

Specification and Implementation of Mapping Rule Visualization and Editing: MapVOWL and the RMLEditor

Pieter Heyvaert^{a,1,*}, Anastasia Dimou^{a,1}, Ben De Meester^a, Tom Seymoens^b, Aron-Levi Herregodts^c,
Ruben Verborgh^a, Dimitri Schuurman^c, Erik Mannens^a

^a*IDLab, Department of Electronics and Information Systems, Ghent University – imec, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium*

^b*imec – smit, Vrije Universiteit Brussel, Pleinlaan 9, 1050 Etterbeek, Belgium*

^c*imec – Ghent University – MICT, Korte Meer 7, 9000 Gent, Belgium*

Abstract

Visual tools are implemented to help users in defining how to generate Linked Data from raw data. This is possible thanks to mapping languages which enable detaching mapping rules from the implementation that executes them. However, no thorough research has been conducted so far on how to visualize such mapping rules, especially if they become large and require considering multiple heterogeneous raw data sources and transformed data values. In the past, we proposed the RMLEditor, a visual graph-based user interface, which allows users to easily create mapping rules for generating Linked Data from raw data. In this paper, we build on top of our existing work: we (i) specify a visual notation for graph visualizations used to represent mapping rules, (ii) introduce an approach for manipulating rules when large visualizations emerge, and (iii) propose an approach to uniformly visualize data fraction of raw data sources combined with an interactive interface for uniform data fraction transformations. We perform two additional comparative user studies. The first one compares the use of the visual notation to present mapping rules to the use of a mapping language directly, which reveals that the visual notation is preferred. The second one compares the use of the graph-based RMLEditor for creating mapping rules to the form-based RMLx Visual Editor, which reveals that graph-based visualizations are preferred to create mapping rules through the use of our proposed visual notation and uniform representation of heterogeneous data sources and data values.

Keywords: graph, Linked Data, mapping rule, MapVOWL, RMLEditor

1. Introduction

Nowadays Linked Data still stems from (semi-)structured formats. A few of the most well-known and larger Linked Data sets² [23] are: DBpedia dataset³ [6], with approximately 1.39 billion triples derived from Wikipedia⁴ where the data is originally represented in the wikitext syntax; Linked Geo Data⁵ [58] with approximately 1.38 billion triples derived from Open Street Map planet files

loaded in multiple databases; UniProt [12] (UniProtKB, Uniref and UniParc), with approximately 45 billion triples across 3 datasets derived from the UniProt Knowledgebase⁶; and Bio2RDF⁷ [2], with approximately 11 billion triples across 35 datasets.

Overall, Linked Data generation includes the following: (i) *multiple heterogeneous raw data sources* whose data values might need transformation, (ii) *ontologies* used to annotate the RDF terms [13] that are generated from the data fractions of the different data sources, (iii) *actual mapping rules* which define how data fractions are semantically annotated and used to generate RDF terms and triples, and (iv) *generated Linked Data*.

*Corresponding author

Email address: heyvaer.heyvaert@ugent.be
(Pieter Heyvaert)

¹These authors contributed equally to this work.

²<http://stats.lod2.eu/rdfdocs?sort=triples>

³<http://dbpedia.org>

⁴<http://wikipedia.org>

⁵<http://linkedgedata.org>

⁶<http://www.uniprot.org/help/uniprotkb>

⁷<http://bio2rdf.org/>

Several datasets are generated by tools that incorporate directly in their implementation how Linked Data is generated. This means when new or updated semantic annotations are needed, knowledge of Semantic Web technologies is required, as well as dedicated software development cycles for adjusting and extending the implementations.

To the contrary, mapping rules may also be defined, according to a specified mapping language syntax, such as R2RML [16] or RML [20]. Mapping languages define declaratively how terms are generated from corresponding raw data and annotated with ontology terms to form the desired Linked Data. This way, mapping rules are detached from the implementation that executes them. Nevertheless, knowledge of the underlying mapping language is required to define mapping rules, while manually editing and curating them requires a substantial amount of human effort [31]. Therefore, the creation of mapping rules still remains complicated.

To this end, a significant number of mapping editors were implemented to facilitate mapping rules creation and editing, such as Map-On [56], the RMLEditor [31], and the RMLx Visual Editor⁸. Only a few of them provide graph-based visualizations, although user evaluations suggest that such visualizations are suitable for supporting users to intuitively generate their desired Linked Data [31].

Nevertheless, how such user interfaces should be designed is not thoroughly investigated so far: (i) a *visual notation* specification for mapping rules does not exist. Such a specification would provide a formal description for mapping rules and allows multiple tools to implement it, improving the accessibility for users across different tools; (ii) current mapping editors do not uniformly present *multiple heterogeneous data sources*. Therefore, creating mapping rules that define relationships between heterogeneous data sources is not always straightforward; (iii) *data value transformations* cannot be defined in current visualization-based mapping editors; (iv) *scalability* is not thoroughly addressed. Even if graph-based visualizations are used, large graphs cause difficulties to users when editing the corresponding mapping rules.

In this work, we present and extend our ongoing work towards a uniform graphical user interface (GUI) to create and edit Linked Data mapping rules. Such a GUI is implemented in the RMLEditor,

which we presented in the past. The RMLEditor offers a graph-based interface for specifying mapping rules for raw data to generate Linked Data. Its target group of users have knowledge about both Linked Data and the domain of the data. The mapping rules creation and editing is based on graph visualizations, without requiring knowledge of the underlying mapping language. Our novel contributions include in particular:

- (i) a rich graph-based visual notation for mapping rule visualization;
- (ii) an approach for manipulating rules when large visualizations emerge;
- (iii) an approach to uniformly visualize data fractions of data sources combined with an interactive interface for uniform data fraction transformations;
- (iv) an implementation of these three contributions in the RMLEditor; and
- (v) additional evaluations to compare the use of
 - the visual notation to present mapping rules to the use of a mapping language directly, which reveals that the visual notation is preferred; and
 - the graph-based RMLEditor to create mapping rules to the form-based RMLx Visual Editor, which reveals that the RMLEditor is preferred to create mapping rules through its use of the visual notation and uniform representation of heterogeneous data sources and data values.

The remainder of the paper is structured as follows. In Section 2, we elaborate on Linked Data, mapping rules, and RDF terms. In Section 3, we discuss related work. In Section 4, we introduce our research questions and hypotheses. We present in Section 5 our proposed visual notation for mapping rules, and in Section 6, the RMLEditor. In particular, we elaborate in Section 6.5 on the manipulation of large graphs in the RMLEditor, and in Section 6.6 on how the RMLEditor deals with heterogeneous data sources. In Section 7, we present the evaluations and the results both for the visual notation and the new version of the RMLEditor. In Section 8, we summarize this work’s conclusions.

2. Preliminary

Linked Data refers to data whose meaning is explicitly defined, that is published on the Web in a machine-interpretable way, and that is linked to other external data sets [5]. Nowadays, *RDF* [13] is the prevalent framework to represent Linked Data. Most of the time, Linked Data originally stems from

⁸<http://pebbie.org/mashup/rml>

(semi-)structured formats (CSV, XML, and so on). Their RDF representation is obtained by repetitively applying mapping rules according to an iteration pattern which specifies the extract of data that is considered during each iteration.

Mapping rules define correspondences between data in different schemas [24]. In the case of Linked Data generation, mapping rules define how RDF terms, i.e., IRIs, literals or blank nodes [13], are generated from data fractions derived from one or more data sources which are annotated with ontologies. These RDF terms are used to form RDF triples.

With the term *data fractions*, we do not only mean *raw data values* as they are in the original data source, but also *transformed data values* that result from a raw data value after applying a function to process the original data values. Data value transformations are needed to support changes in the structure, representation or content of data, such as string transformations. For instance, when a data fraction contains a date in the format “DD-MM-YYYY”, it might be needed to be transformed to the format “YYYY-MM-DD”.

References to raw or transformed data fractions might be combined with constant values (*template-valued* [16]) or only constant data values (*constant-valued* [16]) might be used to form the desired RDF terms. As such, a complete mapping rule includes (i) a reference to zero or more raw data fractions and (ii) one or more ontology terms. Therefore, when visualizing a mapping rule, its interrelation with the raw data as well as with its semantic annotation with an ontology term should be visualized when presented to and edited by users.

3. State of the Art

The creation and editing of mapping rules, as well as their visualization is closely related to Linked Data, ontologies, and query visualizations (Section 3.1), and is performed using different types of mapping editors (Section 3.2).

3.1. Interfaces for Semantic Web Visualizations

In this section, we elaborate on visualizations for different aspects of the Semantic Web: Linked Data, ontologies, and queries which become relevant. We discuss the characteristics of each aspect and their visualizations that are implemented in existing tools or formalized as a specification.

3.1.1. Linked Data Visualizations

As mapping rules resample how Linked Data will eventually be generated [31], visualizations that are applicable for Linked Data would be expected to be applicable for mapping rules visualizations too.

Efforts to improve Linked Data accessibility, resulted in tools offering either (i) *text-based* presentations, or (ii) *visualizations*. Dadzie and Rowe [15] conducted a survey on both approaches. They concluded that text-based solutions, such as Sig.ma [60], (i) fail to provide an overview of the available information, and (ii) are only suitable for users with understanding of the underlying technologies, whereas lay users need additional support.

Visual presentations lead to approaches relying on network maps [35], diagrams [35], geographic maps [53], timelines [53], charts [3], and graphs [27, 62, 19]. The latter was the default in the past according to Dadzie and Pietriga [14], because (i) ontologies are often hierarchically structured and used to annotate Linked Data; (ii) RDF’s data model is a directed labeled graph [13]; and (iii) network analysis is one of the most common visualization-driven tasks carried out within the field, to explore, e.g., collaborations and other interrelationships between researchers within research data, and social networks at large. However, they fail to provide meaningful visualizations when graphs become large, even when styled to better convey the resources’ and properties’ semantics [49]. Furthermore, exposing the data’s graph structure is not always needed, because the model may be of little importance to users [14]. Therefore, current efforts are more focused on user interface components designed for different types of data, such as temporal and geographical data. Nonetheless, graph visualizations can still be used, but they are no longer the main component to be represented.

Regardless of the used visualization, research is being done to improve the support for large datasets. For example, Bikakis et al. [4] introduce a generic model for organizing, and analyzing numeric and temporal data in a multilevel fashion to deal with the challenges that come with the use of large datasets, such as information overload. The model is not tied to a specific visualization and, thus, it can be used with any of the aforementioned tools to improve the support for such datasets.

3.1.2. Ontology Visualizations

Ontology visualizations are also close to mapping rule visualizations. Mapping rules define the mod-

eling, namely the conceptualization, of raw data as Linked Data, using ontologies. Thus, approaches applicable for ontology representation which visualize the schema could be adjusted for visualizing mapping rules too. To improve the ontology presentation and editing, a number of tools were developed. Such tools offer visualizations that represent ontology-specific elements. Only a limited number of efforts defined a specification for such visual notations that can be implemented by any other tool.

Tools. Graphs offer a natural way to depict the structure of ontological elements and relationships [40]. Therefore, graph-based visualizations are present in a high number of visualization tools for ontologies, such as GrOWL [36], OWLViz [33] and KC-Viz [43]. In these cases, classes and datatypes are represented as nodes and properties as edges. However, when similar graphical notations are used for different ontological elements it is difficult for users to distinguish them.

Furthermore, a number of graph-based visualizations are focused on a single task, e.g., providing insights regarding the class hierarchy, and neglect other aspects of ontologies, including the properties and datatypes. This makes them less suitable for other tasks [40], such as gaining insights in the relationships between the classes via the different properties. When visualizing large ontologies the graphs might become too large for users to cope with. Different approaches, applied by several tools [43, 38, 34], were developed to tackle this challenge: (i) showing detailed information only on demand, i.e., selecting a specific node results in displaying more detailed information about that node; (ii) displaying the parts of the graphs that are selected by a given filter; (iii) zooming in and out on the graphs, enabling them to gain more information about a specific region of interest of the graph.

