Model-driven approach in data specification modeling and management

Štěpán Stenchlák
Department of Software Engineering
Faculty of Mathematics and Physics, Charles University
Prague, Czech Republic
stepan.stenchlak@matfyz.cuni.cz

***Abstract*—The description of data structure and semantics is critical for large domains to ensure the correct interpretation and make data interoperable and consistent across different systems. Traditional data modeling processes often overlook that semantic and structure design are two separate disciplines, leading to bad data modeling practices. Also, in many domains, the models are not created from scratch, causing other problems with synchronization and reuse. We propose a robust framework on which tools will be built to support and ease the development and management of data specifications mapped to semantically or structurally richer models and keep them consistent through evolution.**

***Index Terms*—data semantics, data structure, conceptual models, structural models, model-driven**

I. Problem

A data specification describes the structure and semantics of data stored within systems or used for communication between various parties. The structure determines how the data are represented, whereas semantics defines meaning and how the concepts should be interpreted. The degree of explicitness of expressing structure and semantics depends on a specific use case. In the case of small domains, the data itself can implicitly describe its structure, and the structure can describe the semantics if the person knows the domain well.

However, in complex domains where potential design flaws would be costly, the precise description of both syntax and semantics may be essential [1]. As the syntax is technical, it is often sufficient to provide a technical schema. However, for semantics, documentation with diagrams and examples may be helpful. Some domains may benefit from additional documents, such as application skeletons, API specifications, tests, etc. The whole package describing the data will then be called the **data specification**.

Manually designing such data specifications is a laborious and error-prone task. It may require much monotone work with copying names and definitions and may be easy to miss concepts or misplace texts. The designer needs to know the semantics well and be able to create all necessary parts of the data specification. The latter task can be handled by numerous tools automatically generating various documents. For the first task, numerous tools have emerged, typically based on the model-driven approach [2], which assists with the design by deriving the structure from conceptual models. However, we observe a prevalent issue with these tools leading to poor practice. They typically do not adequately separate the semantics and structure, thereby forcing users to think about conceptual models in the context of structure. The lack of separation often results in poor-quality conceptual models that do not reflect reality. For example, the designer may merge two concepts for optimization reasons, even if they are semantically different. The ideal way is to have a conceptual model focused on rich semantics, the so-called semantic model [3] (from now on we will use the term semantic model in this paper), isolated from the process of designing a structure [1], [4], [5]. Moreover, while semantic models require a perfect knowledge of the domain, for data structure design, familiarity with technical parameters is needed. However, the separation of structural and semantic models brings several problems, such as the availability of tools [6], [7] that can handle the separation, keeping the models consistent and up-to-date, or well-defined methodologies how to manage everything properly.

The idea of the model-driven approach of deriving structural models from semantic models, described above, can be carried over to the design of semantic models themselves. In some cases, such as maintaining interoperability or easing work, it may be beneficial to reuse existing models instead of creating them from scratch. These are often application profiles that define a semantic model for use in specific application scenarios but are based on more generic, application-independent semantic models.

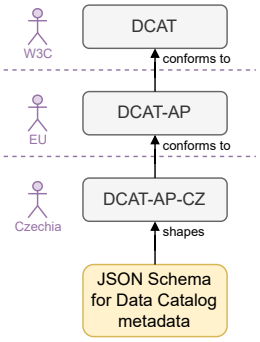
An example is DCAT-AP[1]. DCAT [8] is the Data Catalog Vocabulary from W3C designed to facilitate interoperability between data catalogs published on the Web. DCAT-AP is then an extension of the vocabulary for the needs of the European Union. DCAT-AP-CZ [9] for the Czech Republic is derived from it. Each adds or changes some of its specifics. From DCAT-AP-CZ is then derived a specific data specification defining a JSON data structure [10], but with the semantics described in DCAT, DCAT-AP and DCAT-AP-CZ. See Figure 1a.

