Composable Semantic Data Transformation Pipelines with Chimera

Marco Grassi, Mario Scrocca, Alessio Carenini, Marco Comerio and Irene Celino

Cefriel – Politecnico di Milano, Milan, Italy

Abstract

In a multi-stakeholder ecosystem, data exchange is not sufficient and semantic interoperability should be achieved to ensure different information systems can communicate without loss of meaning. A semantic conversion procedure relying on a reference ontology can enable an efficient conversion between data formats sharing common semantics, whilst generating an integrated and interoperable knowledge graph. The Chimera framework proposes a flexible and configurable solution to address different requirements for the integration of semantic converters across heterogeneous systems. Chimera minimises the effort required to specify custom semantic data transformation pipelines, offering different ready-to-use components to integrate heterogeneous data sources, manipulate knowledge graphs, and execute declarative mapping rules for their construction and exploitation. We present the Chimera framework as a resource for the Semantic Web community and we demonstrate its usage considering a challenging use case in the transportation domain.

Keywords

Semantic conversion, Declarative mappings, Data interoperability

1. Introduction

The issue of data interoperability is a significant concern when operating within a multi-stakeholder ecosystem where diverse actors employ heterogeneous data formats, specifications, and semantics. The ability to exchange data without any loss of meaning among communicating parties is an essential objective, but it is notoriously challenging to achieve also due to different requirements for the integration of heterogeneous information systems. Chimera[1], is an open-source framework for the definition of composable semantic data transformation pipelines. The proposed solution addresses data interoperability issues by leveraging Semantic Web technologies to extract and harmonise the intended semantics of heterogeneous data sources by means of transformation procedures. The framework is designed as an extendable set of building blocks to simplify the configuration of semantic data transformations and to provide flexibility in addressing diverse scenarios and requirements. Chimera, originally designed to support the conversion use case presented in [1], has been completely refactored to address

KGC '23: Knowledge Graph Construction Workshop co-located with the ESWC 2023, May 28, 2023, Hersonissos, Greece

marco.grassi@cefriel.com (M. Grassi); mario.scrocca@cefriel.com (M. Scrocca); alessio.carenini@cefriel.com (A. Carenini); marco.comerio@cefriel.com (M. Comerio); irene.celino@cefriel.com (I. Celino)
000-0003-3139-3049 (M. Grassi); 0000-0002-8235-7331 (M. Scrocca); 0000-0003-1948-807X (A. Carenini); 0000-0003-3494-9516 (M. Comerio); 0000-0001-9962-7193 (I. Celino)

[1]https://github.com/cefriel/chimera

more generic requirements and to facilitate its extensibility and reusability. In this paper, we discuss the main concepts behind Chimera, how it can be reused for different use cases, and we demonstrate its application to support a challenging scenario in the transportation domain. The remainder of the paper is organised as follows: Section 2 provides an analysis of the challenges associated with the semantic data transformation process, Section 3 presents the Chimera framework and describes the resources available, Section 4 exemplifies the usage of Chimera, Section 5 compares Chimera with existing alternative solutions, Section 6 presents conclusions and future work.

2. Problem description

The challenge addressed by Chimera is twofold and concerns how to support data interoperability between different actors relying on different data formats (Challenge 1) and heterogeneous information systems with different functional requirements for data exchange (Challenge 2). In this section, we explain the two challenges and we exemplify them in a concrete and complex usage scenario.

2.1. Challenge 1: different data formats

The commonly arising challenge is about how to deal with heterogeneous data formats with varying semantic interpretations employed by multiple actors within the same domain. This phenomenon may arise due to several factors that make the establishment of standards difficult, for example, the persistence of legacy applications or the usage of proprietary data formats.

To address this first challenge, Chimera adopts a semantic any-to-one centralized mapping approach [2] relying on a global conceptual model, in which each stakeholder only needs to define mappings to/from the global conceptual model. The reasons for adopting such a solution and its advantages are explained in detail in other works, including [1, 3].