So far, ontology visualizations were focused on presenting the ontologies. Nevertheless, more and more tools are built to facilitate editing of ontologies [45, 11], as ontologies are subject to change. This leads to the development of visualization plugins that show the (updated) ontology inside the tool [39]. However, such an approach disconnects the visualization of the ontology from its editing and requires users to understand both the visualization and the interface used to perform the updates. This is overcome when graph-based visualizations allow users to update the nodes and edges [36, 57]. The updates will result in the cor-

responding changes in the ontology. TurtleEditor [57], for instance, uses both methods: text editor and visualization to present and update the ontology, leaving the choice to the users.

Specifications for Visual Notations for Ontologies. In most cases, tools create a new stand-alone visualization to present the ontology [43, 34]. However, when users switch between tools they need to learn a new visualization to work with the new tool. Recent efforts result in the creation of specifications for visual notations. They provide a formal description of the visual elements used to convey the required information to users. These specifications are independent of the tools that implement them, and, thus, can be applied by multiple tools. The latter allows to improve the user's ability to switch between these tools as they provide the same predefined visualization [25]. The most significant specifications are Graffoo [25] and VOWL [40].

The Graphical Framework For OWL Ontologies (Graffoo) [25] defines a graph-based visual notation for ontologies. Classes are represented with yellow rectangles and datatypes with green parallelograms. The properties are represented with lines connecting the rectangles or parallelograms, and have different colors depending on the property's type, i.e., object and datatype property. Graffoo is implemented in yEd⁹, a diagram editor. However, it does not propose an intuitive ontology visualization that is immediately understandable to casual users due to its diagrammatic approaches [40].

The Visual Notation for OWL Ontologies¹⁰ (VOWL) defines a visual language for user-oriented representation of ontologies and provides graphical depictions for elements of the Web Ontology Language (OWL) [40]. The initial specification of VOWL [44] focused on the visualization of ontology elements (i.e., classes, properties, and datatypes), together with the dataset's instances, the so-called TBox and ABox in Description Logic, respectively. However, based on their user study, they concluded that even when visualizing only a few instances, providing additional information already leads to difficulties for understanding the visualization.

Therefore, VOWL 2 focuses on visualizing only the TBox. It relies on directed graphs to visualize ontologies, and is implemented in WebVOWL [38], a web application. Classes are represented by blue,

⁹<http://www.yworks.com/products/yed>

¹⁰<http://purl.org/vowl/spec/>

circular nodes, and datatypes by yellow, rectangular nodes. Links are used to visualize how classes are related to other classes or datatypes through properties. Graphical elements are added to links between classes to represent characteristics, such as disjoint and union, while the use of colors helps identifying the different elements.

3.1.3. Query Visualizations

Query visualizations aim to hide the query language to end users, whereas mapping rule visualizations aim to hide the mapping language. Thus, approaches applicable for query representation could be adjusted for visualizing mapping rules.

Query formulation is important for the retrieval of information available as Linked Data. SPARQL [50] is the standard query language for Linked Data described using the RDF framework. However, besides users from the Semantic Web community, lay users and domain experts from different areas, i.e., users without knowledge about this language, need support to define valid queries that provide the desired results [26]. Tools, such as NITELIGHT [52], RDF-GL [32], and FedViz [22], were developed to make it easier to work with a query language by removing the burden of dealing with the language’s syntax. They apply graph-based visualizations to represent the data’s underlying RDF structure. However, knowledge about SPARQL is still required, which makes them less usable for lay users, with exception of FedViz. This tool allows to browse Linked Data through a graph-based visualization (see Section 3.1.1). Once the users have found the desired data, FedViz generate the corresponding SPARQL query.

Nonetheless, visually spotting the difference between different aspects of a query or Linked Data is difficult as the same graphical notation is used to denote different aspects. QueryVOWL [26] addresses this ambiguity. It is a specification for a visual notation for queries that uses graph visualizations, as it is based on VOWL. QueryVOWL specifies a visual query language, with the goal to be accessible for lay users while still preserving most of SPARQL’s expressiveness. Referents are represented as circular nodes and literal values as rectangular nodes. A referent’s class is added inside the node as text. The same is done for the datatype of a literal value. Links represent the relationships between nodes, using properties. Different colors are used to make distinction between classes, instances, and literals.

Icons represent the update actions to the queries, such as add and delete.

As queries can become large, it is required to support large graphs. Similar to ontology visualizations, different approaches were proposed to tackle the corresponding challenges. For example, details are presented on-demand when users click on a part of the query [26], specific details are shown by applying filters on the query’s results [26], or users can zoom in on specific parts of the query through the corresponding buttons [52].

Specific tools are also developed for specific cases, such as SPEX [53] for spatial and temporal data. Their goal is to provide users with a GUI to explore the data and, subsequently, to help with query construction. Scheider et al. [53] introduced design principles to achieve this goal, such as the use of the results’ feedback into the construction and the automatically handling of space and time data. SPEX, which follows these principles, includes also a graph-based visualization, similar to the aforementioned tools. However, these tools do not follow the design principles making them less usable than SPEX.

3.2. Mapping Editors

Research efforts to facilitate the mapping rule creation and editing to generate Linked Data resulted in the development of two types of graphical user interface (GUI) tools: (i) *step-by-step wizards* and (ii) *visualization tools*.

In the past, **step-by-step wizards** prevailed as an easy-to-reach solution, such as the fluidOps editor [54]. However, such applications restrict data publishers’ editing options, hamper altering parameters in previous steps, and detach mapping definitions from the overall knowledge modeling, since related information is separated in different steps. These tools circumvent dealing with the underlying mapping languages’ syntax, but users are still required to have knowledge of the language’s terminology, limiting thus the support for users who do not have this knowledge.

More recent tools incorporate **graph-based visualizations** to present mappings. A limited number of them also apply these visualizations to edit the mapping rules, such as Map-On [56] and the RMLEditor [31]. We distinguish three prevalent approaches for mapping rule visualizations: (i) visualizing mapping rules as Linked Data; (ii) separate graph visualizations for the raw data and on-

tology; (iii) single graph visualization for both raw data and ontologies.

The first approach is applied by the RMLx Visual Editor¹¹. It visualizes mapping rules by using graphs as if the rules are Linked Data, aiming to present them to the users rather than editing them. This is possible when the mappings are defined in RDF syntax, which is the case for R2RML and RML, while editing is still performed relying on a *step-by-step wizard*. Consequently, users (i) need to understand the mapping language’s terminology to correctly interpret these graph visualizations; (ii) can only view and not edit the graphs, so mappings can only be edited using forms; (iii) need to ‘imagine’ how the triples will look like as the visualization does not communicate how the mapping rules result in the corresponding RDF triples.

The second approach is applied by Map-On. Two graphs are created: one that represents the raw data structure and one that represents the target ontology which is predefined and cannot be adjusted after being loaded. Mapping rules are created by aligning the two graphs, i.e., aligning the data fractions with the corresponding ontology elements. The two graphs are distinguished from each other using different colors. However, by visualizing both the raw data and ontology in the same visualization, the graphs quickly become cluttered and which aggravates when both of them are large.

The third approach is applied by the RMLEditor. The visualization aims to represent the RDF triples that the mapping rules generate. The nodes represent how RDF terms of subjects and objects are generated, while the edges represent the predicates’ RDF terms. More details regarding the RMLEditor are available in Section 6.

4. Problem Statement

Based on the aforementioned, we notice that *graph-based visualizations* is the dominant approach when tools present Linked Data, ontologies, and queries to users, while these visualizations also have revealed benefits for visualizing mapping rules. Linked Data and ontology visualizations mainly aim to present to the users, whereas queries require the users to actively interact with and shape the visualized object, as it also occurs with mapping

rules editing. Moreover, so far, Semantic Web related visualizations present only one of the components involved in Linked Data generation, namely either Linked Data or ontologies, which do not require to interrelate multiple components. However, the mapping rule definition involves the interrelation of the different involved components.

A number of open issues remain regarding mapping rules visualization and editing: (i) the definition of a *visual notation specification for graph-based visualizations of mapping rules*, (ii) *scalability* issues when large graphs emerge, (iii) *uniform representation for heterogeneous data sources*, and (iv) *data values transformation* integration with the presentation of these data sources. More details are presented in Section 4.1, followed by our research questions and hypotheses in Section 4.2.

4.1. Open Issues

Specification. The development of a visual notation specification increases the visualization’s adoption by other tools. Current research efforts and mapping editors neither specify nor implement such a specification. This impedes the users accessibility.

Scalability. Dealing with large graphs is an important challenge for graph-based visualizations. The same occurs when mapping rules are edited by using graphs. However, none of the existing tools tackle this challenge, besides barely applying zooming. The latter is not always sufficient as for users it is not always straightforward to know on which part of the graph they need to zoom in.

Heterogeneity. Linked Data might originally stem from multiple data sources with heterogeneous data formats. Therefore, mapping rules and their corresponding visualizations need to support these data sources. Most existing tools support neither multiple nor heterogeneous data sources. The RMLEditor and the RMLx Visual Editor are the only ones that support multiple heterogeneous data sources. The latter does not show the raw data, while the former does. Nevertheless, even only showing the raw data makes it difficult to derive the data model and especially its data fractions.

Data Transformations. Although, transformations are required when raw data values are desired to be altered to generate RDF terms, only the RMLx Visual Editor provides this functionality. In other

¹¹<http://pebbie.org/mashup/rml>

cases, they are often implemented as custom solutions, or addressed by dedicated applications, thus the range of possible transformations is limited, and cannot be visualized uniformly to the users.

4.2. Research Questions and Hypotheses

Given these open issues, we define the following two research questions:

Q1 How can we design graph-based visualizations that improve the cognitive effectiveness of mapping rules visual representations when generating Linked Data?

Q2 How can we visualize the components of a Linked Data generation process – that is based on legacy non-RDF and (semi-)structured data sources – to improve its cognitive effectiveness?

This question has two sub-questions:

- How to deal with large mapping rules graphs?
- How to uniformly visualize heterogeneous data fractions and their transformations?

Note that cognitive effectiveness is defined as the speed, ease, and accuracy with which a representation can be processed by the human mind [37].

To address Q1, we introduce a visual notation for mapping rules called MapVOWL (see Section 5). To address Q2 and its sub-questions, we introduce an approach to deal with heterogeneous data sources and data values (see Section 6.6) and an approach to improve the understanding of large graph-based visualizations (see Section 6.5). These approaches are implemented in the RMLEditor. Furthermore, these questions lead to two corresponding hypotheses which are validated through two comparative user studies (see Section 7):

H1 MapVOWL improves the cognitive effectiveness of the mapping rules graph-based visual representation to generate the Linked Data compared to using a mapping language directly.

H2 The cognitive effectiveness provided by the RMLEditor’s GUI improves the user’s performance during the Linked Data generation process – that is based on legacy non-RDF and (semi-)structured data sources – compared to the state of the art.

5. Visual Notation for Mapping Rules

In the past, we showed that graph-based visualizations are applicable for presenting mapping rules [31]. It was also confirmed to be an intuitive way to represent the structure of ontologies in a comparative evaluation [40]. So, we rely on *graph-based ontology visualizations* and we investigated how to apply their approaches on corresponding *graph-based mapping rule visualizations*. Therefore, we based the specification for a mapping rule visualization on VOWL, leading to MapVOWL.

5.1. Requirements

The notation needs to support rules visualization that generate triples which, on their own turn, conform with the RDF specification. Each triple consists of three elements: subject, predicate, and object, which are RDF terms: IRI, blank node, or literal. A literal might have a datatype or language.

These terms can be generated based on fractions from the raw data, constant values, or a combination of the two. The data can be annotated by classes, properties, and datatypes from different ontologies.