Since these application profiles are manually created, they

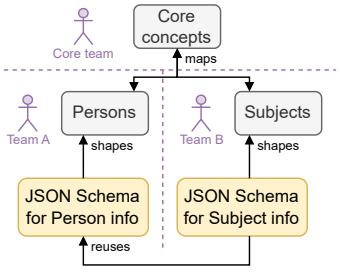
This research is partially supported by SVV project number 260 698 and by the Charles University, project GA UK No 262823.

[1]https://joinup.ec.europa.eu/collection/semic-support-centre/solution/dcat-application-profile-data-portals-europe/release/300

may have similar issues as the derivation of structural models. They are potentially error-prone, hard to develop, and may not be sustainable in the long term. For example, any change in DCAT must be propagated manually to data that conforms to the JSON Schema according to DCAT-AP-CZ. Also, as the process usually includes only creating the technical specification, the expert needs to be familiar with both the semantics and the technical format of the specification. Hence there are greater demands on designers.



(a) Data Catalog Vocabulary and its Application Profiles for use in Czechia's National Open Data Catalog.



(b) Subset of the study information system for persons and school subjects management modules with schemas for provided APIs.

Fig. 1: Real-life examples of using semantic models (in grey) managed by different groups together with structural models (in yellow).

Another example is from microservices architectures. Here, we have multiple models. Each service usually has one semantic model, if any, developed separately [11]. However, for mutual communication, it is necessary to ensure semantic mapping between shared concepts and mutual propagation of changes if the semantics of concepts in the domain change.

In more open domains, each team may use a different tool for its model creation. We also see, with the DCAT example, that some models may already exist. These external models further complicate interoperability as experts must know the technical format of the model and import it into their ecosystem.

In a similar meaningful way to extending semantic models, it may also be desirable to extend structural models. Either to build larger ones by compositing them from smaller or derive new ones based on existing ones. Assume a semantic model from which a structural model describing the base interface is derived, which will be used in other domains, but appropriately extended with additional data. Then we need to keep the models synchronized at the structural level and allow designers to easily derive new ones so that potential errors do not arise again. See Figure 1b, where the structural model for *Subjects* reuses *Persons*.

As a complete example, consider the following:

1) A domain expert decides to create a semantic model for their domain and derive a few JSON Schemas from it describing data exchanged between the system. They may create a semantic model from scratch, but they decide to use Wikidata [12] and Schema.org as it already describes most of their domain. This means they may choose concepts that will be included, may change them, or add new ones as only sometimes the existing models describe the reality as wanted. They may want to create a documentation website from the finished semantic model with diagrams and detailed explanations of individual concepts.
2) If the domain is large, the model may be further extended or modified by other teams that need more detail or have different expectations.
3) Data designers may then create structural models that shape the domain's semantic model. Each structural model is then used to generate schemas for given formats, documentation, examples, and other artifacts.
4) Others may extend the structural models by creating their own or use them as is in other schemas.

The process would be easier to coordinate with supporting tools that can also manage changes in requirements in the given domain. This would mean propagating all changes and publishing new documents with information on what has changed to mitigate the potential errors during a manual uncontrolled update.

The problem can be summarized as follows: Designing a data specification in larger domains is not just designing a simple data schema or semantic model. Such a data specification is a complex system of existing semantic models, its complementary and extending semantic models, models of data structures derived from them, and mappings between them. Several documents (description, example data, example queries, application mockups) and technical artifacts (schemas, transformation scripts) complement this. Creating and maintaining such a complex process is laborious and leads to errors. Moreover, the designer of the data specification must have the freedom they have when they follow the now common path of designing a simple data schema in which they can do whatever they want and is not constrained by external semantic models.

What is needed is a tool that allows users, when designing their data specifications, to build on existing semantic and structural models, reuse them, combine and modify them, and ultimately derive data structure models from them, if necessary, tied to the original semantics, but without constraining them in any way. Managing all this manually is hard. The tool will make it easier by automating as many steps as possible, managing the dependencies between models and their elements, and through these dependencies, also supporting propagation.

However, to our best knowledge, currently, available tools are either impractical or assume significant coupling between models.

