DataSpecer: A Model-Driven Approach to Managing Data Specifications*

Štěpán Stenchlák[0000−0003−4843−2470], Martin Nečaský[0000−0002−5186−7734], Petr Škoda[0000−0002−2732−9370], and Jakub Klímek[0000−0001−7234−3051]

Charles University, Faculty of Mathematics and Physics, Department of Software Engineering, Malostranské nám. 25, 118 00 Praha 1, Czechia

Abstract. In this paper, we demonstrate DataSpecer, a tool for effortless management of data specifications based on a domain ontology. Using DataSpecer, the users can generate technical artifacts such as data schemas, e.g., in JSON Schema or XML Schema, and human-readable documentation for a specific dataset based on the provided ontology while maintaining the semantic mapping from the generated artifacts to the ontology. This significantly eases the task of developing data specifications and keeping the corresponding technical artifacts consistent in the process. The tool is based on a previously studied model-driven development (MDD) approach [5] that divides data modeling into layers. This approach was already partially implemented in the tools XCase [2] and eXolutio [3], however, only for XML Schemas and only based on a manually created model, not on an existing domain ontology. Our current tool provides support for the implementation of artifact generators for any target format, including human-readable documentation, and supports domain ontologies as the starting point of the MDD. The tool is already in use in Czechia with the Semantic Government Vocabulary (SGOV) [4] serving as the domain ontology.

Keywords: data schema · data modeling · ontology · automation

1 Introduction

To support the exchange of structured data among different parties, a common data specification must be agreed upon. To create the specification, the parties must agree on the domain semantics of the data to be exchanged. The result of the agreement is typically recorded in the form of a domain ontology and its documentation. Then, a data schema based on a chosen data format, such as CSV, XML, JSON or RDF, is created. The schema defines a specific data structure to be used to express the exchanged data. It typically comprises several data fields, i.e. CSV columns, XML elements, JSON keys, or RDF class and predicate IRIs. The data specification itself then contains human-readable documentation and a set of technical artifacts such as the data schema, data examples, etc.

* This work was partially supported by the Charles University grant SVV 260588.

Developing the technical artifacts can be a complex and error-prone task. This is further complicated as different versions of the artifacts must be maintained. Moreover, in complex data exchange environments, there are data providers and consumers with different needs and technical backgrounds. Agreeing on a single data format may be unnecessarily difficult, and the result is typically not comfortable for everyone. Therefore, it would be useful if multiple formats could be defined while the semantic consistency of their defining technical artifacts would be preserved. However, the more technical artifacts there are to be maintained, the bigger is the challenge to keep all of them consistent.

Consider as an example the exchange of open data about tourist destinations. The data can be published both by a small village trying to attract tourists to visit using only basic information about interesting places and by a large tourist agency managing detailed information about destinations in a whole country. On the other hand, there are also consumers with different needs. Advanced consumers require the RDF representation respecting the domain ontology. Application programmers expect data available in JSON or XML formats depending on their favorite data processing libraries. Data scientists or journalists use tools that usually support only tabular data expressed in CSV or JSON.

In this demonstration, we present DataSpecer, a tool that automates the design and maintenance of such a complex set of semantically related technical artifacts from a given domain ontology. We demonstrate how the tool can be used to define a specification of data formats for exchanging open data about tourist destinations. As the domain ontology, we use an existing ontology defined and used by the Ministry of the Interior of the Czech Republic (MinIntCZ) called Semantic Government Vocabulary (SGOV) [4]. We show how two data structures for publishing open data about tourist destinations can be created from the domain ontology in DataSpecer using a graphical user interface. We also show how data schemas and human-readable documentation defining different data formats for these two data structures can be generated automatically by technical artifact generators. The purpose of this demo is to demonstrate the overall approach to generating technical artifacts. Specific technical artifact generators are our work-in-progress. Currently, DataSpecer supports generators of XML and JSON schemas and their human-readable documentation.