Furthermore, a notation has to be cognitively effective. In the case of mapping rules this includes the understanding of the different components of the mapping process and their relationships: multiple heterogeneous raw data sources, ontologies, mapping rules, and the generated Linked Data. Therefore, a cognitive effective notation needs to support rules that define how to generate

R1.1 a RDF term: IRI, blank node, and literal;

R1.2 a triple’s subject (IRI/blank node), predicate (IRI), and object (IRI/blank node/literal).

More, a notation should support references to

R1.3 data values of data sources,

R1.4 constant values,

R1.5 a combination of data and constant values

R1.6 ontological elements from various ontologies.

In Linked Data visualizations similar requirements were fulfilled in order to depict the RDF’s data model, together with the used ontologies. However, they did not include the need to visualize the relationships with the raw data.

5.2. MapVOWL

We introduce MapVOWL, a user-oriented visual notation for mapping rules which defines how Linked Data is generated from raw data. MapVOWL builds on top of VOWL’s graphical elements [40]. Relying on MapVOWL’s unified graphical elements, the mapping rules can be created and edited entirely using visual representations, while the mapping rules in the underlying mapping language’s syntax are generated by the mapping editor without user intervention. This way, MapVOWL hides the underlying mapping language, as QueryVOWL hides the query language.

Overall, MapVOWL aims to make mapping rules creation and editing for generating Linked Data more accessible to Linked Data experts, even more for those who already know VOWL or QueryVOWL. A comprehensive and uniform visual representation would be beneficial and would have great impact particularly on users with knowledge about the underlying mapping language, as it occurs in the case of ontology visualizations [21].

The MapVOWL specification can be found at <http://rml.io/mapvowl>. Moreover, we applied MapVOWL to the RMLEditor. You may access a demo instance of the RMLEditor with MapVOWL implemented at <http://rml.io/jws-demo>. Moreover, you may find a screencast explaining the graph elements of MapVOWL as adopted in the RMLEditor at <http://rml.io/jws-screencast>.






In Section 5.2.1, we present the graphical primitives. In Section 5.2.2, we discuss the color scheme. In Section 5.2.3, we elaborate on the visual elements. In Section 5.2.4, we discuss how MapVOWL improves the cognitive effectiveness.

A user study between MapVOWL and RML compares the use of the visual notation to present mapping rules to the use of a mapping language directly (see Section 7.1), while a user study between the RMLEditor implementing MapVOWL and the RMLx Visual Editor compares the use of a *graph-based* GUI to a *form-based* GUI to create mapping rules (see Section 7.2).

5.2.1. Graphical Primitives

MapVOWL provides a small set of unambiguous graphical primitives based on VOWL, as shown in Table 1. It is designed to be modular and, thus, it distinguishes the primary elements that are crucial for the Linked Data generation, such as the RDF terms to be generated, from elements which provide supplementary details, such as their (data)types.

Table 1: Graphical Primitives

Primitive	Application
	referents
	blank nodes
	relationship labels and literal values
	relationships
	relationship direction
text	textual information about mapping rules

Referents and literals are depicted as nodes of the mapping rule graph visualization, whereas the relationships between them form the graph’s edges. This choice is made because the shape plays a special role in discriminating between symbols as it represents the primary basis on which we identify objects in the real world; the shape is the primary visual variable for distinguishing between different constructs.

Shape. In the case of VOWL, the shape is the graphical primitive used to distinguish the classes from datatypes which are the two fundamental constructs of ontologies. Following the same principle, in the case of mapping visualizations, the type of the RDF term which will be generated is the most fundamental construct. Thus, deviation in the node’s shape determines whether a referent (IRI or blank node) or a literal will be generated.

In the case of mapping visualizations, the constructs which can have a class assigned to them are depicted as circles (R1.1), whereas the rest has a rectangular shape (R1.1). This follows VOWL where classes are depicted as circles and datatypes as rectangles. For the classes, there is no restriction on which ontologies to use (R1.6). Constructs that are not meant to represent an entity, but are an attribute of an entity are depicted as rectangles, similarly to datatypes in VOWL and might be assigned a datatype or have the default.

Edge. MapVOWL considers directed edges whose label is defined in a rectangle to associate referents

with each other or with literals. MapVOWL uses edges as VOWL does to represent the properties which will be generated or (re)used to associate the entities among each other or the entities with the attributes (R1.2). However, even though in VOWL the property’s label is detached from the edge, MapVOWL incorporates the label in the edge using a rectangle. This way, we make it explicit that a property is another RDF term as a referent or a datatype valued node and it could be defined under the same conditions as the rest of them. Namely, we do not distinguish generation of a property from the other RDF terms. Last, a property’s direction is indicated by an arrowhead, as it is for VOWL too. This states the property’s subject and object (R1.2).

Text. While nodes and edges depict mapping rules, text is used to provide additional information about these rules. Text is added to a circle to denote (i) its class, and (ii) how the IRIs are formed, either template-valued or constant-valued (see Section 2; R1.3, R1.4, and R1.5), and to a rectangle to denote (i) the datatype or language tag (R1.1), and (ii) how the literal values are formed, either template-valued or constant-valued (R1.3, R1.4, and R1.5). For the datatype there is no restriction on which ontologies to use (R1.6). When a language is specified, the language tag is preceded with “@” and the datatype is automatically set to `rdf:langString` [13]

Border. As borders can convey meaning, they are considered as a distinct graphical primitive used to represent a certain notion. A border is only present when an entity is a blank node and then it is dashed, following VOWL’s choice for using dashed borders and properties’ lines for special classes and properties. In the case of VOWL, dashed circles are used when circles represent `owl:Thing` and, thus, they do not carry relevant domain information. Similarly, blank nodes might be generated and annotated; however, they differ from typical entities, as the ones related to blank nodes do not receive IRIs.

Size. Similarly to VOWL, MapVOWL also recommends to vary the node size if possible and desired, without determining the scaling method though. VOWL associates the node size with the number of individuals which are members of a class. However, this is not possible in the case of mapping rule visualizations as the individuals are not generated yet.

Although, this number may be estimated by analyzing the data source for the specified data fraction(s) from which the individuals are generated in the end. If the analysis is not possible or desired, all nodes should have the same predefined size.

5.2.2. Color Scheme

Color. A color scheme is employed to interrelate data sources and mapping rules. In contrast to VOWL where the color indicates different types of classes, in MapVOWL the color depends on the data source the term is (partially) derived from. If the color of a node is grey, it is not associated to any data source (yet). A color scheme is recommended by MapVOWL, but any other color scheme may be considered. Alternatively, a schema that relies on texture usage instead of different colors may be considered to support color-blinded people if needed.

Brightness. In addition to color schema, the brightness level is employed to make the visualization more comprehensive. Changes on the brightness level occurs when a user interacts with the graph. To be more precise, the brightness is higher when a graph element is not selected and darker once the element is selected by the user.

5.2.3. Visual Elements

MapVOWL defines visual elements for mapping rules. These elements are based on the aforementioned graphical primitives and color scheme. Figure 1 shows examples of these elements.

Mapping rules can be created to generate referents, i.e., they will be represented by an IRI or a blank node which can be annotated with a class to determine its type. For instance, in the examples, referents are generated for employees and their addresses (see Figure 1a). For each employee a IRI is generated, using a template, and for each address a blank node, thus its node remains gray as its definition does not depend on any data source. Their classes are `foaf:Person` and `vivo:Address`.

Mapping rules can be created to generate literals, i.e., they are represented with the default datatype or a user-specified datatype. For instance, in our example, literals are generated for the names of projects. It has the default datatype `xsd:string`. The start date of a project has the user-specified datatype `xsd:date` (Figure 1b). When a literal value is a string, users can denote the value’s language. For instance, in our example, the city of an address is provided in English (Figure 1c).

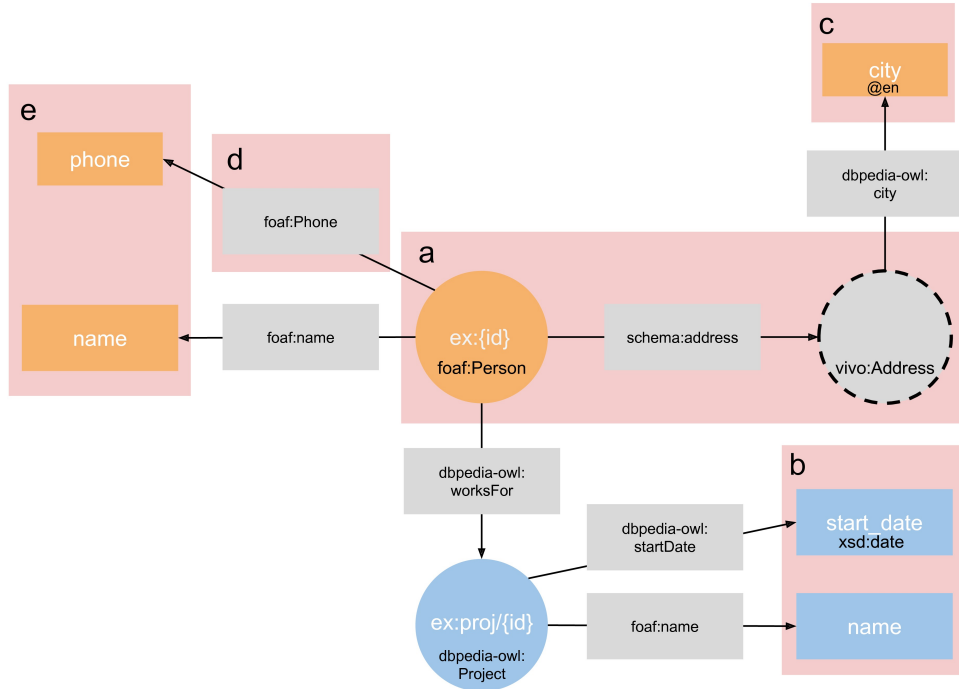


Figure 1: Examples of visual elements

Mapping rules can be created to generate properties, which can be represented by an IRI only. In our example, properties are used to denote the relationship between employees and their corresponding literals, such as their phone number using the property `foaf:phone` (see Figure 1d).

Nodes have different size based on the available data in the data source, unless the information is not available (either deliberately or ignored). In our example, all employees have a name; however, not all employees provided their phone number. This is reflected in the size of the nodes that represent these literals (see Figure 1e).

The overall visualization is combined to a graph that represents the envisaged model for the Linked Data. The position of the graph elements is imposed by a force-directed graph layout algorithm. This means that the highly connected nodes are placed more together, which puts more focus on their relationships.

5.2.4. Conform to Physics of Notations

The notation also needs to be designed to be cognitively effective. To achieve this we rely on the design theory “Physics of Notations” by Moody [42]. The theory presents a set of principles to which a

notation should adhere to be effective. The principles are semiotic clarity, perceptual discriminability, semantic transparency, complexity management, visual expressiveness, dual coding, graphic economy, coding fit, and cognitive integration. As a result the notation does not just only fulfill the requirements, but does this in a cognitive effective manner to improve the usability.

Semiotic Clarity. When specifying a visual notation it is important to consider the principle of semiotic clarity as it impacts the notation’s effectiveness of addressing the problem it tries to solve [42]. Semiotic clarity is determined by the correspondence between the semantic constructs and the graphical notations. In the case of MapVOWL, the nodes and edges of the graphs form the graphical notations, and the possible combinations of RDF terms, which the user can create to form the generated triples, are the semantic constructs (see Figure 2). The semiotic clarity is determined by the presence of four anomalies: (i) symbol redundancy, (ii) symbol overload, (iii) symbol excess, and (iv) symbol deficit.