2.2. Challenge 2: heterogeneous systems

The problem of interconnecting heterogeneous information systems (which also employ different data formats) is usually addressed by defining custom solutions (e.g., ad-hoc software components), which are hard to maintain and can show scalability issues. The second challenge deals with the fact that there is no single interoperability problem and, therefore, it is not possible to define a single interoperability solution [4].

A flexible and configurable set of specialized tools is needed to cope with different functional requirements to implement data interoperability across heterogeneous information systems. While a semantic conversion process can offer a valid solution to define transformations across data formats, the integration of such processes considering different data sources and sinks is something that requires a case-by-case analysis (e.g., protocols adopted and interfaces exposed). Additional requirements, that are not strictly related to the conversion process may be defined, such as the validation of the produced data or their processing by external systems.

To address this second challenge, Chimera is built on top of a well-known open-source integration framework, which was already conceived to easily configure the integration of various systems consuming or producing data according to heterogeneous requirements.

2.3. Example scenario: multimodal traffic management

As an example, we consider in this paper a use case emerging from the European H2020 TANGENT project[2] in which Chimera is used to define a solution for data harmonisation and fusion in the context of multimodal traffic management.

In this scenario, the objective is to implement a real-time integrated dashboard for city and transport authorities that could enable intelligent incident detection and thus facilitate prompt intervention. To this end, real-time data from multiple data sources such as semaphores, parking sensors and traffic sensors need to be collected and integrated. Additionally, data related to planned events, such as concerts or sporting events need to be taken into account because they have a large and predictable effect on traffic conditions.

The definition of a semantic conversion process should consider the following requirements: data from sensors and devices are collected through a Kafka[3] deployment but are represented using heterogeneous data formats; additional data on planned events are stored as datasets in a data catalogue and should be accessed through its REST API; harmonised data should be stored for additional processing by other systems (e.g., for the training of machine learning models on historical data); a monitoring platform is used to track relevant events about the process; the dashboard receives data through a WebSocket expecting a specific JSON format.

The presented scenario for multimodal traffic management highlights the need for data interoperability considering different data formats (Challenge 1) and a good number of custom requirements that can be elicited for enabling data interoperability across heterogeneous systems (Challenge 2). After presenting in detail the Chimera framework, we will explain how Chimera can successfully address and solve the requirements from this scenario.

3. Chimera

The Chimera framework aims at addressing the two mentioned challenges by (i) providing an efficient set of software components to implement an any-to-one centralized mapping approach through Semantic Web technologies, and (ii) integrating these components in a broader ecosystem facilitating the implementation and deployment of semantic data transformation pipelines among heterogeneous systems. The design of the Chimera framework follows a modular and low-code approach to minimise the effort required to specify and configure a pipeline for different scenarios.

3.1. Building Blocks for a Semantic Data Transformation Pipeline

The decision of the modular approach is based on the assumption that a semantic data transformation pipeline could be broken down into a set of smaller, composable and reusable building

[2]https://tangent-h2020.eu/
[3]https://kafka.apache.org/

blocks. Moreover, the aim is to facilitate the extensibility of the framework to integrate additional blocks. We defined four types of building blocks for a semantic data transformation pipeline (shown in Figure 1):

1. **Graph Construction**. Data from heterogeneous sources is converted to RDF according to a reference ontology. The conversion process from a specific data format to RDF, also known as lifting, can be handled by employing different approaches for knowledge graph construction [3].
2. **Graph Transformation**. An RDF knowledge graph can be augmented by means of graph operations such as: adding RDF triples to the graph, generating a new graph through SPARQL Construct queries, and applying inference considering the reference ontology.
3. **Graph Validation**. The validation of the graph (e.g., by using SHACL [5] shapes) can be useful and necessary to validate the correctness of the implemented procedure and/or of the ingested data.
4. **Graph Exploitation**. The information in an RDF knowledge graph is extracted and converted to a specific target data format through a lowering process.