2 DataSpecer

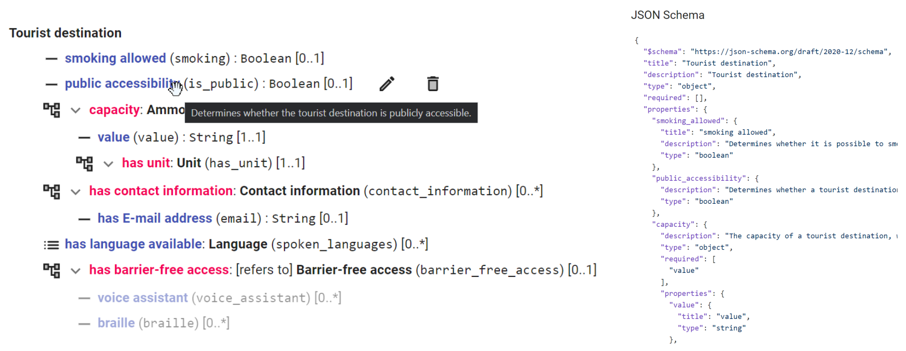
DataSpecer[1] follows the model-driven development (MDD) approach. It considers models at three levels of MDD. The computation independent model (CIM) represents the domain semantics which is defined by a given domain ontology. The platform independent model (PIM) is a computational model which describes technical but platform independent details for expressing domain data. For example, it adds data types and technical labels to concepts from CIM. The platform specific model (PSM) specifies the details of how the data is expressed in various data formats and structures.

[1] `https://dataspecer.com/`

At the CIM level, DataSpecer reads a given domain ontology on the input and extracts classes and properties. An ontology can be expressed in various ontology specification languages. The typical languages are RDF Schema or OWL. However, there are more options, such as SHACL or the Unified Foundational Ontology (UFO) [1]. There are also proprietary approaches used by Schema.org or Wikidata to express ontologies. DataSpecer is not restricted to a specific ontology language. For a given language, it needs a specific adapter. We currently support only ontologies based on UFO-A, a subset of UFO incorporating only endurants - objects and their ontology, applied by MinIntCZ in SGOV [4] to design various government ontologies.

PIM is represented as a subset of CIM. It describes the semantics of a given data specification comprising one or more data structures. PSM is then represented by individual data structures designed in DataSpecer. To design a data specification comprising a set of data structures, a designer follows this typical workflow, which is also demonstrated in a video[2]:

1. Data specification First, the designer initializes the data specification, which is a set of data structures to express the same data using different structures. In our demonstration, we aim at a sample data specification for publishing open data about tourist destinations.



Tourist destination

— smoking allowed (smoking) : Boolean [0..1]

— public accessibility (is_public) : Boolean [0..1]

capacity: Ammo

Determines whether the tourist destination is publicly accessible.

— value (value) : String [1..1]

has unit: Unit (has_unit) [1..1]

has contact information: Contact information (contact_information) [0..*]

— has E-mail address (email) : String [0..1]

has language available: Language (spoken_languages) [0..*]

has barrier-free access: [refers to] Barrier-free access (barrier_free_access) [0..1]

— voice assistant (voice_assistant) [0..*]

— braille (braille) [0..*]