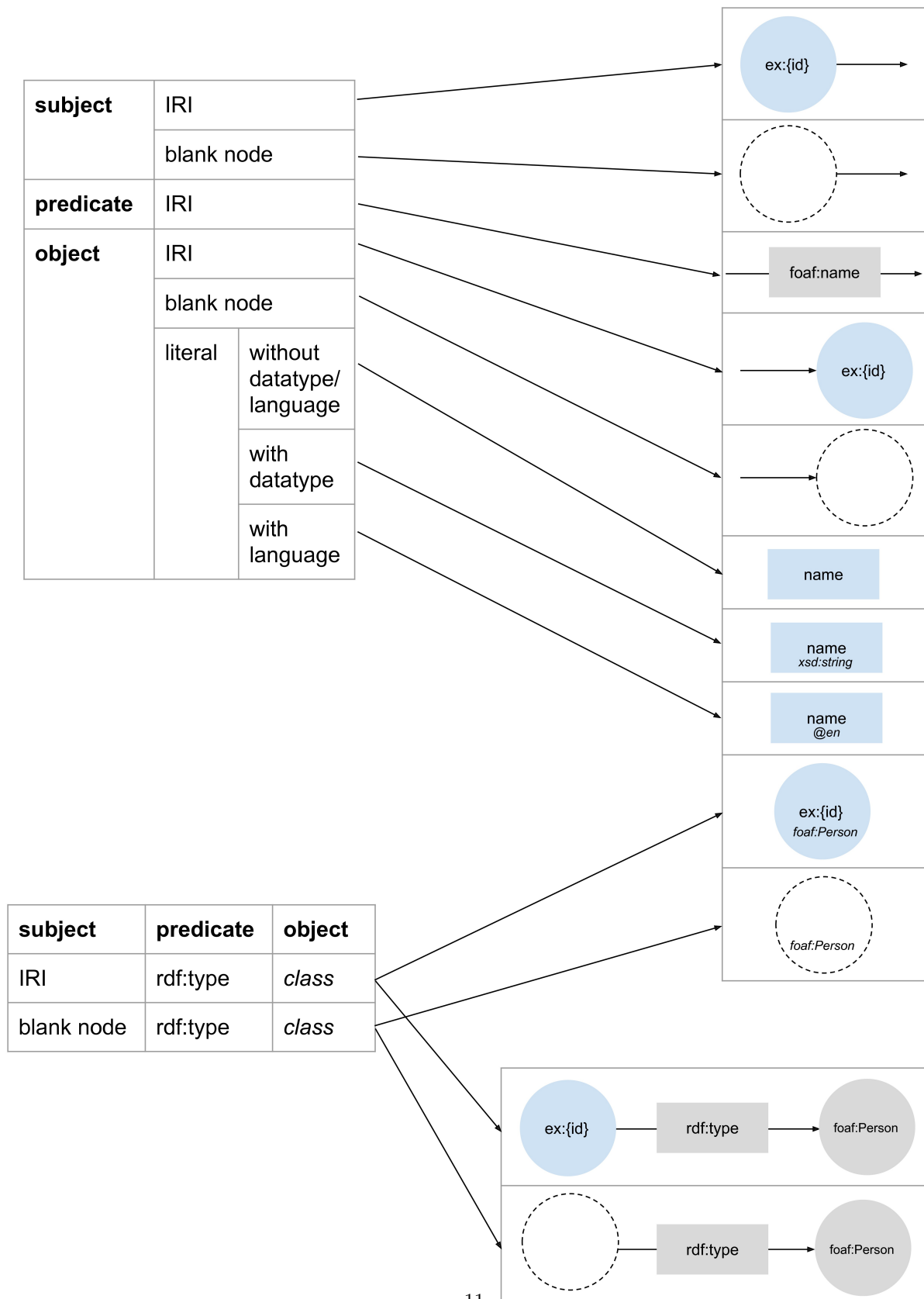


Figure 2: Semantic constructs (left) and graphical notations (right)

Symbol Redundancy When a semantic construct can be represented by multiple graphical notations, called symbol redundancy, it has two negative effects on the user. On one hand, users have difficulties deciding which notation to use. On the other hand, users need to remember multiple notations for a single construct.

In the case of MapVOWL, a subject, which might be an IRI or a blank node, is represented by a circle from which an arrow starts. An IRI is different from a blank node, because no border is used, while a dashed border is used for a blank node, i.e., two distinct notations for subject. A predicate, which is always an IRI, is represented by an arrow with a rectangle. When an object is an IRI or a blank node, it is represented by a circle where an arrow ends, which distinguishes it from a subject. When an object is a literal, a rectangle is used. The use of a datatype or language is differently visualized: the datatype is added at the bottom of the rectangle, and the language is added at the bottom of the rectangle preceded with “@”.

The class of an entity can be represented with two different notations: (i) a circle with the class at the bottom, as datatype is for literals, and (ii) a circle with an arrow and rectangle to another circle with the class. The former occurs in most cases, when the class is constant. To support rare cases where the class might depend on raw data values, the longer notation can be used, without excluding the support for a constant value. This might be a small symbol redundancy, nevertheless, it leads to a shorter notation in most cases.

Symbol Overload When a single graphical notation can represent multiple semantic constructs, called symbol overload, it leads to ambiguity and potential misinterpretation. In the case of MapVOWL, no symbol overload is observed. Each graphical notation aligns with a single semantic construct, as can be observed in Figure 2 where only a single arrow arrives for each graphical notation on the right. In detail:

Subject. A (dashed) circle from which an arrow starts aligns with a subject, corresponding with either an IRI or blank node.

Predicate. A rectangle with an arrow through it aligns with a predicate.

Object. A (dashed) circle where an arrow arrives aligns with an object, corresponding with either an IRI or blank node; a rectangle without text at the bottom with a literal without datatype or language; a rectangle with text about the datatype with a literal with a datatype; a rectangle with text preceded with “@” with a literal with a language.

Class. A circle with text at the bottom aligns with an IRI as subject, `rdf:type` as predicate, and a class as object. If this circle has a dashed border, then it aligns with a blank node as subject instead of IRI. A circle with an arrow and rectangle to a circle with text in the center aligns with an IRI as subject, `rdf:type` as predicate and a class as object. If this notation has dashed border for the first circle, then it aligns with a blank node as subject instead of IRI.

Symbol Excess When a graphical notation is used that does not represent a semantic construct, called symbol excess, it clutters the visualization and increases the complexity of the visualization. Both have a negative impact on the understanding by users. In the case of MapVOWL, no symbol excess is present as for each graphical notation there is a corresponding semantic construct, as explained for symbol overload (at least one arrow arrives for each graphical notation on the right in Figure 2).

Symbol Deficit As opposed to the aforementioned anomalies, symbol deficit is desired to limit the diagrammatic complexity. Symbol deficit occurs if not all semantic constructs are represented by a corresponding graphical notation. The deficit is desired when representing all semantic constructs would reduce, instead of improve, the notation’s cognitive effectiveness. In the case of MapVOWL, no symbol deficit was introduced as the graphical notations required to represent all constructs do not lead to a complex notation (Figure 2).

Perceptual Discriminability. Perceptual discriminability is the ease and accuracy with which graphical notations can be differentiated from each other [42]. For mapping rules, it is important to perceive the difference between (i) the components of a triple (subject, predicate, and object), and (ii) the RDF terms (IRI, blank node, and literal). The former is done by using directed graphs, thus, an edge has an explicit direction, where the start is the subject and the end is the object. The latter is done by using a circle without a border for an IRI,

Table 2: Visual variables

Variable	Power	Capacity
horizontal position	internal	10-15
vertical position	interval	10-15
size	interval	20
brightness	ordinal	6-7
colour	nominal	7-10
texture	nominal	2-5
shape	nominal	unlimited
orientation	nominal	4

and a circle with a dashed border for a blank node, regardless of whether they are subject or object. A rectangle is used on the edge between two nodes to represent the IRI, which is the only option for a predicate. A rectangle is also used for literals, but it is distinct since it can only be at the end position of the edge as it can only be the object of a triple.

Semantic Transparency. Semantic transparency is defined as the extent to which the meaning of a notation can be inferred from its appearance [42]. When Linked Data is presented using RDF, RDF graphs represent the model of the data. By using graph-based visualizations for mapping rules, users already see the model that will be used for the generated RDF triples, which form an RDF graph. Nodes in the mapping graphs that are the start/end of an edge will result in subjects/objects of triples in the RDF graph. The edges of the graphs will result in the predicates of the triples in the RDF graph.

Complexity Management. Complexity management refers to the ability of a visual notation to represent information without overloading the human mind [42]. Graph-based visualizations can be difficult to understand by users when they become too large. Therefore, when the application that implements the visual notation for mapping rules fails to address it, it might overload its users.

Visual Expressiveness. Visual expressiveness is defined as the number of visual variables used in a notation [42]. The number is proportional to the understanding of a notation, as it exploits multiple visual communication channels and maximizes computational offloading. In total, there are eight visual variables: horizontal position, vertical position, size, brightness, color, texture, shape and orientation (see Table 2). Each variable has a power and capacity. The power denotes which type of

information can be used: interval, ordinal, or nominal. The capacity denotes how many perceptible steps are needed to understand the variable.

MapVOWL uses five out of the eight visual variables, as it uses size, brightness, color, texture, and shape (see Section 5.2.1). They are used in accordance to their power and capacity. Size denotes how often a data fraction is used in a data sources, ranging from 0% to 100% (interval), with steps of 10% resulting in 10 perceptible steps (< 20 (capacity in Table 2)). Brightness denotes which graph elements is selected, which is either true or false (ordinal, < 6). Color denotes the different data sources (nominal), and in most cases the number of data sources is limited to less than 5 (< 7). The texture is an alternative to denote a data sources (nominal, < 5). Shape denotes the limited graph elements, and corresponding mapping rules (nominal, unlimited). Even though users are able to adjust the horizontal and vertical position, and orientation, which might improve their understanding, it does not carry a specific meaning in MapVOWL, because MapVOWL does not have a symbol deficit that would benefit from using these variables.

Dual Coding. According to dual coding theory [48], using text and graphics together to convey information is more effective than using either on their own. MapVOWL does not specify the use of dual coding. However, in the RMLEditor it is used. Information about the used data source for an RDF term is available through the color of the nodes and edges (graphics), and when clicking on the details icon (text). Other details about the mapping rules are only available as either text or graphics, because they do not have a corresponding graphical notation or because that would introduce the use of (mapping language) terminology while MapVOWL’s goal is to avoid that (see Section 5.2).

Graphic Economy. Graphic complexity is defined by the number of graphical notations: the size of its visual vocabulary [42]. The principle of graphic economy states that this number should be cognitively manageable. In the case of MapVOWL, the number is low. There are six different graphical primitives (see Section 5.2.1).

Cognitive Fit. Cognitive fit means a visual notation should include different representations to support users with skills ranging from novice to expert, and the representational medium, such as whiteboards,

paper and computer screens [42], which is influenced by the perceptual discriminability, semantic transparency, and visual expressiveness.

The creation of MapVOWL has as main goal to support users who have knowledge about Linked Data and the data domain. Therefore, a different representation for others is not defined. MapVOWL is designed to be displayed on a screen. However, the perceptual discriminability is high enough as MapVOWL relies on differently shaped graphic notation, i.e., circles, rectangles, and lines. The semantic transparency is high enough as MapVOWL uses basic geometric shapes. The visual expressiveness might not be high enough though. For example, MapVOWL only uses colors to denote the different data sources. However, on paper easy-to-draw symbols can be used to compensate this.

6. RMLEditor

The RMLEditor is a graph-based visualization tool for mapping rules that define how Linked Data is generated from multiple heterogeneous data sources [31]. It allows users to load raw data sources, create and edit mapping rules, and preview the resulting RDF triples. Its main goal is to allow users who have domain and Linked Data knowledge, but no knowledge about the mapping language, to create mapping rules (see Figure 3).

In this section, we discuss the GUI requirements for the creation of Linked Data mappings (see Section 6.1), the extended RMLEditor’s architecture (see Section 6.2), the GUI’s design and features (see Section 6.3), and how the mapping rules are executed (see Section 6.4).

6.1. Requirements

In previous work [29], we introduced a list of desired GUI features for the creation of mappings.

