputs>
    <output> gender </output>
    <output> countryName </output>
  </outputs>
  <sparqlquery><![CDATA[
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX st4rt: <http://www.semanticweb.org/Mehdi/ontologies/2017/2/st4rt#>
PREFIX it2r: <http://www.it2rail.eu/ontology#>
PREFIX customer: <http://www.it2rail.eu/ontology/customer#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
SELECT ?gender ?countryName
  WHERE {
  ?p rdf:type customer:Passenger .
  ?p <http://xmlns.com/foaf/0.1/gender> ?gender
  ?p st4rt:isInCountry_Alpha2 ?alpha2 .
  ?country countries:countryCodeISO3166Alpha2 ?alpha2 .
  ?country countries:countryNameISO3166Short ?countryName }
]]></sparqlquery>
</query>
```

Fig. 4 Lowering query for multiple attributes

The *LoweringContactInformation* SPARQL query (which is of type *Lowering*) returns two values; the *outputs* annotation, an ordered set of attribute names, tells the converter to invoke the setter methods of attributes *gender* and *country*, using the results of the SPARQL query, respectively. Dually, when a *@Sparql* annotation has the *Lifting* type the converter retrieves the values of the attributes appearing in the *inputs* set by invoking their getter methods and prepares the query to be executed.

4. Architecture of converters

The design of the Interoperability Framework is based on the provision of a set of semantic interoperability services that can be deployed in multiple configurations, and that do not mandate a specific set of communication protocols or frameworks, leaving the choice of deployment strategies to partners that may opt to re-use a shared enterprise service bus, perhaps on a virtual private network protected by specific security and authentication protocols, or decide to engage in pure peer-to-peer exchanges over the public world wide web, or a mixture of these or other options.

A ST4RT converter can play a central role in the Interoperability Framework, since it can be embedded as a library to implement message conversion inside an enterprise service bus or in a dedicated Web Service. Figure 5 gives an overview of the main layers of a ST4RT converter, which are briefly described in the following.

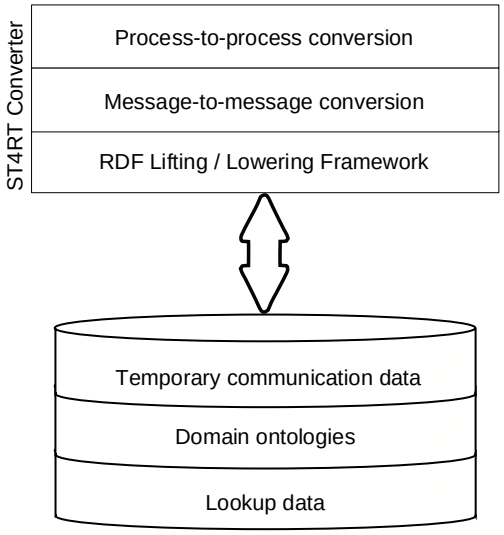

Fig. 5 Overview of the layers of a ST4RT converter

The RDF Lifting and Lowering framework provides Java Persistence API Architecture (JPA)-like capabilities extending it with the ability to operate on triple stores – i.e., on database systems designed for storage and retrieval of RDF statements ("triples") through semantic queries. Much as JPA annotations on Java classes provide automatic object-relational mappings from these classes to relational database schemas and vice-versa, the RDF framework provides automatic mappings to/from RDF statements. This enables the developer to delegate the mechanics of connection, input/output and storage/retrieval of the data across the network to the framework, while concentrating on the manipulation in pure Java of objects and relationships which, unlike the JPA entities, represent logical statements connected to the domain's axiomatic description of the domain's ontology. For example, triples generated by a SPARQL query as a result of logical inference – i.e., new logical statements inferred based on the axioms and the existing data – are automatically instantiated as Java objects and relationships and are therefore available for "ordinary" programming.

The message-to-message conversion layer is built on top of the lifting and lowering capabilities and is able to manage a simple request/response business process. Such feature is implemented by introducing a "communication context" and by letting developers store intermediate triples inside a temporary RDF repository. The repository also hosts domain ontologies and lookup data (RDF datasets representing nearly static assets such as Stop Places and Locations). It becomes therefore possible to create the response message not just according to data provided by the destination service, but also according to the corresponding request. This is mandatory for converting messages whenever a "session" is defined.