Research question: Is it feasible to design a framework and build a set of tools on it for the management of semantic and structural models, which would be practically applicable and lead to an improvement in the process of data specifications development?

II. Proposed solution

The issues described in the previous chapter can be viewed through a model-driven approach. Any semantic or structural model managed atomically by a single entity is a model from a model-driven standpoint. In Figure 1, all rectangles are individual models, and we also consider external models, such as one constructed in Enterprise Architect and stored in an SQL database.

Then the problems addressed in the previous chapter can be divided into three groups. (i) Change propagation, model derivation from more general models, and model management can be addressed by a generic model-driven based framework. (ii) We then build tools for deriving semantic and (iii) structural models on top of this general framework. A high-level representation of our solution is shown in Figure 2.

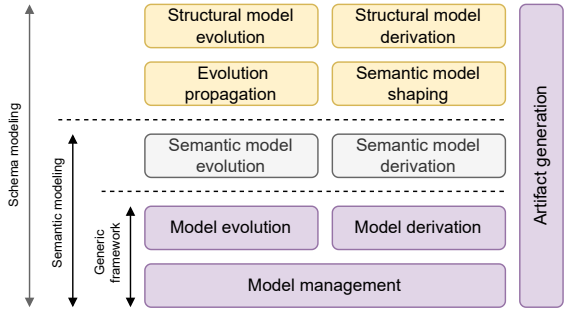


Fig. 2: Proposed decomposition of the project into logical units with a generic framework on which semantic modeling and schema modeling tools will be built.

To generalize the idea, by **model**, we mean a representation of either structure, semantics, or any other representation of arbitrary data. This allows us to build other applications that may benefit from change propagation and model derivation and are not directly related to semantic or structural models. For example, consider an application that manages OpenAPI specifications[2]. Although most of the specification consists of data schemas, some operations may change in time and may be necessary to propagate changes between various specifications and other technical artifacts properly.

The generic framework aims to provide model management and access. For the project's success, the tools must also be easy to use from the command line or via GitHub Actions.

To achieve that models can be extended and modified, we introduce a **model derivation** as seen in Figure 1. Individual models will refer to each other, creating a rooted **model graph** with the model of a question as the root. According to the type of reference and based on the context, the information from the referenced models will be interpreted differently, resulting in a single aggregated model with the desired properties. This does not mean that the base model contains all the content but rather consists of another model. Depending on the situation, we may treat the base model as an aggregation, hence the full model, and generate, for example, a documentation of all concepts with diagrams, or just as a patch that fixes the included model and generate only the patch documentation. The base model may overwrite, modify or delete entities from the included model or include only explicitly listed entities. This may be beneficial, for example, with the Wikidata model, as it is too large, and only part of it may be relevant in a given situation.

Individual types of entities that will be stored in the model depend on the model type and use, and their exact list and function are out of the scope of this paper. To briefly mention them, the semantic model will contain classes and associations; the structural model will contain objects and their properties, choices and references. If the model modifies an existing concept, it will be a patch. Other useful entities include various types of mappings, such as sameAs, or more complex ones between multiple entities.

The construction of a model graph will be further configurable for individual tools that need to know the context of individual models. Let us imagine a situation where we will make an application profile to a semantic model. In this case, we have two models - the original model, which may be external, and the model that patches it. The newly created concepts then need to be stored in the new model. However, it may make sense to put any bug fixes directly into the original model or to create a new model containing bug fixes that are then applied to the original model. Hence the tools should be aware of the model graph and how the individual models are intended to be used.

Alternatively, we may have multiple external models as a source. Then it may be helpful to create a model for potential bug fixes for each external model and then one model that aggregates all of them and provides additional entities that do not belong to any of the source models.

If there is a designated model for bug fixes, it should be possible to generate a patch to the original model or directly commit the changes if the model would support this.

Two sub-activities will be addressed regarding the evolution: model versioning and change propagation. Versioning is only possible for models created within the tools and not externally. The purpose is to keep the whole model history and allow the change propagation to be executed later, not immediately, when change occurs. Change propagation will then be handled individually within the semantic and structural models. Each change to the model will have to be made using a predefined operation. With the knowledge of the operations, it will be possible to apply this list of operations to other models or translate them to a different set of operations for models of other types.

