Contract No. H2020 – 730842

SEMANTIC TRANSFORMATIONS FOR RAIL TRANSPORTATION

D5.2 KPI and metrics for evaluation

Due date of deliverable: 31/01/2018

Actual submission date: 06/08/2018

Leader/Responsible of this Deliverable: Hit Rail

Reviewed: Yes

Document status		
Revision	Date	Description
1	15/12/2017	First draft of the document
2	07/02/2018	Final version after TMC review and approval
3	06/08/2018	Adjustments following comments from PO

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: 01/11/2016 Duration: 24 months

EXECUTIVE SUMMARY

This deliverable provides a detailed description of the testing process that will be put in place to hopefully demonstrate the feasibility of the conversion of messages through a common ontology.

It also defines the Key Performance Indicators that the project intends to adopt as measurement of the quality of the achieved results, and the numerical values expected to be measured for each of them.

ABBREVIATIONS AND ACRONYMS

Abbreviation	Description
B2B	Business to Business
CPU	Central Processing Unit
EAI	Enterprise Application Integration
FSM	Full Service Model
FTP	File Transfer Protocol
GB	Gigabyte
GUI	Graphical User Interface
H2020	Horizon 2020 framework programme
HEROS	Hermes Open Services
JAXB	Java Architecture for XML Binding
KPI	Key Performance Indicator
MQ	Message Queuing
RAM	Random Access Memory
RDF	Resource Description Framework
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
ST4RT	Semantic Transformations for Rail Transportation
TAP	Telematics Applications for Passenger service
TSI	Technical Specification for Interoperability
UML	Unified Modeling Language
WP	Work Package
WS	Web Service
XML	eXtensible Mark-up Language
XSD	XML Schema Definition

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Once the Work Packages 3 and 4 have defined the specifications for the semantic conversion of our use case messages, it is the goal of Work Package 5 to show, through a practical demonstrator, whether and how well the semantic approach can solve the problem arising from the existence of different communication standards.

Therefore, the experiment with the demonstrator must prove not only that the conversion is possible, but also that the process can run with acceptable and stable performances.

The final scope of this deliverable is to define which indicators will be adopted as a measure of the demonstrator performances, and how to observe and process them.

In order to allow the reader to understand what those indicators represent, we start by describing the hardware/software environment where the demo will take place (a description was already sketched in deliverable D5.1, but it is better detailed here), and by explaining how the test will be run: the actors, the roles, the steps.

2. HARDWARE/SOFTWARE ENVIRONMENT

2.1 PLATFORM

The demo will run on one of the servers that Hit Rail uses to provide its services [1]. Those servers are located in Brussels, in the computer center of Infrabel, and used by Hit Rail in hosting mode.

The specific server used for the demo will be the Linux virtual server currently used by Hit Rail for Test and Acceptance.

It has Operating system Red Hat EL 6, 2 CPU's, 8GB RAM, 80 GB Disk.

The server will continue to be used in parallel for Hit Rail's test and acceptance needs. The average usage of the server for the needs of ST4RT during the demo phase is estimated to be around 40-50%.

2.2 MIDDLEWARE

On the virtual server runs the middleware NIKLAS, supplied by Copernicus www.copernicus.nl/niklas-integration-platform/

NIKLAS Integration Broker is a Business to Business (B2B) and Enterprise Application Integration (EAI) software that enables an organization to exchange electronic information with other organizations and its internal applications without manual intervention.

The core functionality of NIKLAS is receiving, classifying, transforming and sending of messages between various trading partners. During the transformation process NIKLAS can apply business logic for transformation and for enrichment of data.

A NIKLAS instance is a collection of NIKLAS engines. These engines differ in function and design and each fulfill a distinct purpose. Each NIKLAS instance can run only one of each type of engine.

NIKLAS can manage different data exchange protocols, such as WS, MQ and FTP. For the ST4RT demo it has been agreed to use Web Services.

Web services were already used on Hit Rail's test instance (TST) of NIKLAS, but they implemented Hit Rail's proprietary HEROS SOAP architecture. For the scope of ST4RT Hit Rail has provided, through Copernicus, a completely new web service using the FSM [2] SOAP architecture (limited to the booking part of the FSM spec and the SOAP layer. The full FSM spec is much larger but it is not required for the purpose of our demo and its deployment would have involved too much time and cost).