R2.1: independent of the underlying mapping language. The visualization of mapping rules should be independent of the underlying language to execute the rules. This is to remove dependencies on the language and to support the editor to switch the underlying mapping language for one that might better suit the use case at hand.

R2.2: support multiple data sources. The GUI should support multiple data sources, as a single Linked Data set might be derived from several.

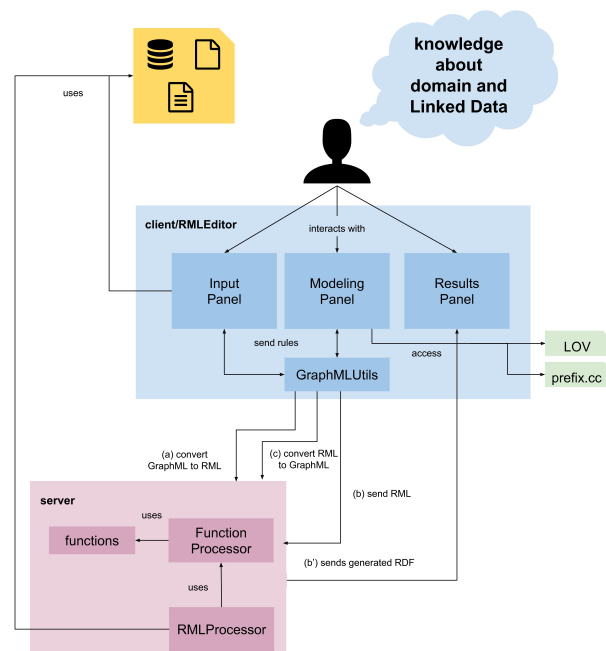


Figure 3: Overview of the RMLEditor

R2.3: support heterogeneous data formats. The original data from which Linked Data is derived might stem from different data formats, such as CSV, JSON, and XML. Thus, the GUI should support the creation of rules on top of different formats. Even more, it should be independent of the format.

R2.4: support multiple ontologies. Different ontologies, which model complementary or overlapping aspects of domain-level knowledge, are available. The GUI should support the creation of a set of rules that uses different ontologies at the same time.

R2.5: support multiple alternative modeling approaches. When users create rules they can follow different modeling approaches [30]. The used approach will be dependent on the user’s preference and use case at hand. The GUI should enable using different approaches to support users and use cases.

R2.6: support non-linear workflows. During the creation of mapping rules different factors are involved, including data, ontologies, vocabularies, mapping rules, and resulting Linked Data. When these factors are presented to the user in isolation, as done with non-linear workflows, the relationships between these factors are obscured. Thus, the GUI should support non-linear workflows to enable users to keep an overview of the different factors.

R2.7: independent of mapping execution. The mapping rules definition and execution are different aspects. An editor and its GUI should not restrict the execution to a specific library or tool. This improves the rules' interoperability and reusability.

6.2. Architecture

The RMLEditor's high-level architecture is based on the multilayered architecture pattern [51]. This allows to separate the presentation of the mapping components, the logic of the mapping process and the access to data and external APIs, using the presentation, application, and data access layer, respectively. The latter only communicates with the application layer. Communication between the presentation and data access layer is not possible, as the architecture prohibits communication between layers not directly under or above each other.

The presentation layer consists of several panels with which users interact. The application layer consists of the modules to process the interactions triggered through the panels. The data access layer handles the requests that require data or mapping rules from outside the RMLEditor. RML [20] is the underlying mapping language of the RMLEditor, because RML supports mapping rules with data from multiple heterogeneous data sources.

The communication with the application layer is facilitated by the command pattern [47] and the flux pattern [7]. The flux pattern replaced the Model-View-Controller (MVC) pattern [47] that was used in the initial version of the RMLEditor. The MVC pattern causes problems in both performance and development, because of the bidirectional communication, where one change can loop back and have cascading effects across the codebase [7]. The flux pattern dictates an unidirectional flow. This flow is stricter than the flow in MVC, because it does not allow to start new flows when another flow is still being executed. For the presentation layer, the Webix JavaScript library¹² is used to build the GUI and the d3.js library [8] to build the graphs.

The graph-based visualizations are encoded using an XML document, based on Graph Markup Language (GraphML) [9] with custom extensions, to represent them independently of the underlying mapping language. This allows users to export the graphs, besides the mapping rules, in an application-independent format. Additionally, the

graphs' GraphML representation is used to generate the corresponding RML statements (see Figure 3a). When users want to load mapping rules into the RMLEditor, they can load (i) a GraphML document, or (ii) an RML mapping document which is converted to GraphML and loaded (Figure 3c). The corresponding graphs are shown in the GUI.

Data transformation descriptions are also encoded in the GraphML representation and thus they are also present in the resulting RML document which is enriched in this case with descriptions based on the Function Ontology for the data transformations [18, 17]. When an RML document is being processed by the extended RMLProcessor server-side, the raw data values are extracted by the RMLProcessor, and data transformations are handled by passing the raw data values and the function which is desired to be applied to the raw data values to the Function Processor.

6.3. Graphical User Interface

In this section, we discuss the RMLEditor's GUI and its interaction elements to manipulate the graph-based visualizations.

6.3.1. Panels

The GUI consists of three views: Input Panel, Modeling Panel and Results Panel (Figures 3, and 4) [31]. They are aligned next to each other, but when users want to focus on a specific panel, they can hide the Input or Results Panel.

The **Input Panel** shows the data sources (the left panel in Figure 4). Multiple data sources can be loaded and they can be in different data formats, such as CSV, JSON, and XML. Each data source is assigned with a unique color to be used in the graphs. In the previous version of the RMLEditor, this panel only shows the raw data, thus, providing a different representation for each data format.

The different representations for each format were circumvented. Now, the Input Panel is divided into two subpanels. The top panel displays the data sources structure. This makes it independent of the data format. The structure can be manipulated if data values need to be transformed. The bottom panel shows the raw data of the source. This allows users to view both the structure and data values.

The **Modeling Panel** shows the mapping rules using MapVOWL (the middle panel in Figure 4). The color of each node and edge depends on the data source that is used in a specific mapping rule,

¹²<http://webix.com/>

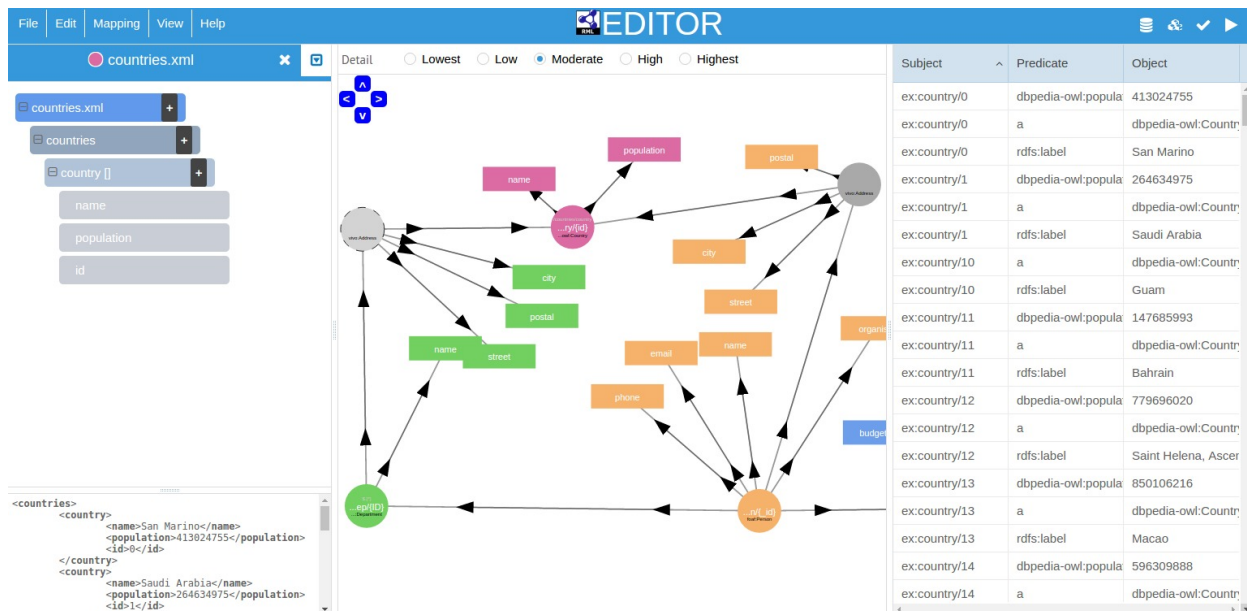


Figure 4: Overview of the RMLEditor

if any. It offers the means to manipulate the nodes and edges of the graphs to update the mapping rules. Semantic annotations can be added using multiple ontologies, which can be either defined locally or online. Linked Open Vocabularies [61] can be consulted via the GUI to get suggestions on which classes, properties and datatypes to use. It is possible to use prefixes instead of using a full IRI. These prefixes can be customized and defined for both local and online ontologies. Users can consult <http://prefix.cc> to search for well-known prefixes and their namespaces.

In the previous version of the RMLEditor, scalability was only addressed using geometric zooming [28], but interactive filtering [55] was introduced to address usability issues with large graphs (more details are available at Section 6.5). Geometric zooming and interactive filtering are two established methods to address large graphs.

The **Results Panel** shows the resulting RDF triples when mapping rules created in the Modeling Panel are executed on the data in the Input Panel (right panel in Figure 4). For each RDF triple, it shows the subject, predicate, and object.

Furthermore, the use of the three panels allows to follow different mapping generation approaches [30, 31]. The data-driven approach uses the input data sources as the basis to construct the mapping rules. Classes, properties, and datatypes








Figure 5: Interaction elements on a node

of the schemas are then assigned to the mapping rules. When users start with the ontologies to generate the mapping rules, the schema-driven approach is followed. Next, data fractions from the data sources can be associated to the mapping rules.

The RMLEditor implements the aforementioned requirements for a GUI for Linked Data mappings. The visualization of the mapping rules is independent of the underlying mapping language through the use of MapVOWL in the Modeling Panel (R2.1). Multiple, heterogeneous data sources are supported via the Input Panel (R2.2 and R2.3). Multiple ontologies are supported via the Modeling Panel (R2.4). The collaboration between the three panels supports multiple alternative modeling approaches (R2.5) and non-linear workflows (R2.6). The independence of mapping execution is implemented because the mapping rules can be exported as both GraphML and mapping documents (R2.7).

Table 3: Interaction Elements

Element	Application
	show/edit details
	create relationship
	delete node/edge
	collapse graph
	expand graph

6.3.2. Interaction

Besides visually understanding the mapping rules, users should also be able to interact with the graphs, which results in updating the corresponding rules. QueryVOWL provides this functionality for queries, but its approach can also be applied to mapping rules. Therefore, similar to QueryVOWL, the RMLEditor uses a set of interaction elements (see Table 3): icons placed on the border of a node or edge (see Figure 5). They correspond with five actions: edit details of a mapping rule, create relationship, delete node/edge, and collapse/expand graph. The different icons were chosen to be simple, but concrete and distinctive to increase the user performance [41]. Every action has a distinctive icon, with exception for collapse and expand graph, but they are never shown at the same time.

These icons are only shown when users hover over the node or edge to reduce the visual overload.

6.4. Mapping Execution

Once the mapping rules are created, they can be executed by the RMLProcessor to generate Linked Data. This might happen also via the RMLEditor. We developed a Web API, using Node.js¹³, to separate this functionality from the RMLEditor (see Figure 3). This makes it also easy to switch between different tools that execute mapping rules that support multiple, heterogeneous data sources.

The API offers three functions: (i) executing a mapping document on a set of data sources (see Figures 3b and 3b’); (ii) converting a GraphML-based document to RML to execute the mapping rules using the RMLProcessor (see Figure 3a), and (iii) converting RML statements to a GraphML-based document to visualize the loaded mapping rules (see

Figure 3c). Another mapping language and corresponding processor can be used, only leading to adjustments to the RMLEditor’s code that converts the visualization to mapping language statements, while no adjustments to the GUI are required.