Despite being presented in a specific order, each operation is independent from the others and can be used to support different pipelines. For example, the result of the pipeline can be an RDF graph constructed and then transformed (thus omitting a validation and an exploitation step), or a pipeline can be implemented only to validate the RDF graph exchanged between two systems.

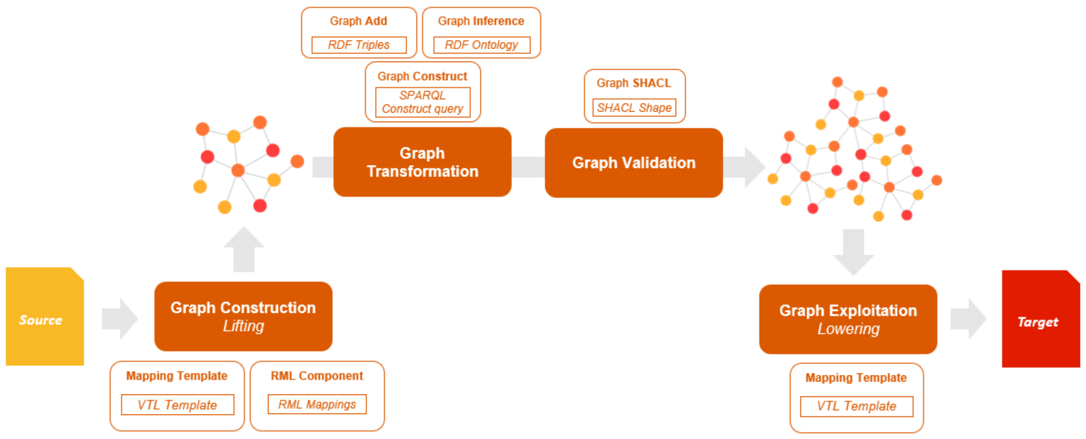


Figure 1: Overview of the building blocks of a semantic data transformation pipeline and available Chimera components.

We implemented Chimera on top of Apache Camel[4], a Java integration framework to facilitate the integration with various systems consuming or producing data. We chose Camel to inherit its features and advantages: not only it is a completely open-source, configurable and extensible

[4]https://camel.apache.org/

solution, but it also implements best practices and patterns to solve the most common integration problems, including the Enterprise Integration Patterns (EIP) [6]. Finally, being a robust and stable project, Camel supports out-of-the-box several components, runtimes and formats to access and integrate a large set of existing systems and environments.

Apache Camel relies on the basic concept of *Route* defining a certain logic to load, extract, integrate, transform and output data. Each *Route* is a pipeline composed of a set of components that are applied in a specific sequence to a certain *Exchange*, i.e., an entity going through a *Route*. The *Exchange* is identified by an identifier and it is similar to an envelope, it contains the messages (e.g., the data being processed) but also a set of properties that can be used to carry an additional state during the *Route* execution.

The basic idea of Chimera is to define additional components[5] for Apache Camel supporting operations on an RDF graph that is passed along the Camel *Route* within an *Exchange*. All Chimera components use the RDF4J library[6] to process and handle RDF graphs. The *RDF Graph* in Chimera pipelines is an abstraction that can refer to a local knowledge graph (in-memory, filesystem), or a remote graph stored in a triplestore or accessible through a SPARQL endpoint. Chimera components can be used also in conjunction with already available and established components from the Apache Camel framework[7] and/or custom components defined by the user (e.g., for a custom pre-processing). Moreover, the definition of additional components for each graph operation is possible (e.g., the integration of an additional solution for knowledge graph construction) by simply leveraging the *RDF Graph* abstraction.