For external models that are not versioned, we would have to obtain the list of operations manually. We can address this by using another model as a mirror. By comparing these models, we can infer changes that can then be propagated. In some cases, this may not be necessary. Then, if the inconsistency is found, users would have to fix it by themselves.

The individual models may serve as the output if they are in the desired format. Although we plan to introduce a native format for storing the models, users may use adapters for other

[2]https://swagger.io/specification/

formats if the given format disposes of the required functionalities. Then we call the model to be **external**. External models can be a common use case for representing semantic models in various formats, such as Enterprise Architect or Draw.io.

However, in most scenarios, users may want to **generate artifacts** from models using various generators. For example, a JSON Schema from a structural model, RDFS [13] from a semantic model, documentation, etc. From an architectural perspective, three types of generators can be distinguished.

- *single model* - generates artifacts from one model in a specific version. Either directly from the model or by aggregating it with all models from the model graph,
- *evolution* - generates from a sequence of changes that are propagated from the model, for example, SQL UPDATE statements or an HTML document with a list of changes in a human-readable form,
- *diff* - the difference between two models, such as transformation scripts that produce data that conforms to one structural model from data that conforms to the other. This, however, is the most challenging part of the project and, in many use cases, unnecessary.

III. Related work

This work builds on our previous research [14]–[17] from 2012 to 2015 of the author's supervisor and his team. The team developed a tool eXolutio for deriving XML [18] schemas from semantic models. The approach allowed the creation of a semantic model without a particular schema in mind, and data structures could be derived and automatically updated in case of changes in the semantic model. However, the approach was limited to only two models, semantic and structural, without the inclusion of external semantic models as the starting point. Furthermore, it only supported XML.

Some European countries are pursuing a similar goal in developing tools for publishing data specifications to describe open data according to the New European Interoperability Framework guide [4]. In some parts, it identifies the issues described in section I, thus confirming the correct direction of our research and the possible application in practice. Hoverer, the quality of these tools is often low regarding the general and easy use outside their domain.

An example is the Finnish tool Tietomallit[3], which focuses only on creating semantic models. Thus, it does not allow the designer to modify the resulting data structure and assumes only simple models with implicit structure, which is sufficient for their domain. Moreover, the tool itself is currently not reusable outside of the Finnish use case due to dependencies on the Finnish environment.

A Belgian toolchain for technical standardization on a national level OSLO[4] also focuses only on semantics without any customization for data schemas. It uses Enterprise Architect to design semantic models from which it can generate various artifacts automatically.

[3]https://tietomallit.suomi.fi/

[4]https://joinup.ec.europa.eu/collection/oslo-open-standards-linked-organisations-0/about

Neither of the tools supports change propagation or multiple model management.

The Linked Data Modeling Language LinkML [19] is a modeling language and a more technical tool for generating various artifacts from a YAML document containing easy-to-understand descriptions of concepts, their relations, and other data to author schemas. It supports various formats and can also transform data between them by loading them into its internal representation. However, the tool also binds the resulting schema structure to the semantic model that the user must design, forcing the user to contemplate the specific schema rather than the concepts that the schema describes. Given the wide range of supported technologies and their use, we plan to implement a LinkML model generator into our tool so that the resulting structure is also instantly usable in these tools.

There are also many more generic tools for creating semantic models [20]. However, to our knowledge, they only support creation and editing and do not deal with the advanced features mentioned in this paper, such as multiple models, patches or generation. However, for the success of the tools, their integration will have to be considered.

IV. Current status

The project started with the author's master thesis, which aimed to develop a proof-of-concept tool called Dataspecer [21] to facilitate and improve the work of creating so-called Formal Open Standards (FOSes) [22] in the Czech Republic. FOSes are data specifications for open data publishers and public institutions. Currently, the tool can effectively replace most of the original manual work and is actively used for that. It generates HTML documentation of the concepts with diagrams, JSON Schema [10], JSON-LD [23], XSD [24], CSVW [25], SPARQL queries [26], and XSLT [27] transformation scripts between XML and RDF [28] representation.