6.5. Manipulation of Large Graphs

When applying graph-based visualizations, large graphs are inevitable. Users have difficulties editing and keeping an overview of the mapping rules when large-scale graphs are not addressed. Geometric zooming [28] and interactive filtering [55] are two established methods to address large graphs. The former provides enlargement of the graphs. The zoom level can be set through, e.g., scrolling, buttons, or keyboard shortcuts. The latter allows users to filter out the relevant elements of the graphs.

VOWL does not specify any methods on how to deal with large graphs. However, WebVOWL, which implements VOWL, applies both aforementioned methods. Geometric zooming is done through scrolling. Interactive filtering is applied through a filter feature which gives users access to four options that hide specific information or graphical notations and reduce the size of the graph: hide datatype properties, solitary subclasses, information about disjointness, and set operators.

Similar to VOWL, geometric zooming and interactive filtering is not specified by MapVOWL. Geometric zooming is implemented in the RMLEditor via scrolling, as in WebVOWL. Interactive filtering is implemented via the use of detail levels. Interactive filtering requires to define which filters can be applied on the graphs. Instead of letting the users select the filters, as WebVOWL does, we opted to group a set of appropriate filter together in a so-called detail level. The detail levels are arranged from highest to lowest, where each level applies one or more additional filters to the higher level.

Highest Level. The highest level shows all visual elements of MapVOWL. This level is used when users want to inspect and edit all details of the mapping rules, and when the general overview of the mapping rules is less important (see Figure 6).

High Level. The high level hides the literals’ datatypes and languages (see Figure 7). This level focuses more on how different graph elements are related to each other, hiding the specifics, which might be the most redundant for the users.

¹³<https://nodejs.org>

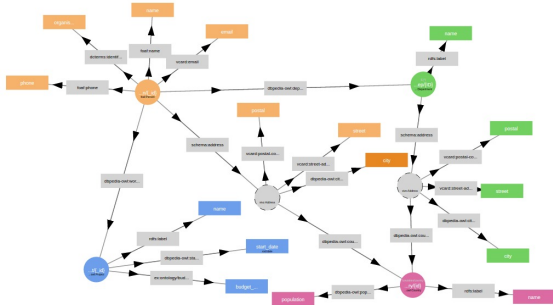


Figure 6: Highest detail level in the RMLEditor

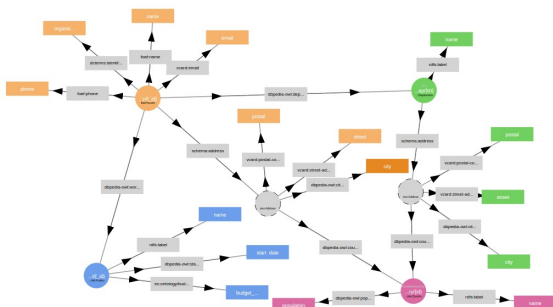


Figure 7: High detail level in the RMLEditor

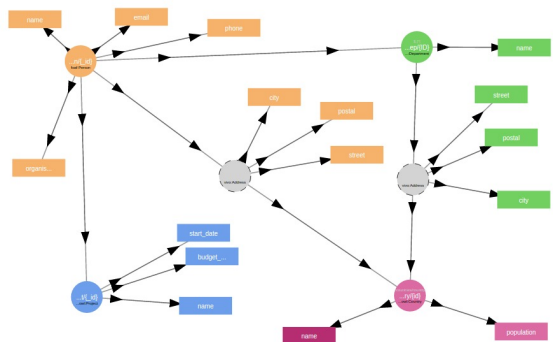


Figure 8: Moderate detail level in the RMLEditor

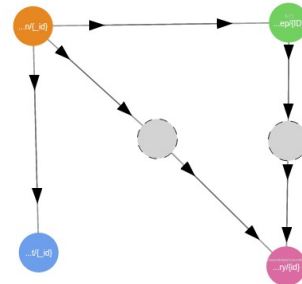


Figure 9: Low detail level in the RMLEditor

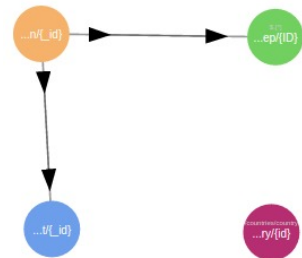


Figure 10: Lowest detail level in the RMLEditor

Moderate Level. The moderate level hides the properties of the edges (see Figure 8). This puts more focus on which subjects and objects are related to each other, instead of how, i.e., only the relationship, without a specific predicate. Furthermore, the graphs become less dense, leading to a better overview of the mappings rules.

Low Level. The low level hides the literals (see Figure 9). This puts more focus on how the different entities are related, and more specific how the different data sources are connected. Furthermore, the graphs become less dense, leading to a better overview of the mappings rules.

Lowest Level. The lowest level hides the blank nodes (see Figure 10). This only shows entities that have an IRI assigned, which allows the users to deal only with entities of the different data sources and how they are directly related, to improve their understanding of the links between the data.

More, when users are at a certain detail level, they might want to show or hide details of a specific subgraph, without the desire to change the overall level. In the RMLEditor, this is accomplished using the collapse and expand actions (see Section 6.3.2).

When a mapping document is loaded in the RMLEditor an appropriate detail level is selected

based on the graphs size, i.e., the amount of nodes is higher than a specific threshold, to provide users with the mapping rules overview. If the level is too high for the size of the graph, users lose the overview of the mapping rules, which makes it difficult to navigate to the part of the graph which is desired to be edited.

6.6. Heterogeneous Data Values Manipulation

Data sources might have heterogeneous data formats. Thus, user interfaces for mapping rules need to support multiple data formats. Moreover, presenting both the data structure and the data values allows users to gain insights in both the data model, and corresponding data. Transforming one or more data fractions when generating Linked Data might also be required. For example, if a data source has a data fraction that contains a date in the format “DD/MM/YYYY”. A transformation is needed to convert it to “YYYY-MM-DD” when the datatype `xsd:date` is used for the corresponding literal.

In the RMLEditor, the support for manipulating heterogeneous data values and data sources is facilitated by the Input Panel (see Figure 4). It features both the structure of the data, and the raw data. The structure includes data fractions that users may use to generate the desired Linked Data and can be updated to reflect also the required data transformations. This shows that the transformed data values can be used as any other data value.

7. Evaluation

We conducted two comparative studies to validate our two hypotheses (see Section 4): one study compares the representation of mapping rules via MapVOWL and via RML directly, and another study compares the creation of mapping rules via the RMLEditor and the RMLx Visual Editor.

7.1. MapVOWL vs. RML

With this study, we aim to validate H1. We compared what knowledge users are able to extract from both a set of mapping rules that are represented via MapVOWL or RML directly. This knowledge aligns directly with the six requirements for a visual notation (see Section 5.1). Furthermore, we evaluated which representation they prefer. In Section 7.1.1, we discuss the method, namely the procedure and participants. In Section 7.1.2, we discuss the results, which can also be found

Table 4: Blocks of the MapVOWL vs. RML questionnaire

block	description
introduction	Questions about participants’ socio-demographics & Linked Data expertise
MapVOWL vs. RML	MapVOWL-specific questions followed by four test cases to compare MapVOWL & RML
post-assessment	Questions on preference of MapVOWL vs RML and the use of MapVOWL for editing mapping rules

at <https://w3id.org/mapvowl/eval17/results>, and, in Section 7.1.3, the corresponding derived insights.

7.1.1. Method

Procedure. Participants with RML knowledge were directly contacted by the author. Those who agreed to partake in the test had to read an introduction to MapVOWL¹⁴ and complete an online questionnaire. The introduction explains the different elements of MapVOWL through the use of an example which was provided to assure that participants have a basic understanding of MapVOWL, too. The participants completed an online questionnaire, which can be found at <https://w3id.org/mapvowl/eval17/survey>. The questionnaire consists of three main building blocks (see Table 4):

(i) The first block questioned the participants’ main **sociodemographic traits**, such as year of birth, gender, level of education, and employment status. Then, it measured the participants’ **Linked Data expertise** through a self-assessment and a study of their familiarity with the topic and tools.

(ii) The second and essential block provides a **test with eleven questions** about mapping rules that were presented as a MapVOWL visualization to access the understanding of the MapVOWL elements and *ten questions about four test cases* to compare a MapVOWL representation with an RML representation. Three of the use cases are real-world examples and one is artificially created for the study.

Half of the participants answered the questions of the first two use cases via the MapVOWL representation and the last two via RML. The other half of the participants answered the questions of the first

¹⁴<https://w3id.org/mapvowl/eval17/intro>

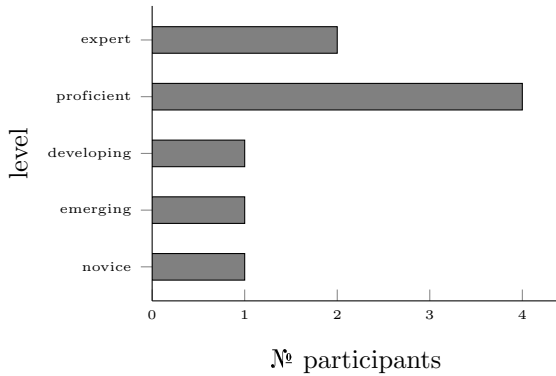


Figure 11: Level of Linked Data expertise of participants of MapVOWL user study

two use cases via the RML representation and the last two via MapVOWL. Participants were randomly assigned to one of the two aforementioned groups.

(iii) The **post-assessment** consists of three questions and gathers information about the participants’ preference regarding the use of MapVOWL versus RML to answer the questions, and whether they want to use MapVOWL to also edit mapping rules, besides only to visualize them.

Participants. The online questionnaire was sent out to potential participants in August 2017. Nine participants have eventually taken part in the experiment, their age range was 20 to 38. All were highly educated: three of the contributors had a PhD, four a master’s degree and two a bachelor’s degree. They were highly familiar with Linked Data as well: 6 of the participants claimed they already generated Linked Data and only one declared to have merely a basic understanding of Linked Data (see Figure 11).

7.1.2. Results

Mapping Rules. When answering the eleven questions about the elements of MapVOWL, four questions are answered correctly by all participants: determining the number of literals per entity and relationships, the used ontology and language. Two questions are answered correctly by 8 participants: detecting entities without a class and determining how IRIs are generated. Two questions are answered correctly by 7 participants: detecting the number of literals and their datatypes. Two questions are answered correctly by 6 participants: determining the number of distinct data sources and the number of entities that are not associated with

Table 5: № correct answers regarding MapVOWL

questions	correct answers (max. 9)
№ literals per entity/relationships used ontology/language	9
entities without a class how IRIs are generated	8
№ literals and their datatypes datatypes of literals	7
№ distinct data sources	6
№ entities w/ data source	6
№ relationships ind. of data source	5

Table 6: № correct answers when using MapVOWL and RML

questions	correct answers	
	RML	MapVOWL
types of entities		
№ literals	51	61
relationships b/t entities		
№ (unlinked) data sources	65	50
the language of literals		
use of templates	24	24
use of datatypes		

a data source. One question is answered correctly by 5 participants: determining the number of relationships that do not depend on a data source. This is summarized in Table 5.

Regarding the questions about the use cases, the number of correctly answered questions is slightly higher for RML than MapVOWL (140 correct answers vs. 135), as summarized in Table 6. Questions regarding the types of entities, number of literals, and relationships between entities are answered more often correctly via MapVOWL than RML (61 correct answers vs. 51). Questions regarding the number of (unlinked) data sources and the language of literals are answered more often correctly via RML than MapVOWL (65 correct answers vs. 50). Questions regarding the use of templates and datatypes are answered correctly as often via MapVOWL as via RML (24 correct answers).