3.2. Chimera components

At the time of writing, Chimera provides the following set of components to enable the graph building blocks described above:

Graph Component: The component implements the Graph Transformation and Graph Validation building blocks and allows for the following operations:

- **Graph Get:** accesses an existing RDF Graph to be used in the pipeline.
- **Graph Add:** adds RDF triples from one or more resources to the RDF Graph.
- **Graph Construct:** applies a SPARQL Construct query to the RDF Graph giving the user the possibility of adding the generated triples to the previous RDF Graph or to a new RDF Graph.
- **Graph Inference:** enables inference on the RDF Graph considering a given ontology (currently RDFS [7] is supported).
- **Graph SHACL:** validates the RDF Graph using a (set of) SHACL shape(s).
- **Graph Detach:** clears the RDF Graph and/or closes the pending connections.
- **Graph Dump:** writes the RDF Graph to a file in a user-specified format, e.g. Turtle, N3 and others.

[5]https://camel.apache.org/manual/writing-components.html
[6]https://rdf4j.org/
[7]https://camel.apache.org/components/3.20.x/index.html

All the operations implemented by the graph component support the execution considering specific named graphs.

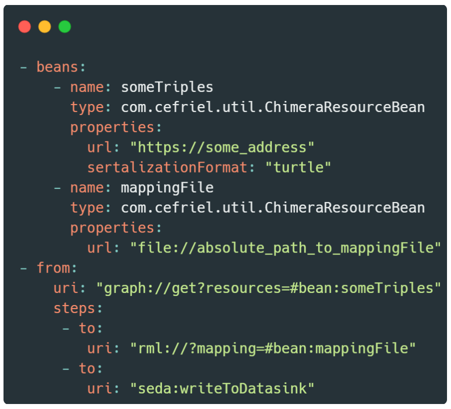
RML Component: The component implements the Graph Construction building block and consists of a lifting block enabling the execution of RML [8] mappings for knowledge graph construction from heterogeneous data sources. The component integrates the `rmlmapper-cefriel`[8], a fork of the `rmlmapper` library[9].

Mapping Template Component: The mapping template component implements both the Graph Construction and Graph Exploitation building blocks and supports both lifting and lowering operations; it is based on Apache Velocity[10] to implement a template-based solution to query the input data, process the result set and generate the output in the desired format. The component integrates the `rdf-template` library[11].

3.3. Using Chimera in practice

The Chimera framework is available on GitHub[12] and is released under the Apache 2.0 license. The repository contains the three components discussed that can be integrated into any Apache Camel project through the definition of a proper *Route*.

A Camel *Route* can be defined by simply providing a configuration file through one of the available DSL languages[13], without the need of writing additional code. Figure 2 shows an example Chimera pipeline configured using the YAML DSL of Apache Camel.



```yaml
- beans:
    - name: someTriples
      type: com.cefriel.util.ChimeraResourceBean
      properties:
        url: "https://some_address"
        sertalizationFormat: "turtle"
    - name: mappingFile
      type: com.cefriel.util.ChimeraResourceBean
      properties:
        url: "file://absolute_path_to_mappingFile"
- from:
    uri: "graph://get?resources=#bean:someTriples"
    steps:
     - to:
        uri: "rml://?mapping=#bean:mappingFile"
     - to:
        uri: "seda:writeToDatasink"
```

Figure 2: Example of YAML configuration for a Camel *Route* using Chimera components.

In the example route: (i) an RDF dataset is accessed via HTTP and integrated into an RDF Graph, (ii) a lifting operation is executed on the incoming message considering an RML declarative mapping file from the local filesystem and the result is added to the RDF Graph, (iii)

[8]Repository: https://github.com/cefriel/rmlmapper-cefriel
[9]https://github.com/RMLio/rmlmapper-java
[10]https://velocity.apache.org/
[11]Repository: https://github.com/cefriel/rdf-template, can be used as a library or as a standalone tool
[12]https://github.com/cefriel/chimera
[13]https://camel.apache.org/manual/dsl.html

the result is sent to a separate route (e.g., writing the output to a specific data sink). Thanks to the huge set of Camel's predefined components, Chimera can fulfil different integration requirements by leveraging multiple input and output channels. For example, input data from this route can be acquired by polling directories or FTP servers, or by exposing REST services, Web APIs and SOAP services, by using different publish-subscribe technologies such as JMS or AMQP.