Being able to convert individual messages from one standard to another one is the starting point for process-to-process conversion, which is currently being investigated, harvesting on the results of previous research projects about Semantic Web Services (SA-WSDL (Farrell, 2007)), Semantic Orchestration (OWL-S (Martin, 2004), WSMX (Haller, 2005)) and Semantic Business Processes (envisioned in (Hepp, 2005) and then researched in IP-Super EU project). A single request from the originating standard could require being translated into a much complex process in the destination standard. To enable such transformations, a more thorough analysis of the standards will be required. Being able to convert a message using semantics means being able to understand the meaning of each data structure. Being able to convert processes is far more complex, since it requires deep understanding of all the data exchanges and sequences that compose the process, and to identify whether the information that is required to forge messages is always available or whether it has to be obtained by calling external services or other legacy systems.

Figure 6 shows a UML Sequence Diagram (SD for short, (Fowler, 2013)) depicting the general steps performed by a ST4RT converter during the transformation process.

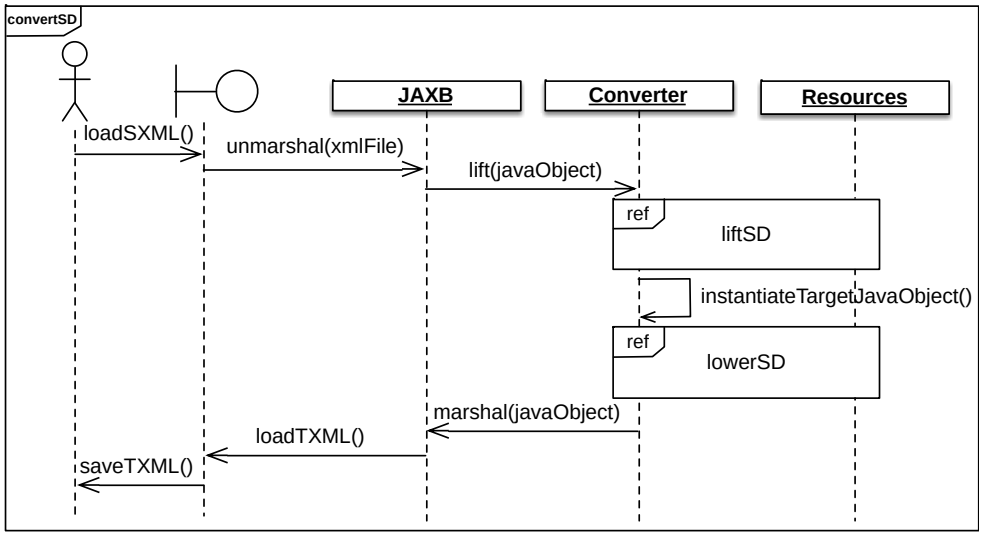

Fig. 6 General conversion mechanism

As the figure above shows, first the XML file conforming to the source data representation (e.g., FSM) is loaded, and it is used by the JAXB framework to instantiate a corresponding Java object. Then, the ST4RT converter uses the annotations defined for the source standard, and additional resources such as repositories of RDF triples on which to run SPARQL queries, to lift the Java object to the integrated ontology. After the lifting process has been completed, a new Java object, capturing the concepts of the target standard (e.g., TAP-TSI 918 XML) is instantiated, and it is filled out with the relevant information during the lowering phase (again, by exploiting resources such as SPARQL endpoints). Finally, the Java object is used by the JAXB framework to create an XML file that conforms to the target standard.

Figure 7 depicts the details of Sequence Diagram *liftSD* (Sequence Diagram *lowerSD* is similar, and is not shown here for the sake of brevity).