2.3 ST4RT CONVERTER

The translator software will be developed by OLTIS Group, according to the specifications defined in WP3 and WP4. It will be loaded on NIKLAS by OLTIS Group, if necessary with support of Copernicus. This process will be repeated every time a new version of the software is developed, to correct errors or to improve the performances.

The functions to be performed by the translator software are described in detail in deliverable D3.3. For the benefit of the reader we repeat here the main concepts.

In the ST4RT approach, it is assumed that the legacy data representations to be mapped through the reference ontology – and the corresponding data instances – are given in terms of standard technologies, such as XSD (XML Schema Definition, [3][3]]) and XML (eXtensible Markup Language, [4]). **Error! Reference source not found.** shows the lifting/lowering process in the case where for b oth the source and target standards the data representation is defined through XSD files, as is the case for both FSM and TAP TSI [5] 918 [6] XML messages.

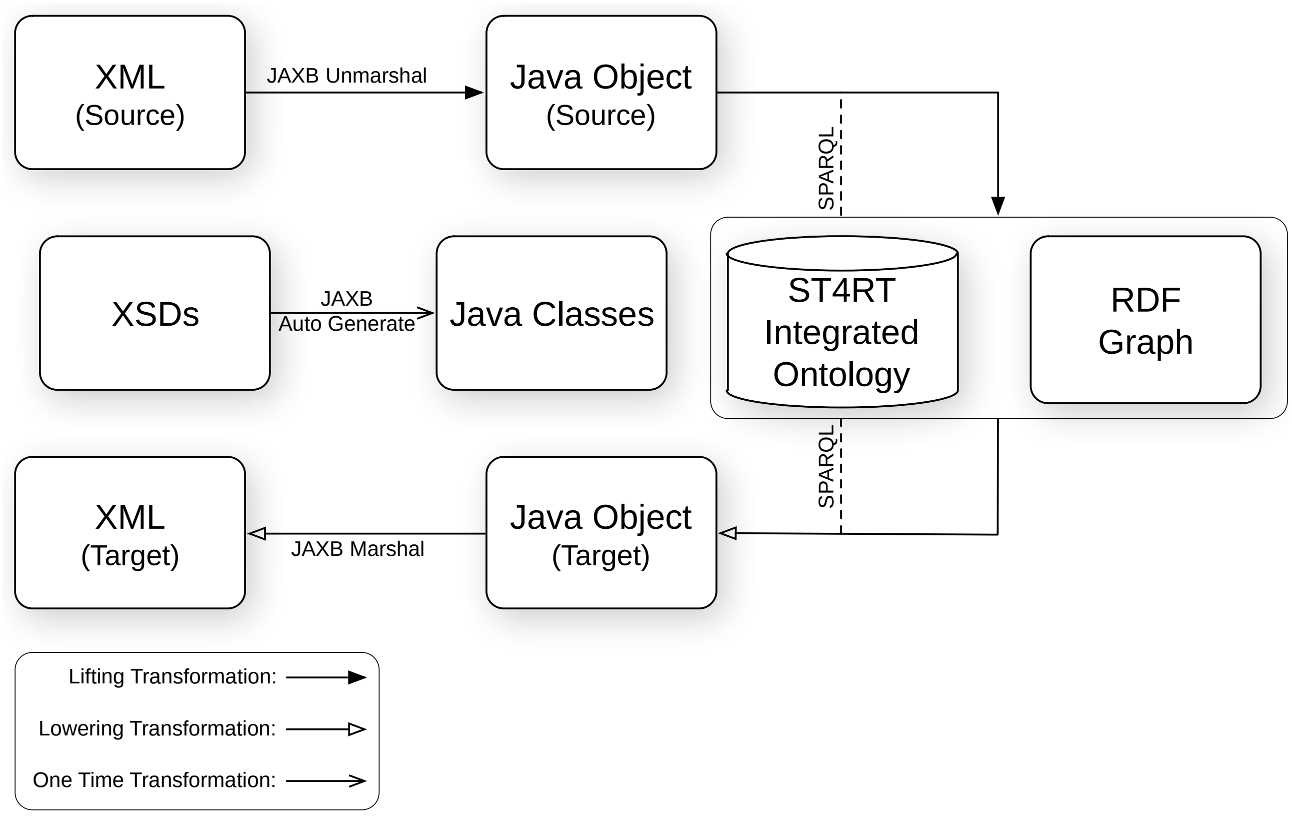

Figure 1: Approach workflow in the case where source and target standards are defined through XSD files