3.4. Learning to use Chimera

A tutorial showing an example project to construct semantic data transformation pipelines is also available on GitHub[14]. The tutorial aims at introducing each Chimera component and showing how to configure them through the Apache Camel Spring XML DSL. For demonstrative purposes, a sample GTFS[15] feed is considered as input and the Linked-GTFS ontology[16] is used as the reference ontology. Multiple pipelines that make use of the Chimera components are presented and described in detail. Each pipeline is exposed as an HTTP endpoint, showing also the integration with the HTTP Apache Camel components. The tutorial also exemplifies different deployment options for Chimera pipelines, such as standalone JAR and/or container-based execution. Notably, in the proposed approach the defined Chimera pipelines can be modified through the XML file without requiring a new build process.

3.5. Improvements of Chimera w.r.t. previous work

In our previous In-Use paper [1], the initial release of Chimera is discussed on a practical conversion use case to demonstrate the feasibility and advantages of the proposed approach to solve challenge 1. The first implementation of Chimera mainly focused on the definition of the lifting/lowering approaches and their analysis for performance and scalability [9]. The current version of Chimera is the result of a complete refactoring effort over the initial framework and integrates the feedback collected from several EU research projects (e.g., SPRINT[17], RIDE2RAIL[18], TANGENT) and innovation projects with customers. Indeed, the application of Chimera to different use cases highlighted the importance of focusing on the overall framework to improve the reusability of the components and facilitate the configurability of complex pipelines for heterogeneous integration requirements. We re-designed and extended the building blocks of a semantic data transformation pipeline (e.g., adding Graph Validation), and we focused on addressing the issues discussed in challenge 2 to enable the application of Chimera to complex integration scenarios such as the one discussed in Section 2.3.

To facilitate the definition of semantic data transformation pipelines, we re-implemented Chimera as a set Apache Camel Components[19] that represent the basic abstraction for the extension of the Camel framework. As a result, it is easier to define pipelines reusing Chimera components in combination with existing Camel components and considering EIPs. Moreover,

[14]https://github.com/cefriel/chimera-tutorial
[15]https://developers.google.com/transit/gtfs
[16]https://github.com/OpenTransport/linked-gtfs
[17]http://sprint-transport.eu/
[18]https://ride2rail.eu/
[19]https://camel.apache.org/manual/component.html

we enhanced the decoupling of the different components by defining the expected input and output of each operation implemented. Both aspects also simplify the extensibility of the Chimera framework, i.e., the definition of additional components implementing one or more building blocks of a semantic data transformation pipeline. Finally, to facilitate the configuration of Chimera components we harmonised the set of accepted parameters across components and we identified the required set of parameters for each operation. As an example, the *Chimera Resource* abstraction (used in the Camel *Route* in Figure 2) has been introduced to generalise the specification of resources (local or remote) that Chimera components may need for their configuration.

The comparison of Camel *Routes* defined for the first[20] and the latest[21] version of the Chimera tutorial demonstrates the advantages of the implemented changes in the explicit definition of operations and parameters used for their configuration.

4. Addressing interoperability challenges with Chimera

To exemplify the potentiality of the Chimera approach for data interoperability across heterogeneous systems, Figure 3 describes the configuration of a pipeline fulfilling the requirements for the multimodal traffic management use case presented in Section 2. In the following we explain how Chimera is able to address both challenge 1 of harmonising data across different formats (i.e., heterogeneous sensor data formats, format of planned events, expected JSON output) and challenge 2 of integrating heterogeneous information systems (i.e., Kafka deployment, data catalogue, triplestore, monitoring platform, dashboard).