Post-Assessment. 7 participants preferred the application of the MapVOWL notation for *viewing* the presented cases. 1 participant preferred using RML directly over MapVOWL and 1 was neutral (see Figure 12). Similarly, when they were asked which they prefer to use for *editing* mapping rules, 7 partici-

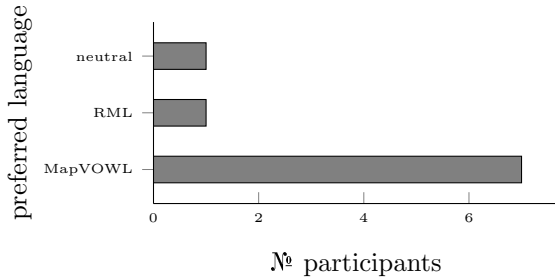


Figure 12: Preferred language for viewing the rules

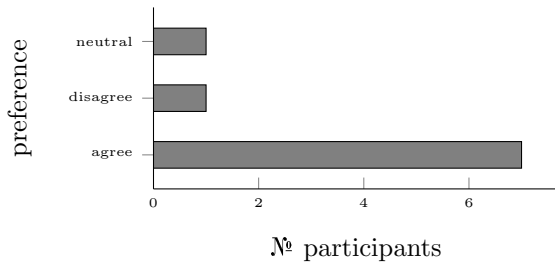


Figure 13: Participants' agreement to use MapVOWL to edit the rules

pants agreed that they prefer MapVOWL, 1 slightly disagreed and 1 remained neutral (see Figure 13).

7.1.3. Insights

The answers to MapVOWL-specific questions provide evidence that **the visual notation is cognitive effective**. When comparing the answers of questions that are both answered using MapVOWL and RML, it provides evidence that both options are possible candidates to represent mapping rules. However, the post-assessment shows that **users prefer MapVOWL** over RML to visualize mapping rules, even though RML results in a slightly higher number of correct answers.

7.2. RMLEditor vs. RMLx Visual Editor

With this study, we aim to validate H2 by comparing the use of the RMLEditor and RMLx Visual Editor to edit mapping rules. Specifically, the visualization of the mapping process components with special attention to the rules is studied. The RMLx Visual Editor was chosen, as it allows to annotate multiple, heterogeneous data sources and values. More, it also uses graph visualizations to represent the mapping rules; however a form-based approach is used to edit the rules.

Table 7: Steps of the RMLEditor vs. RMLx Visual Editor user study

step	description
introduction	Questions on participants' socio-demographics & Linked Data expertise
use cases	2 use cases per participant on mapping rules creation
additional RMLEditor test	Questions on heterogeneous data sources & data values, & RMLEditor's large graphs
post-assessment	Questions on participants' experience with the used tool

In Section 7.2.1, we discuss the participants and procedure. In Section 7.2.2, we discuss the results, and in Section 7.2.3, the derived insights.

7.2.1. Method

Procedure. This second study is modeled according to the following procedure¹⁵¹⁶. Potential participants from imec and Ghent University were approached. Those agreeing to contribute to the user evaluations then had to read an introduction¹⁷ to the RML and the RMLx Visual Editor and work through the different steps of the study together with one of the authors. More details about the experimental setting of the latter can be found at <https://w3id.org/rml/editor/eval17/setting>. The introduction explains the different elements of both tools. For the RMLEditor, it also includes the introduction to MapVOWL, because it implements MapVOWL. The user study is divided into four steps (see Table 7):

(i) The participants are presented with questions about their **socio-demographics** and Linked Data **expertise**, identical to ones asked to participants of the MapVOWL vs. RML study (Section 7.1).

(ii) Two **use cases** are presented to the users for which mapping rules need to be created. The first use case deals with data about employees and the projects they are working on. Both data sources are provided as a CSV file. This use case is *data-driven*: every data fraction should be represented in the resulting Linked Data. The second use case deals with data about movies and their directors.

¹⁵<https://w3id.org/rml/editor/eval17/survey1>

¹⁶<https://w3id.org/rml/editor/eval17/survey2>

¹⁷<https://w3id.org/rml/editor/eval17/intro>

The movie data source is provided as a CSV file and the director data as a JSON file. This use case is *schema-driven*: based on a given set of classes, properties, and datatypes Linked Data needs to be generated using the provided raw data.

Half of the participants completed the first use case with the RMLEditor and the second use case with the RMLx Visual Editor. The other half completed the first use case with the RMLx Visual Editor and the second use case with the RMLEditor. Participants were randomly assigned to one of the two groups. The outcomes were assessed based on the generated Linked Data correctness and the experience with each tool was recorded.

(iii) An **additional test** was completed for the RMLEditor. Here, the participant got presented with an online survey, containing specific questions regarding the heterogeneous data sources and values, and large graphs. During the second and third step of the user study, each participant was supervised by one of the authors making use of the think-aloud protocol. Widely used by usability professionals, think-aloud originated out of the incapability to know what users think when completing tasks [46]. In the concurrent think-aloud process as implemented in the current usability study, users are probed to specify their thoughts and actions as they occur. This allows the attending author to discern when and why a participant faces difficulties.

(iv) After each use case, a **post-assessment** gathers information about the participants' experience with the used tool. Included are questions about the perceived difficulty of the tasks and confidence in a successful completion. A central part of the post-assessment is assigned to the System Usability Scale (SUS) [10], a procedure to quickly collect users' usability ratings of a technology. Benefits of applying SUS are its conciseness, its transferability over several technologies, and applicability with small sample sizes [1, 59]. As proposed by Bangor et al. [1], an adjective rating using a 7-point Likert scale, which assesses the tools user-friendliness from worst imaginable to best imaginable, was also added to the questionnaire. It should be sharply noted though that the outcome of the adjective rating and the system usability scale should be supported by observations by the attending supervisor.

Participants. The participants of our study were people from imec and Ghent University, that were contacted directly by the authors. This ensured that they had a sufficient level of Linked Data ex-

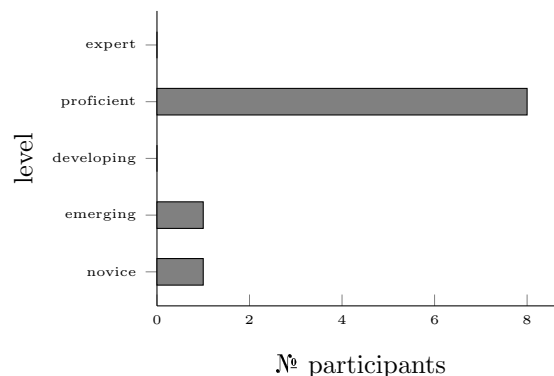


Figure 14: Level of Linked Data expertise of participants of RMLEditor vs. RMLx Visual Editor user study

Table 8: Completeness of rules and SUS-score of RMLEditor and RMLx Visual Editor

	RMLEditor	RMLx Visual Editor
completeness of rules (%)		
use case 1	91	83
use case 2	98	82
SUS-score		
	82.75 (good)	42 (poor)

pertise, required for a meaningful participation in the user study. A total of 10 participants, age range 27 to 39, were recruited. Their level of education was high: they all hold a master's degree, have conducted advanced graduate work, or hold a PhD. 8 participants assessed themselves as having at least a proficient level of Linked Data expertise, 1 considered himself to be a novice and 1 observed his Linked Data expertise as emerging (see Figure 14). They all had heard of the RMLEditor before, 2 participants had used it; whereas 6 participants had taken notice of the RMLx Visual Editor before.

7.2.2. Results

Mapping Rules. An analysis of the created mapping rules by the participants, which can be found at <https://w3id.org/rml/editor/eval17/results/mappings>, revealed several aspects that can be circumvented by updating the GUI of the tools. For the first use case 91% of the expected mapping rules were present when using the RMLEditor. In the case of the RMLx Visual Editor, it was only 83%. The same is observed for the second use case with 98% for the RMLEditor and 82% for the RMLx Visual Editor (see Table 8).

In more details, the iterator was forgotten by 3 participants using the RMLx Visual Editor, but this was not an issue with the RMLEditor. The use of constants, reference, and templates was incorrect for 2 participants using the RMLx Visual Editor, but this was not an issue with the RMLEditor.

All participants added the data transformation with both tools, but 4 participants forgot to use the output of the transformation as an object using the RMLx Visual Editor, while only 2 using the RMLEditor. This led to triples that did not contain the transformed data value, but the original value. When interlinking datasets, all but one participant created the required mapping rule using the RMLx Visual Editor. Using the RMLEditor, all participants created the mapping rule, but 3 participants did not provide the required join condition which interlinks entities that are not related. This was not an issue using the RMLx Visual Editor.

Furthermore, we pay special attention to new features of the RMLEditor introduced in this work. Users are able to distinguish the different data sources, but when they have to count the total number of data sources that are loaded 4 participants forget to count the currently selected data source. 9 participants were able to correctly determine how many data transformations are applied when presented with a new set of mapping rules. They were also able and perceived it as easy to determine which data transformation and data fractions are used. When users create and edit mapping rules that link entities in large graph visualizations, 7 of them use the lower detail levels. When users create and edit mapping rules that generate literals the preferred detail levels varies from the highest (2 participants) to the lowest level (1 participant), and 2 participants even stated no preference.

Post-Assessment. The first part of the post-assessment questioned the participants with regard to the **difficulty** of performing the use case with the tools and their confidence in a successful completion of the tasks. The results of the post-assessment can be found at <https://w3id.org/rml/editor/eval17/results/post>. The difficulty of performing the use cases was rated on a 7-point likert scale from extremely difficult to extremely easy.

Regarding *difficulty*, 1 participant rated completing the tasks with the RMLEditor as “slightly difficult”, all others assessed it as “neither easy nor difficult” or higher, with 3 participants valuating it at “extremely easy” (see Figure 15). For the RMLx

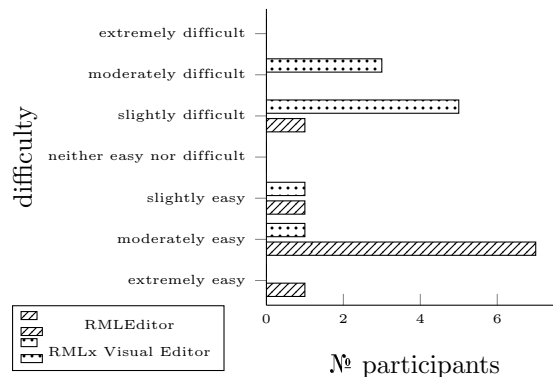


Figure 15: Difficulty of the RMLEditor vs. RMLx Visual Editor

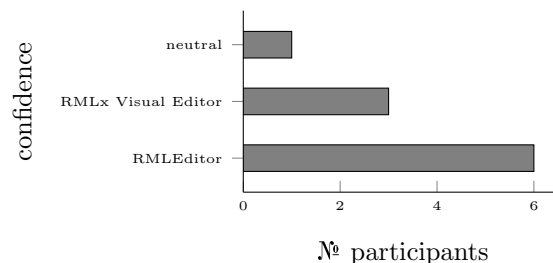


Figure 16: Confidence of using the RMLEditor vs. RMLx Visual Editor

Visual Editor, all participants except 1 consistently rated executing the tasks more difficult than with the RMLEditor. 5 participants scored it as “slightly difficult” and the other 3 as “moderately difficult”.

The *confidence* for having correctly executed the tasks was positively rated for both tools. 6 participants were more confident in performing the tasks with the RMLEditor, 1 rated both tools equal and 3 were more confident with the RMLx Visual Editor (see Figure 16).

The second part of the post-assessment applied the SUS and the accompanying **user-friendliness** rating proposed by Bangor et al. [1]. The user-friendliness of the RMLEditor was rated on a 7-point likert scale. 9 participants scored it as excellent, the other 1 rated it as good. In contrast, the user-friendliness of the RMLx Visual Editor was valuated as “good” by 1 respondent, “ok” by 1 respondent, “poor” by 6 respondents and “awful” by the remaining 2 (see Figure 17).