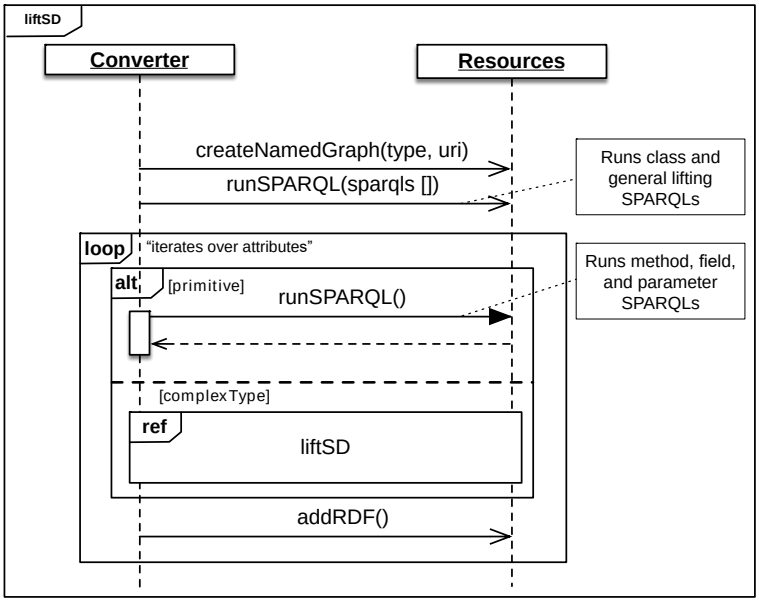

Fig. 7 Detail of the *liftSD* Sequence Diagram

As Figure 7 shows, during the lifting process the converter analyses the *@Sparql* annotations of the source standard, and runs them on a suitable resource (i.e., an RDF triple store). If the element to be lifted is a complex

one, this requires itself a lifting through the same mechanisms. After all necessary elements have been lifted, the generated RDF triples are added to the knowledge base.

5. Overview of the ST4RT demonstrator

The ST4RT project is developing a demonstrator that realizes the mechanisms defined in Section 3 and that integrates the prototype architected in Section 4 in a commercial tool. More precisely, the demonstrator will be based on the HEROS platform[§§] developed by Hit Rail[***], one of the partners in the ST4RT project. HEROS (HERmes Open Services) is a cloud service, based on the Niklas[†††] middleware platform, offering interoperability support to the European railway industry.

One of the applications implemented by Hit Rail on HEROS is a message translator, converting the messages used for seat reservation requests from the bit-oriented format defined in the TAP TSI Regulation to its corresponding XML format, and back. This translation can be considered a relatively simple mapping, since the two TAP TSI formats (bit-oriented and XML) have different syntaxes, but the same semantics – notice, however, that in practice the translation is not straightforward, since it must take into account the context of the message, it builds complex 918 structures, and it maps asynchronous messages to/from synchronous ones. The ST4RT project will add a complexity level, focusing on a case study concerning the transformation of messages written in standards (TAP-TSI 918 XML and FSM) with different semantics.

Figure 8 shows a general architecture of the demonstrator that will realize the mechanisms outlined in Figure 6 and Figure 7, with reference to the mapping of an FSM request to a TAP-TSI 918 XML message.

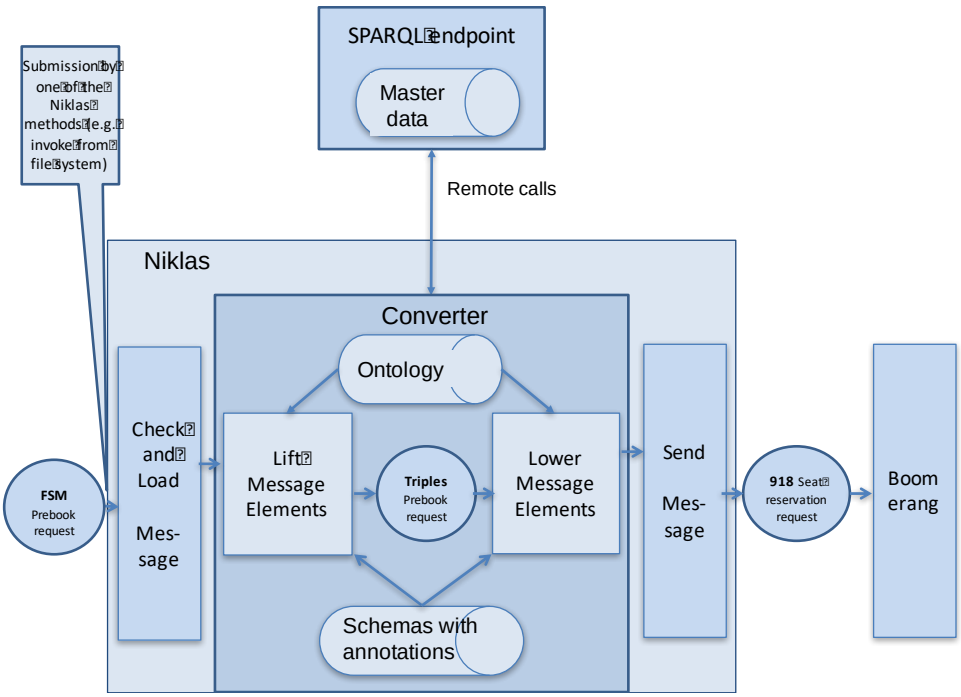

Fig. 8 General architecture of the demonstrator including a ST4RT converter

As shown in Figure 8, after loading the incoming (FSM) message, the converter uses the annotated schemas, the integrated ontology, and a SPARQL endpoint on which to run queries such as those presented in Section 3 to create new "lifted" RDF triples; in turn, these are used, along with the aforementioned elements, to create the

[§§] https://www.hitrail.com/heros

[***] http://www.hitrail.com

[†††] https://www.copernicus.nl/niklas-integration-platform/

"lowered" message to be sent to the 918-speaking service. A similar architecture, not shown here, is used to translate back the response of the 918-speaking service to the FSM-speaking requestor.