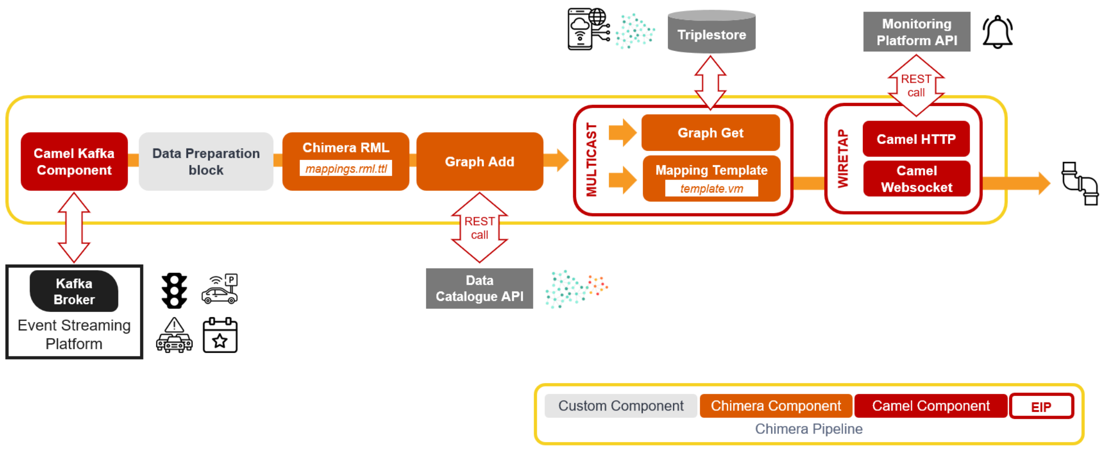


Figure 3: Chimera pipeline for the TANGENT use case presented.

The Apache Camel Kafka component[22] is able to ingest data from a Kafka broker and can

[20]https://github.com/cefriel/chimera-tutorial/blob/36d7eef3a91f03cffa6b1b571df5f56c64fcf274/src/main/resources/routes/camel-context.xml
[21]https://github.com/cefriel/chimera-tutorial/blob/main/src/main/resources/routes/chimera-route.xml
[22]https://camel.apache.org/components/3.20.x/kafka-component.html

be directly incorporated into a Chimera pipeline. The data coming from Kafka may require an additional data preparation step, in this case, the user can integrate into the pipeline a custom Apache Camel component defined for the specific use case. The benefit of defining such logic as a Camel Component is that once it has been defined, then it can be reused in other pipelines. For the harmonisation of incoming data using a common ontological data model, the Chimera RML component is integrated into the pipeline and configured to execute RML mappings for knowledge graph construction. The harmonised RDF data can then be augmented with RDF triples from external resources, such as the planned events available as an RDF dataset in a data catalogue, by means of the *Graph Add* operation. Using the *multicast* Enterprise Integration Pattern (EIP) provided by Apache Camel, the result of previous processing can be forwarded in parallel to multiple destinations. For example, it can be sent to a remote triplestore for storing but also continue the processing into the pipeline. Through the Mapping Template component, the knowledge graph is lowered to the target JSON format. The data in the obtained format can then be forwarded to the relevant components using the Camel-provided Wiretap EIP that allows to: (i) asynchronously process and send data to the monitoring platform, (ii) synchronously forward the data to their final destination using the Camel WebSocket component[23].

5. Related Works

The survey from Van Assche et al. [3] provides an overview of solutions for data transformation towards RDF graph generation. These approaches can be integrated within Chimera as implementations of the Graph Construction building block. Whilst only two approaches are currently implemented in Chimera (i.e, the RML Component and the Mapping Template component) the framework can be extended by packaging RDF knowledge graph construction tools as Camel components that are able to generate an *RDF Graph*. In the literature about RML processors [3, 10], the Chimera RML Component is sometimes referred to as Chimera but represents an implementation of one of the building blocks in the overall Chimera framework.

Considering the lowering, a limited set of standardised solutions exists in the literature. Potential approaches were proposed in [11], by querying RML lifting mappings to recreate a target CSV format, and in [12], by relying on SPARQL to define RDF transformation rules. In our work, we adopt a declarative approach to implement lowering, relying on SPARQL for querying data and on a template engine to efficiently serialise data according to the target format. Any approach that allows specifying a transformation from an RDF graph to a target format can be integrated as a Chimera component implementing the Graph Exploitation building block.