A key consideration during the tool's development was user-friendliness. It reads an external semantic model and suggests concepts that can be used to construct the schema in a graphical bullet list-like format. See Figure 3. It lets users rename keys and move properties in the tree to create schemas that fit their needs. The format is not tied to a specific technology but rather generic. Therefore it is possible to generate JSON, XML, and CSV schemas from the same structural model. Possible format-specific constructs can be added, but our proposed model is sufficient for all the FOSes we are working on.

Regarding the graph model described in section II, the three-layer architecture is currently embedded in Dataspecter. The user chooses the source of the external semantic model. A new, empty model is created underneath, which is used to patch changes and simultaneously, copies all concepts used in case the external semantic model is unavailable. Underneath are the individual structural models from which the tool generates the artifacts, as mentioned earlier.

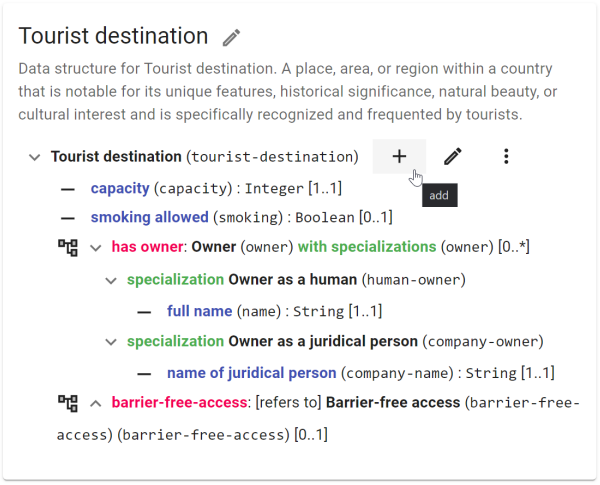


Fig. 3: Screenshot of Dataspecer tool with a structural model for tourist destinations. This is a human-friendly view, where the owner can be of two types. From this, for example, a JSON Schema with a oneOf construct can be generated together with HTML documentation describing all used concepts.

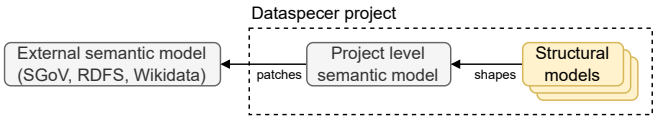


Fig. 4: The current fixed graph model of Dataspecer.

Although this architecture is sufficient for many cases, we would like to generalize it for other scenarios, such as instances where the user works with multiple semantic models or wants to create their own model during schema modeling.

We use the Czech Semantic Government Vocabulary (SGoV) [29] as the external semantic model. It contains concepts from Czech laws and important domains, their definitions, and relations. For example, it contains concepts about Tourist destinations that are used in FOS to prescribe a JSON structure for publishing open data about them. A typical publisher is a local city administration that needs to attract visitors by providing standardized data that various online mapping and planning services can consume. It also contains more than 40 different properties characterizing the barrier-free accessibility of a public place that requires a deeper understanding of the needs of people with disabilities. Hence it would be too demanding for one person to design a structure without semantics defined in advance.

We are also actively working with several students on other generators such as SHACL, sample application generator, or example data generator. We are also working on a Wikidata adapter as an external semantic model.

In addition to a schema generation tool, we want to focus on a tool that would work primarily with semantic models - load external ones, create application profiles, and publish those in different formats and technologies according to section II.

In the following year (the 2nd year of the author's Ph.D. studies), the author plans to complete all parts of the generic framework, redesign Dataspecer to support a full model graph, and coordinate the development of the semantic modeling tool. In the next year, we plan to start working on evolution and methodology based on the feedback we get. We will also optimize the developed tools for use in practice. In the last year, we want to analyze and implement mappings between different concepts and further work on integrating the tools into practice.