The process will make use of Boomerang, another tool developed on HEROS by Hit Rail. Boomerang is a service that can receive any type of seat reservation message, either in bit-oriented or in XML version, check its syntactical correctness and send back a coherent answer in the same syntax.

6. Conclusion

This paper presented the approach used in the ST4RT project to facilitate the interoperability of transport services that are based on different standards for the representation of data. The approach is based on the idea of annotating the legacy data representations in order to map the concepts included therein to semantically similar concepts in a reference ontology, and to use SPARQL queries (and possibly inferences) to fill any gaps or missing information. The approach is currently being implemented in a prototype component, and it will be part of a demonstrator built on existing technology provided by one of the partners of the project. The developed demonstrator will contribute to the realization of the vision put forth by IP4 of Shift2Rail, which centres around the idea of the "Interoperability Framework" to allow for the creation of a distributed semantic "web of transport" and for the elimination of the barriers that currently hamper the realization of a Single European Transport Area.

In addition to the realization of the prototype implementation and of the demonstrator, the ST4RT project will also extend the approach outlined in this paper to take into account for the possibility of complex mappings not only from the point of view of the data, but also of the workflows realized – e.g., to allow for the transformation of single messages into a set of request/response interactions rather than into another single message. The project is also investigating the possibility to enrich the annotations with declarations in the Shapes Constraint Language (SHACL[‡‡‡]).

7. References

Arnold, K. a. (2005). The Java programming language. Addison Wesley Professional.

Bray, T. a.-M. (1998). Extensible Markup Language (XML), W3C (World Wide Web Consortium).

Della Valle, E., Cerizza, D., Celino, I., Estublier, J., Vega, G., Kerrigan, M., M. Kerrigan, Ramírez J., Villazon B., Guarrera P. G. Zhao, G. Monteleone, G. a. (2007) SEEMP: an semantic interoperability infrastructure for e-government services in the employment sector. The Semantic Web: Research and Applications, 220-234, 2007.

Farrell J, Lausen H. (2007). Semantic Annotations for WSDL and XML Schema. W3C Recommendation, 2007.

Fowler, M. a. (2003). UML Distilled, Addison Wesley.

Gao, S. a.-M. (2009). W3C XML schema definition language (XSD) 1.1 part 1: Structures (Vol. 30). W3C Candidate Recommendation.

Haller A. a. (2005) Wsmx-a semantic service-oriented architecture. Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on. IEEE, 2005.

Harris, S. a. (2013). SPARQL 1.1 query language. W3C recommendation, 21(10).

Hepp M. a. (2005) Semantic business process management: A vision towards using semantic web services for business process management. e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on. IEEE, 2005.

Lassila, O. a. (1999). Resource description framework (RDF) model and syntax specification. W3C Recommendation, 1999.

Martin A. a. (2004). OWL-S: Semantic Markup for Web Services. W3C Member Submission, 2004.

ST4RT Project. a. (2017) D4.1 "Analysis of state-of-the-art ontology conversion tools". Available at: http://st4rt.eu/. 2017.

Vetere, G., Lenzerini, M. a. (2005) Models for semantic interoperability in service-oriented architectures. IBM Systems Journal, 44(4), 887-903, 2005.

Acknowledgements

The authors would like to thank all the partners that are involved in the project. These are CEFRIEL, D'APPOLONIA, Hit Rail, Oltis Group, Politecnico Di Milano, TRENITALIA, UIC and UNIFE.

‡‡‡ https://www.w3.org/TR/shacl/