JSON Schema

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Tourist destination",
  "description": "Tourist destination",
  "type": "object",
  "required": [],
  "properties": {
    "smoking_allowed": {
      "title": "smoking allowed",
      "description": "Determines whether it is possible to sm
      "type": "boolean"
    },
    "public_accessibility": {
      "description": "Determines whether a tourist destinatio
      "type": "boolean"
    },
    "capacity": {
      "description": "The capacity of a tourist destination,
      "type": "object",
      "required": [
        "value"
      ],
      "properties": {
        "value": {
          "title": "value",
          "type": "string"
        },
```

Fig. 1: A data structure for Tourist destinations represented in DataSpecer and its JSON schema. Attributes are blue; edges are red.

2. Data structures The designer then creates one or more data structures in the data specification. A data structure is a rooted oriented tree that is agnostic to a specific serialization in a chosen syntax (see Figure 1). Its nodes represent classes from the domain ontology (CIM). Its edges represent properties from the ontology connecting the classes. The nodes specify individual data fields for representing data semantically described by the ontology. The edges represent how the data fields are nested in the data structure.

In our demonstration, the designer creates two data structures. The first one aims at publishing generic information about tourist destinations, such as their

[2] `https://youtu.be/lZoM794_uFk`

titles and contact points. The other defines how information about the accessibility of tourist destinations shall be published. For a new empty data structure, the designer selects its root by selecting a class from the CIM, i.e., the class *Tourist destination*. For a non-empty data structure, the designer can choose an existing node mapped to an ontology class and define its content by choosing one or more properties related to the class from the ontology. The designer can also refine the resulting data structure by adding technical details such as technical labels to be used in the schemas, data types, etc. In the demonstration, we show how the designer selects *Tourist destination* as a root for both data structures and how different content for them can be derived from the CIM.

DataSpecer represents the data structure created by the designer as a PSM mapped to the PIM of the data specification. The tool maintains the PIM as a subset of the CIM automatically as the designer creates new nodes and edges in the PSM. We demonstrate these mappings from the data structures to the CIM by showing how chosen nodes and edges are mapped to the domain ontology.

3. Data schemas and their previews For each data structure, the designer can choose a data format in which the data structure shall be represented. DataSpecer currently supports XML and JSON. The data structure, together with the chosen generic format, defines a specific data format. In the demonstration, we will show how the designer can choose the data formats and how the corresponding data schemas can be generated not only when a data structure is finished but also during its creation as a live preview.

4. Technical artifacts generation Finally, the designer requests DataSpecer to generate technical artifacts for the specified data formats. DataSpecer executes the selected technical artifacts generators and bundles the result as a single archive which can be downloaded by the designer. Currently, DataSpecer contains generators for XML Schema, JSON Schema[3], and a human-readable documentation in Bikeshed[4]. In the demonstration, we will show how the designer can download the technical artifacts for both resulting specific data formats.

3 Current Usage and Future Work

DataSpecer has been applied by the MinIntCZ to design recommended JSON data structures for publishing open data. This included data specifications for tourist destinations, bulletin boards of local governments, etc. Another application in the context of MinIntCZ was to design XML data structures for sharing data among government information systems. We demonstrated the usage of DataSpecer on a set of information systems in the field of road traffic and personal transport. While the first application produces only JSON structures and the second only XML structures, both rely on SGOV as the common ontology.

[3] `https://json-schema.org/`

[4] `https://tabatkins.github.io/bikeshed/`

DataSpecer is still under development. We are working on adding support for domain ontologies expressed in RDF Schema and OWL as well as for proprietary ontologies such as Schema.org and Wikidata as CIM. We are also working on adding support for the CSV format and support for more detailed configuration for data schema generators regarding preferred constructs in the individual schema languages. Our vision is that data specification authors use existing public ontologies, possibly combined with their own ontologies, to design their data specifications in a systematic, MDD based way.

We are also working on the interoperability of the generated data specifications. In the same way, we generate data schemas, we are working on deriving transformations among the individual data structures. For each data structure, there will be a lifting transformation transforming the data expressed using this structure to the RDF representation corresponding to the domain ontology. Moreover, there will be a lowering transformation transforming the data from the RDF representation back to the defined structure. Therefore, data represented using one data structure can be transformed to a representation using another data structure via the RDF representation using the generated transformations.

Another crucial objective is supporting modifications and change propagation, both on the ontology and schema levels. Changes introduced in the ontology will be used to derive new schemas automatically, thus creating data transformations between the old and new schemas.

Last but not least, DataSpecer currently supports only manual data structure design, where the designer needs to go through the individual properties of a chosen class from the domain ontology and manually select which will be represented in the data structure. We plan to extend DataSpecer with support for a semi-automated process of data structure definition.

References

1. Guizzardi, G., Benevides, A.B., Fonseca, C.M., Porello, D., Almeida, J.P.A., Sales, T.P.: UFO: Unified Foundational Ontology. Applied Ontology **Pre-press**(Pre-press), 1–44 (2021), https://content.iospress.com/articles/applied-ontology/ao210256
2. Klímek, J., Kopenec, L., Loupal, P., Malý, J.: XCase - A Tool for Conceptual XML Data Modeling. In: Advances in Databases and Information Systems, Associated Workshops and Doctoral Consortium of the 13th East European Conference, ADBIS 2009, Riga, Latvia, September 7-10, 2009. Revised Selected Papers. LNCS, vol. 5968, pp. 96–103. Springer (2009). https://doi.org/10.1007/978-3-642-12082-4_13
3. Klímek, J., Malý, J., Nečaský, M., Holubová, I.: eXolutio: Methodology for Design and Evolution of XML Schemas Using Conceptual Modeling. Informatica **26**(3), 453–472 (2015), https://content.iospress.com/articles/informatica/inf1065
4. Křemen, P., Nečaský, M.: Improving discoverability of open government data with rich metadata descriptions using semantic government vocabulary. J. Web Semant. **55**, 1–20 (2019). https://doi.org/10.1016/j.websem.2018.12.009
5. Nečaský, M., Mlýnková, I., Klímek, J., Malý, J.: When conceptual model meets grammar: A dual approach to XML data modeling. Data & Knowledge Engineering **72**, 1–30 (2012). https://doi.org/10.1016/j.datak.2011.09.002