From a publishing point of view, the author wants to describe the structure of the models and how they can be applied in different domains, the principle of evolution, and mapping.

V. Plan for Evaluation and Validation

The primary part of the evaluation and validation will be the use of the tools in practice. Specifically, several data experts will actively use the tool for schema generation in the context of FOSes development and management for Czechia. Evolution will also be used in long-term management, as the source semantic models continuously change. Occasionally, the semantic model creation tool will also be used to create application profiles.

The author will be actively using the semantic model management tool to manage concepts from the business analysis as he is involved in the development of a new student information system. Due to the size of the project, multiple semantic models will be created and need to be synchronized. This corresponds to Figure 1b with several semantic models and one main model containing the definition of the core concepts. Subsequently, Dataspecer will be used to create API specifications for the interfaces between the different modules of the system. As the system is incrementally rewritten, the semantic model will evolve. The changes will need to be documented, and API updates will need to be issued accordingly. We expect the tools to do most of the work automatically.

We also actively seek involvement in other projects, such as European Union's SEMIC, where data specifications are being developed.

We then plan to conduct a methodological survey among users to determine whether these tools make their work more effortless than a manual approach, whether all use cases can be covered, and whether manual intervention in the model design process is still needed.

VI. Expected Contributions

The author's expected contribution includes the design and implementation of the essential components of the system and the project management. We expect an easy-to-use ecosystem of tools for work with semantic and structural models and integration with other existing tools and technologies for semantic models. Furthermore, we will propose a methodology for creating and managing semantic models and data specifications to eliminate typical mistakes. Finally, we will continue using our tools and methodology in practice.

References

[1] A. Doan, A. Y. Halevy, and Z. G. Ives, *Principles of Data Integration*. Morgan Kaufmann, 2012. [Online]. Available: http://research.cs.wisc.edu/dibook/

[2] D. C. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering," *Computer*, vol. 39, no. 2, pp. 25–31, 2006.

[3] J. Peckham and F. Maryanski, "Semantic data models," *ACM Computing Surveys (CSUR)*, vol. 20, no. 3, pp. 153–189, 1988.

[4] E. Commission and D.-G. for Informatics, *New European interoperability framework : promoting seamless services and data flows for European public administrations*. Publications Office, 2017.

[5] View, "Towards a model-driven organization (part 1) – anchor modeling," Aug 2022. [Online]. Available: https://www.anchormodeling.com/towards-a-model-driven-organization-part-1/

[6] F. Rademacher, J. Sorgalla, and S. Sachweh, "Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective," *IEEE Softw.*, vol. 35, no. 3, pp. 36–43, 2018.

[7] P. Espinoza-Arias, D. Garijo, and Ó. Corcho, "Crossing the chasm between ontology engineering and application development: A survey," *J. Web Semant.*, vol. 70, p. 100655, 2021.

[8] D. Browning, S. Cox, R. Albertoni, A. Perego, A. G. Beltran, and P. Winstanley, "Data Catalog Vocabulary (DCAT) - Version 2," W3C, W3C Recommendation, Feb. 2020. [Online]. Available: https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/

[9] J. Klímek, "Dcat-ap representation of czech national open data catalog and its impact," *Journal of Web Semantics*, vol. 55, pp. 69–85, 2019.

[10] A. Wright, H. Andrews, B. Hutton, and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents," Internet Engineering Task Force, Internet-Draft draft-bhutton-json-schema-01, Jun. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-bhutton-json-schema/01/

[11] S. Newman, *Building Microservices*. O'Reilly Media, Inc., 2021. [Online]. Available: https://www.oreilly.com/library/view/building-microservices-2nd/9781492034018/

[12] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[13] D. Brickley and R. Guha, "RDF Schema 1.1," W3C, W3C Recommendation, Feb. 2014. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/

[14] M. Nečaský, I. Mlýnková, J. Klímek, and J. Malý, "When conceptual model meets grammar: A dual approach to XML data modeling," *Data & Knowledge Engineering*, vol. 72, pp. 1–30, 2012.