An evaluation of the performances of the RML Component in comparison with other RML processors is available in [10, 13]. A performance and scalability evaluation of the lifting and lowering approaches integrated into Chimera is discussed in [9].

The paper from Van Assche et al. [14] highlights the importance of defining sources and targets of the RDF graph generation process and proposes a solution based on the extension of RML. The Apache Camel DSL allows users to declaratively specify how to access data sources/sinks in a Chimera pipeline through Camel components. The RML-based definition of source and targets

[23]https://camel.apache.org/components/3.20.x/websocket-component.html

can be possibly automatically translated as a portion of a Chimera pipeline, thus avoiding the integration of additional libraries and functionalities within a Graph Construction component.

Finally, we cite similar works in the literature defining semantic-based ETL ("Extract, Transform and Load") pipelines. Talend4SW[24], similarly to the approach of Chimera with Apache Camel, leverages an existing tool (Talend) and offers additional components to interact with RDF data. UnifiedViews [15], LinkedPipes [16] and Barnard59[25] exploit Semantic Web principles to feed and curate RDF knowledge bases, thus focusing on the graph construction, transformation and validation. Each of mentioned approaches has its advantages for specific use cases, Chimera focuses on the definition of pipelines for data integration and interoperability among heterogeneous systems.

6. Conclusions and future work

In this paper, we presented the Chimera framework that addresses the problem of data interoperability in a multi-stakeholder ecosystem through the definition of semantic data transformation pipelines. The Chimera framework has been implemented by extending Apache Camel to offer a flexible and configurable solution for integrating semantic data transformation pipelines considering different data formats across heterogeneous systems. Chimera specifically defines Camel components to address four building blocks: graph construction, graph transformation, graph validation and graph exploitation. The framework reduces the effort required to define custom integration solutions by providing configurable components that can be integrated into the overall production-ready ecosystem of Camel components. We compared Chimera with existing tools and we highlighted how we advanced the framework with respect to its initial implementation, in particular, to facilitate its reusability and extensibility by the community. Finally, the successful usage of Chimera was exemplified through a multimodal traffic management use case that shows both data format and information system heterogeneity in a scenario of data exchange between different and independent stakeholders. The Chimera framework is open-source, actively developed and maintained on GitHub under an Apache-2.0 license. A complete tutorial is available documenting the expected usage of the different components and exemplifying the definition of Chimera pipelines.

In future work, we would like to explore the integration of Chimera into existing graphical tools for the construction and visualization of Apache Camel pipelines, e.g., Apache Karavan[26]. Such integration would simplify the definition of semantic data transformation pipelines also by suggesting the set of available parameters and configurations for each component. Moreover, within the SmartEdge[27] project we will investigate the adoption of the presented approach on the edge (resource-constrained devices) and on the cloud (to offer high scalability), e.g., by investigating the ecosystem of deployment runtimes supported by Apache Camel and/or by defining new Chimera components. Finally, to facilitate the adoption of Chimera components within Apache Camel projects, we plan to release Chimera to the Maven Central[28] repository.

[24]Talend4SW, cf. https://github.com/fbelleau/talend4sw.
[25]https://github.com/zazuko/barnard59
[26]https://github.com/apache/camel-karavan
[27]https://www.smart-edge.eu/
[28]https://search.maven.org/

Acknowledgments

The presented research was partially supported by the TANGENT project (Grant Agreement no. 881825), co-funded by the European Commission under the Horizon 2020 Framework Programme, and the SmartEdge project, co-funded under the Horizon Europe Framework Programme (Grant Agreement no. 101092908).

References

[1] M. Scrocca, M. Comerio, A. Carenini, I. Celino, Turning transport data to comply with EU standards while enabling a multimodal transport knowledge graph, in: Proceedings of the 19th International Semantic Web Conference, volume 12507, Springer, 2020, pp. 411–429. doi:10.1007/978-3-030-62466-8_26.