Initially, data schema definitions – for both source and target standards – given in terms of XSD files are translated into declarations in the Java programming language using existing frameworks such as JAXB [7], and are annotated to match them with concepts in the reference ontology. The annotated Java code is used to produce, during the lifting phase at runtime, sets of RDF (Resource Description Framework [8]) 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.

Error! Reference source not found. shows a UML sequence diagram [9] depicting the general s teps performed by a ST4RT converter during the transformation process.

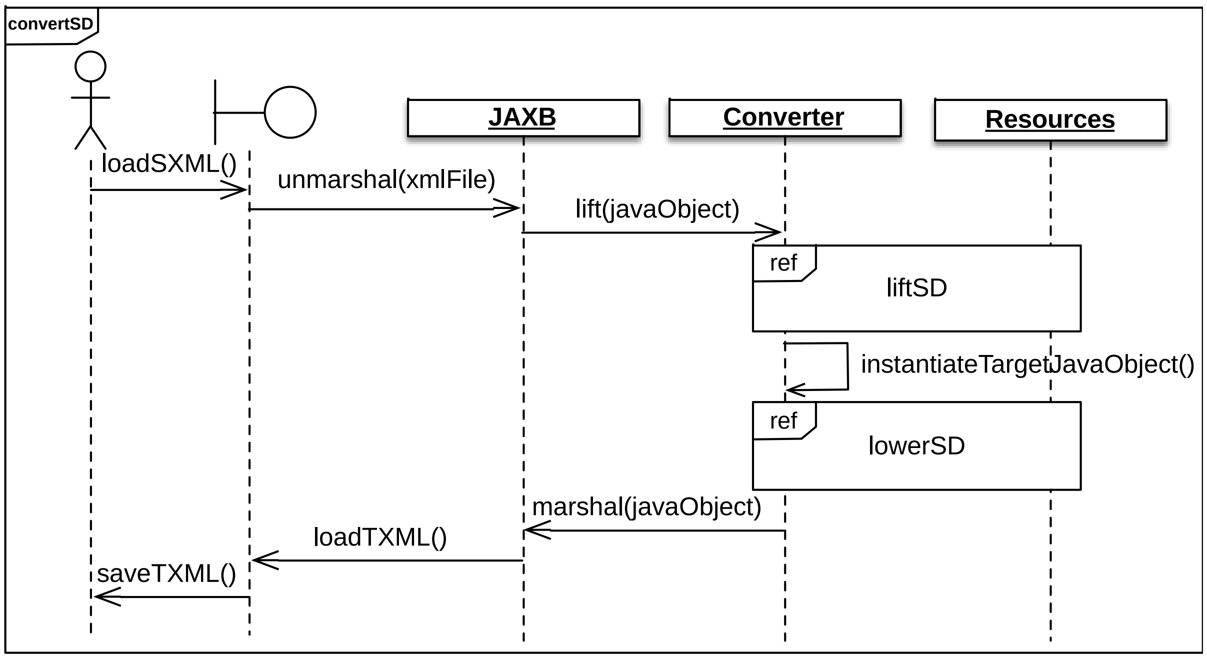

Figure 2: General conversion mechanism

As the figure 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.

2.4 Support Tool

The first half of the demo will consist of the translation of an FSM PreBookingRequest into a 918 ReservationRequest, and the second half will consist of the reverse translation of the 918 ReservationReply into an FSM PreBookingResponse.

To close the loop between the first and second half at runtime it is necessary to use a tool capable of reading the request coming out of the first step, and producing the correct answer to feed the second step.

As already stated in deliverable D5.1, to this end we will adapt the existing tool Boomerang, also developed by OLTIS Group based on specifications from Hit Rail.