[15] M. Nečaský, J. Klímek, J. Malý, and I. Mlýnková, "Evolution and change management of XML-based systems," *Journal of Systems and Software*, vol. 85, no. 3, pp. 683–707, 2012.

[16] J. Klímek, L. Kopenec, P. Loupal, and J. Malý, "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*, ser. LNCS, vol. 5968. Springer, 2009, pp. 96–103.

[17] J. Klímek, J. Malý, M. Nečaský, and I. Holubová, "eXolutio: Methodology for Design and Evolution of XML Schemas Using Conceptual Modeling," *Informatica*, vol. 26, no. 3, pp. 453–472, 2015. [Online]. Available: https://content.iospress.com/articles/informatica/inf1065

[18] M. Sperberg-McQueen, E. Maler, F. Yergeau, T. Bray, and J. Paoli, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," W3C, W3C Recommendation, Nov. 2008. [Online]. Available: https://www.w3.org/TR/2008/REC-xml-20081126/

[19] S. A. T. Moxon, H. Solbrig, D. R. Unni, D. Jiao, R. M. Bruskiewich, J. P. Balhoff, G. Vaidya, W. D. Duncan, H. Hegde, M. Miller, M. H. Brush, N. L. Harris, M. A. Haendel, and C. J. Mungall, "The Linked Data Modeling Language (LinkML): A General-Purpose Data Modeling Framework Grounded in Machine-Readable Semantics," in *Proceedings of the International Conference on Biomedical Ontologies 2021 co-located with the Workshop on Ontologies for the Behavioural and Social Sciences (OntoBess 2021) as part of the Bolzano Summer of Knowledge (BOSK 2021), Bozen-Bolzano, Italy, 16-18 September, 2021*, ser. CEUR Workshop Proceedings, J. Hastings and A. Barton, Eds., vol. 3073. CEUR-WS.org, 2021, pp. 148–151. [Online]. Available: https://ceur-ws.org/Vol-3073/paper24.pdf

[20] T. Tudorache, "Ontology engineering: Current state, challenges, and future directions," *Semantic Web*, vol. 11, no. 1, pp. 125–138, 2020.

[21] Š. Stenchlák, M. Nečaský, P. Škoda, and J. Klímek, "Dataspecer: A model-driven approach to managing data specifications," in *European Semantic Web Conference*. Springer, 2022, pp. 52–56.

[22] "Directive (EU) 2019/1024 of the European Parliament and of the Council of 20 June 2019 on open data and the re-use of public sector information." [Online]. Available: http://data.europa.eu/eli/dir/2019/1024/oj

[23] P.-A. Champin, D. Longley, and G. Kellogg, "JSON-LD 1.1," W3C, W3C Recommendation, Jul. 2020. [Online]. Available: https://www.w3.org/TR/2020/REC-json-ld11-20200716/

[24] H. Thompson, D. Beech, M. Maloney, N. Mendelsohn, M. Sperberg-McQueen, and S. Gao, "W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures," W3C, W3C Recommendation, Apr. 2012. [Online]. Available: https://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/

[25] G. Kellogg and J. Tennison, "Model for Tabular Data and Metadata on the Web," W3C, W3C Recommendation, Dec. 2015. [Online]. Available: https://www.w3.org/TR/2015/REC-tabular-data-model-20151217/

[26] A. Seaborne and S. Harris, "SPARQL 1.1 query language," W3C, W3C Recommendation, Mar. 2013, https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[27] M. Kay, "XSL transformations (XSLT) version 3.0," W3C, W3C Recommendation, Jun. 2017, https://www.w3.org/TR/2017/REC-xslt-30-20170608/.

[28] Y. Raimond and G. Schreiber, "RDF 1.1 Primer," W3C, W3C Note, Jun. 2014. [Online]. Available: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

[29] P. Křemen and M. Nečaský, "Improving discoverability of open government data with rich metadata descriptions using semantic government vocabulary," *J. Web Semant.*, vol. 55, pp. 1–20, 2019.