[2] G. Vetere, M. Lenzerini, Models for semantic interoperability in service-oriented architectures, IBM Systems Journal 44 (2005) 887–903. doi:10.1147/sj.444.0887.

[3] D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester, A. Dimou, Declarative rdf graph generation from heterogeneous (semi-)structured data: A systematic literature review, Journal of Web Semantics (2022) 100753.

[4] M. Sadeghi, P. Buchníček, A. Carenini, O. Corcho, S. Gogos, M. Rossi, R. Santoro, et al., SPRINT: Semantics for PerfoRmant and scalable INteroperability of multimodal Transport, in: 8th Transport Research Arena TRA 2020, 2020, pp. 1–10. URL: http://hdl.handle.net/11311/1132635.

[5] Shapes Constraint Language (SHACL), 2017. URL: https://www.w3.org/TR/shacl/.

[6] G. Hohpe, B. Woolf, Enterprise integration patterns: Designing, building, and deploying messaging solutions, Addison-Wesley Professional, 2004.

[7] D. Brickley, R. V. Guha, B. McBride, Rdf schema 1.1, W3C recommendation 25 (2014) 2004–2014.

[8] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, R. V. de Walle, RML: A generic language for integrated RDF mappings of heterogeneous data, in: Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), volume 1184, CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[9] M. Scrocca, A. Carenini, M. Comerio, I. Celino, Semantic Conversion of Transport Data Adopting Declarative Mappings: An Evaluation of Performance and Scalability, in: D. Chaves-Fraga, P. Colpaert, M. Sadeghi, M. Scrocca, M. Comerio (Eds.), Proceedings of the 3rd International Workshop Semantics And The Web For Transport, volume 2939 of *CEUR Workshop Proceedings*, CEUR, Online, September, 2021. URL: https://ceur-ws.org/Vol-2939/#paper2, iSSN: 1613-0073.

[10] J. Arenas-Guerrero, M. Scrocca, A. Iglesias-Molina, J. Toledo, L. Pozo-Gilo, D. Doña, Ó. Corcho, D. Chaves-Fraga, Knowledge graph construction with R2RML and RML: an ETL system-based overview, in: D. Chaves-Fraga, A. Dimou, P. Heyvaert, F. Priyatna, J. F. Sequeda (Eds.), Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021),

Online, June 6, 2021, volume 2873 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2873/paper11.pdf.

[11] C. Allocca, A. Gougousis, A preliminary investigation of reversing rml: From an rdf dataset to its column-based data source, Biodiversity data journal 3 (2015) e5464. doi:10.3897/BDJ.3.e5464.

[12] O. Corby, C. F. Zucker, F. Gandon, SPARQL template: a transformation language for RDF, Ph.D. thesis, Inria, 2014.

[13] J. Arenas-Guerrero, D. Chaves-Fraga, J. Toledo, M. S. Pérez, O. Corcho, Morph-KGC: Scalable knowledge graph materialization with mapping partitions, Semantic Web Preprint (2022) 1–20. doi:10.3233/SW-223135, publisher: IOS Press.

[14] D. Van Assche, G. Haesendonck, G. De Mulder, T. Delva, P. Heyvaert, B. De Meester, A. Dimou, Leveraging web of things w3c recommendations for knowledge graphs generation, in: M. Brambilla, R. Chbeir, F. Frasincar, I. Manolescu (Eds.), Web Engineering, Springer International Publishing, Cham, 2021, pp. 337–352.

[15] T. Knap, M. Kukhar, B. Macháč, P. Škoda, J. Tomeš, J. Vojt, Unifiedviews: An etl framework for sustainable rdf data processing, in: European Semantic Web Conference, Springer, 2014, pp. 379–383.

[16] J. Klímek, P. Škoda, M. Nečaskỳ, Linkedpipes etl: Evolved linked data preparation, in: European Semantic Web Conference, Springer, 2016, pp. 95–100.