It will receive via HEROS Web Services the 918 ReservationRequest message produced by the converter and sent by NIKLAS, will validate it against the 918 XSD, will produce the correct answer and will return it to NIKLAS as response to the received Web Service request, to be converted into the FSM PreBookingResponse message.

2.5 Demo Architecture

The newly created ST4RT converter will be inserted inside NIKLAS as a jar library, similarly to what happens with the current translator in the existing solution.

The proposed architecture is depicted in the figure 3 below.

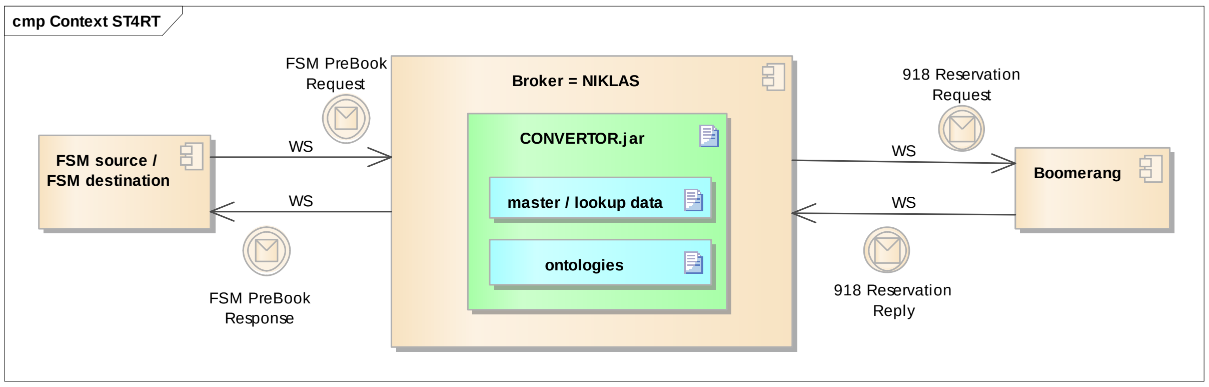

Figure 3: Demo architecture

3. EXECUTION OF THE TEST

This section describes how the project team intends to perform the test of the ST4RT demonstrator. Obviously, deviations from the foreseen process could be decided during the test phase, in order to improve the efficiency of the tests or to overcome inconveniences that could arise during the tests.

The deliverable D5.4 will anyway provide a detailed description of how the tests were actually conducted and the results collected.

3.1 ACTORS AND ROLES

The interaction with the converter will take place via a web service client, that will allow the operator to submit to the system the initial FSM PreBookingRequest and to receive the FSM PreBookingResponse produced at the end of the process. The operator will have access to the statistics, the logs and the performance measurements.

Since the access to the Hit Rail’s servers is only possible, for security reasons, from a terminal connected to the Hermes network, this operation will be performed by persons of OLTIS Group from their offices in Pardubice (Czech Republic).

The other partners of the project will receive regular reports about the results of the tests, and will have the possibility at agreed times of assisting directly to some test session on site or via Skype/Webex.

3.2 DEMO STEPS

The demo will proceed as follows:

1. the FSM PreBook Request is sent to the Broker via a web service
2. the Broker calls the ST4RT converter’s method, forwards the message string and waits for the return value
3. The converter converts the incoming string into the outgoing string (918 ReservationRequest)
4. The converter returns the converted message to the Broker
5. The Broker sends the 918 ReservationRequest to Boomerang
6. Boomerang creates the ReservationReply and sends it back to the Broker
7. Steps 2, 3 and 4 are repeated to translate the ReservationReply into an FSM PreBookingResponse

3.3 Operations

A first phase will be dedicated to the debugging, to eliminate the unavoidable errors always present in a new software, and to the tuning of all elements impacting on the performances (algorithms, annotations, ...).

Once the converter is proven to behave as expected, tests will be repeated a large number of times to collect series of data from which it will be possible to calculate average and variance of the indicators.

It is foreseen to perform the conversion tests with all data in central memory, without access to databases on external storage. Even in case of a possible future commercial use with complete sets of reference data (locations, timetables, etc.) the existing technology allows this solution, with adequately sized servers.

3.4 Results evaluation process

As the server where the tests will run is not dedicated only to ST4RT, for each repetition of the test the corresponding CPU load will be recorded, in order to be able to correlate the average KPIs to the occupancy of the server.

It is to be noted that the KPIs defined below are high level ones that can be identified within the initial phase. During the implementation of the converter software it is foreseen to introduce additional logging points to detect the duration of micro-activities internal to the converter.

The results of these additional Performance Indicators will be described in the final report of WP5.

4. KPI & METRICS

The most relevant indicator showing the performance of a system like our converter, whose goal is to be used in future as a real-time intermediary between a terminal where a customer requests a reservation and a system allocates the place, is the response time.

Therefore, we will assume as the main indicator the total time elapsed between the acceptance of the PreBookRequest by NIKLAS and the sending out of the PreBookResponse must be acceptable.

It must be understood that every time we introduce logging steps (to measure performance time and capturing information) we slow down the process. A real Production system would have less logging capabilities and therefore higher performance.

The acceptable total roundtrip time for a production time would be five seconds or lower. For the experimental system developed for this project we have set an acceptable roundtrip time of seven seconds.

We will also take into account a set of partial times, corresponding to single technical steps (XSD validation by NIKLAS, transfer of payload to the converter, unmarshalling, lifting, etc.). Those measurements will allow us to understand where to introduce improvements (tuning) should the total response time seem to be excessive.

NIKLAS automatically timestamps all its own operations, while OLTIS Group will insert in the translator software the appropriate logging to record the duration of the different sub-processes.

Table 1 below shows the Key Performance Indicators and metrics identified by the project.

Time elapsed between:	Milliseconds
Acceptance of the FSM PreBookRequest by NIKLAS and sending out of the FSM PreBookResponse (Overall response time)	Under 7 seconds
Translation of request (time taken by translator software only), from delivery by Niklas to reception of translated message by Niklas	Under 2 seconds
Niklas request routing/processing overhead time (without translation)	Under 1 second
Boomerang total response message creation time	Under 1 second
Translation of response (time taken by translator software only), from delivery by Niklas to reception of translated message by Niklas	Under 2 seconds
Niklas response routing/processing overhead time (without translation)	Under 1 seconds

Table 1: Project KPIs and metrics

5. CONCLUSIONS AND FUTURE WORK

Starting on month 22 of the project, the Task 5.4 will:

- Generate and collect the results of the testing activity;
- Compare the results vs above defined KPI and metrics;
- Produce a draft report on the test results;
- Collect feedback on the draft and issue the final report, in the form of deliverable D5.4.

REFERENCES

[1] Hit Rail, *H01 Passenger reservations Translator* https://www.hitrail.com/h01-passenger-reservations-translator

[2] TSGA, *Full Service Model – Who we are and what we do*, https://tsga.eu/fsm

[3] XSD: Gao, S., Sperberg-McQueen, C. M., Thompson, H. S., W3C XML schema definition language (XSD) 1.1 part 1: Structures (Vol. 30). W3C Recommendation, 2012. https://www.w3.org/TR/xmlschema11-1/

[4] XML: Bray, T., Paoli, J., Sperberg-McQueen, C. M., Extensible Markup Language (XML), W3C Recommendation, 2008. https://www.w3.org/TR/xml/

[5] ERA, *Telematics Applications for Passenger Services Technical Specifications for Interoperability* (TAP TSI), http://www.era.europa.eu/Document-Register/Pages/TAP-TSI.aspx

[6] UIC, *leaflet 918-1 Electronic reservation of seats/berths and electronic production of travel documents - Exchange of messages* http://www.shop-etf.com/en/electronic-reservation-of-seats-berths-and-electronic-production-of-travel-documents-exchange-of-messages-8906

[7] JAXB, *Java Architecture for XML Binding* https://jaxb.java.net

[8] RDF: Schreiber, G., Raymond, Y., *RDF 1.1 Primer*. W3C Recommendation, 2014. https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

[9] UML: Fowler, M. *UML Distilled*, Addison Wesley, 2003 https://www.amazon.it/Uml-Distilled-Standard-Modeling-Language/dp/0321193687