More, all except 1, who rated both as good, assessed the user-friendliness of the RMLEditor higher than that of the RMLx Visual Editor. The average obtained mean SUS-score of the RMLEditor

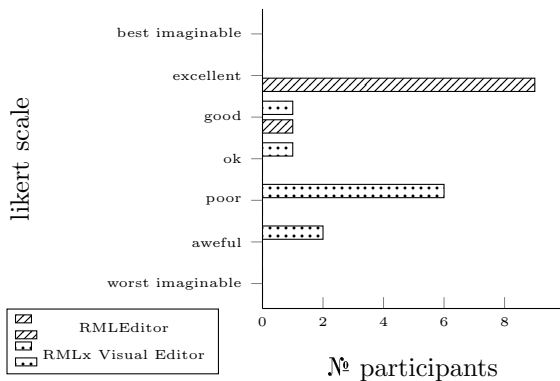


Figure 17: User-friendliness of the RMLEditor vs. RMLx Visual Editor

is 82.75, whereas for the RMLx Visual Editor, it is 42 (see Table 8). When translating these scores on the spectrum created by Bangor et al. [1], the usability of the RMLEditor is identified as good, but that of the RMLx Visual Editor as poor.

Two remarks: (i) a single metric should not be used to make absolute statements about a systems’ usability [1]. The observations done by one of the authors and elaborated in the next paragraph give an enhanced understanding of the process of utilizing both tools; (ii) one of the authors was present during the user study, which could result in a moderator acceptance bias. This happens when participants want to gratify the present author.

Observations. During the study, participants were supervised by one of the authors. The results can be found at <https://w3id.org/rml/editor/eval17/results/obs>. Participants could ask questions and receive feedback while using both tools. Although, users were provided with an introduction to both tools, which they could consult during the study, they still asked questions about how certain actions should be performed.

The participants referred to the *introduction* to find the required information. This happened for 8 participants when using the RMLEditor and for 9 participants when using the RMLx Visual Editor.

The RMLx Visual Editor does not allow to visualize the data sources. Therefore, every participant used text editors and spreadsheet tools to view the data sources. In the case of the RMLEditor only 1 participant first viewed the data sources externally before loading them in the RMLEditor.

Participants required additional information about the use of *parameters* to create a data trans-

formation. This happened for 3 participants using the RMLx Visual Editor and 6 participants using the RMLEditor. The difference occurs because the default parameter was already set in the RMLx Visual Editor, whereas no default parameter is set for the RMLEditor. Nevertheless, still 3 participants have trouble understanding these parameters as shown via the RMLx Visual Editor.

5 participants required additional information about the use of a *baseURI* and how it is defined in the RMLEditor. Although it is perceived as useful, the participants indicated that a clearer presentation in the GUI is required. The RMLx Visual Editor does not offer this functionality.

A common issue with both tools is the use of constants, references, and templates to generate subjects, predicates, and objects. 9 participants for both tools required additional information to understand how they work and when they should be used. Although the RMLEditor selects a default depending on the element in the graph, additional information was still required for the users to fully understand these three options.

8 participants required additional information to understand how to *interlink* entities originating from different data sources using the RMLx Visual Editor. “Parent mapping”, “parent value”, and “child value” were terms difficult to be understood, and participants had trouble understanding how they would affect the resulting Linked Data. This was not an issue during the use of the RMLEditor, because of the use of MapVOWL.

2 participants required information to understand that an *iterator* for a CSV file is not required when using the RMLx Visual Editor. This was not an issue using the RMLEditor, because an iterator cannot be set when using a CSV file as data source, considering the iteration always happens per row.

3 participants required information about the use of the “output var”-field in the RMLx Visual Editor. This field represents the output of a data transformation and the string can be used in other mapping rules to use the output of a data transformation. This was not an issue using the RMLEditor, because data transformations are integrated in the original data fractions in the input panel.

7.2.3. Insights

The resulting mapping rules provide evidence that the use of the RMLEditor leads to a **higher completeness** of rules, and thus of Linked Data, compared to the RMLx Visual Editor. For both

tools updates to the GUI could improve this completeness, such as the inclusion of checks for the use of transformed values and iterators. The RMLEditor was rated as good and the RMLx Visual Editor as poor by the participants. This is in line with the completeness of the rules.

The observations during the user study made clear that specific **terminology**, such as baseURI, reference, and template, needs further **clarification**. This could be done, for example, through tool tips in the GUI or by avoiding the use of the terminology. The observations provided evidence for the use of the RMLEditor over the RMLx Visual Editor when interlinking entities due to the use of the visual notation in the former and the use of forms and specific terminology in the latter, as explicitly mentioned by the participants.

8. Conclusion

Visual tools are implemented to help users in defining how to generate Linked Data from raw data. This is possible thanks to mapping languages which enable detaching mapping rules from the implementation that executes them. However, no thorough research has been conducted so far on how to visualize such mapping rules, especially if they become large and require considering multiple heterogeneous data sources and transformed data values. In this article we introduced MapVOWL, a graph-based visual notation for mapping rules; and an approach for manipulating rules when large visualizations emerge; an approach to uniformly visualize data fractions of raw data sources combined with an interactive interface for uniform data fraction transformations. MapVOWL and the two approaches were implemented in the RMLEditor to provide users with a uniform GUI to create and edit mapping rules that is cognitive effective.

The results of the first study (Section 7.1) show that the use of MapVOWL is preferred over RML for representing mapping rules. Answering questions about these rules leads to the same correctness (accuracy in cognitive effectiveness) by either using MapVOWL or RML for users who understand RML. Nevertheless, most participants indicated that they preferred to use MapVOWL as representation for mapping rules, and they would also use MapVOWL to create and edit rules. This shows that MapVOWL is easier to use than RML directly. Therefore, this study provides evidence towards the acceptance of H1 “MapVOWL improves the cognitive effectiveness

of the mapping rules graph-based visual representation to generate the Linked Data compared to using a mapping language directly.”

The results of the second study (Section 7.2) show that the RMLEditor is preferred over the RMLx Visual Editor for creating and editing mapping rules. Participants were able to create a high percentage of the required mapping rules using both tools (high accuracy). The RMLEditor has a better usability compared to the RMLx Visual Editor, as shown through the SUS-score (82.75 vs 42), making it easier for users to create and edit rules. This is due to the use of MapVOWL, which is independent of the underlying mapping language, as pointed out by the participants. However, through the use of think-aloud, we conclude that updates to both GUIs would improve the mapping process, with specific focus on the creation of the mapping rules. The RMLx Visual Editor’s form-based GUI is strongly influenced by its underlying mapping language RML, which leads to issues for users that are unfamiliar with RML. These issues can only be resolved either by acquiring knowledge about RML or avoiding to rely on knowledge of RML, as done by the RMLEditor. Therefore, this study provides evidence towards the acceptance of H2 “The cognitive effectiveness provided by the RMLEditor’s GUI improves the user’s performance during the Linked Data generation process – that is based on legacy non-RDF and (semi-)structured data sources – compared to the state of the art.”

MapVOWL fills the important gap between uniform mapping rule visualizations and the different mapping editors. It allows multiple tools to incorporate the same visualization, which increases the accessibility for users across these tools. The different semantic constructs of the mapping rules align with the graphical notations of the graph-based visualization. This leads to an effective representation for users. Furthermore, it combines the different components of the mapping process, providing users with insights on the interrelation of these components. Users are able to define and deduct which data fractions of which data sources are used in which mapping rules, and deduct how the mapping rules correspond with the generated Linked Data.

Moreover, MapVOWL and the RMLEditor overcome two issues of current mapping editors, namely, the uniform presentation of heterogeneous data sources and data transformations within Linked Data generation. Multiple data sources are supported and color is an appropriate method to de-

note them. Distinguishing the data fractions that users rely on to generate Linked Data from the original raw data allows to create an abstraction layer. Such an abstraction layer not only enables to easily manipulate the data fractions but also incorporates data transformations uniformly for heterogeneous data sources. This allows users to handle transformed data values just as raw values.

Large graphs might result in a reduction of the graph-based visualizations' efficiency. However, the second study shows that detail levels are appropriate to compensate this for mapping rules. It is efficient for the users to change the detail levels based on the task at hand, and personal preference.

Overall, if users need to create and edit mapping rules, the RMLEditor is a valid choice through the use of the proposed MapVOWL and the approaches to deal with large visualizations and heterogeneous data sources and values. Although, improvements to the GUI are required, which will be addressed in future work, the GUI's core elements have been proven to be effective via the two user studies.

Acknowledgements

The described research activities were funded by Ghent University, imec, Flanders Innovation & Entrepreneurship (AIO), the Research Foundation – Flanders (FWO), and the European Union.

References

- [1] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3): 114–123, 2009.
- [2] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Biomedical Informatics*, 41(5):706–716, 2008.
- [3] Nikos Bikakis, Melina Skourla, and George Papastefanatos. rdf:SynopsViz – A Framework for Hierarchical Linked Data Visual Exploration and Analysis. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 292–297. Springer International Publishing, 2014. ISBN 978-3-319-11955-7. doi: 10.1007/978-3-319-11955-7_37.
- [4] Nikos Bikakis, George Papastefanatos, Melina Skourla, and Timos Sellis. A Hierarchical Aggregation Framework for Efficient Multilevel Visual Exploration and Analysis. *Semantic Web*, 8(1):139–179, 2017.
- [5] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, pages 205–227, 2009. doi: 10.4018/978-1-60960-593-3.ch008.
- [6] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009. ISSN 1570-8268. doi: <http://dx.doi.org/10.1016/j.websem.2009.07.002>. URL <http://www.sciencedirect.com/science/article/pii/S1570826809000225>.
- [7] Adam Boduch. *Flux Architecture*. Packt Publishing, 2016.
- [8] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D³ Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.
- [9] Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. GraphML Progress Report Structural Layer Proposal. In *International Symposium on Graph Drawing*, pages 501–512. Springer Berlin Heidelberg, 2001.
- [10] John Brooke. SUS – A quick and dirty usability scale. *Usability Evaluation in Industry*, pages 189–194, 1996.
- [11] Karlis Cerans, Julija Ovcinnikova, Renars Liepins, and Arturs Sprogis. Advanced OWL 2.0 Ontology Visualization in OWLGrEd. In *Databases and Information Systems VII*, pages 41–54. IOS Press, 2013. doi: 10.3233/978-1-61499-161-8-41.
- [12] The UniProt Consortium. Uniprot: a hub for protein information. *Nucleic Acids Research*, 43(D1):D204, 2015. doi: 10.1093/nar/gku989.
- [13] World Wide Web Consortium. RDF 1.1 Concepts and Abstract Syntax. Technical report, 2014. URL <https://www.w3.org/TR/rdf11-concepts/>.
- [14] Aba-Sah Dadzie and Emmanuel Pietriga. Visualisation of Linked Data – Reprise. *Semantic Web*, 8(1):1–21, 2017. doi: 10.3233/SW-160249.
- [15] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2): 89–124, 2011.
- [16] Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF Mapping Language. Working group recommendation, W3C, September 2012. URL <http://www.w3.org/TR/r2rml/>.
- [17] Ben De Meester and Anastasia Dimou. The Function Ontology. Unofficial Draft, October 2016. <http://users.ugent.be/~bjdmeest/function/>.
- [18] Ben De Meester, Anastasia Dimou, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. An ontology to semantically declare and describe functions. In Harald Sack, Giuseppe Rizzo, Nadine Steinmetz, Dunja Mladenčić, Sören Auer, and Christoph Lange, editors, *The Semantic Web; ESWC 2016 Satellite Events*, volume 9989 of *Lecture Notes in Computer Science*, pages 46–49. Springer International Publishing, October 2016. doi: 10.1007/978-3-319-47602-5_10.
- [19] Laurens De Vocht, Selver Softic, Ruben Verborgh, Erik Mannens, and Martin Ebner. ResXplorer: Revealing relations between resources for researchers in the Web of Data. *Computer Science and Information Systems*, 14(1):25–50, 2017.
- [20] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In *Workshop on Linked Data on the Web*, 2014.

- [21] Martin Džbor, Enrico Motta, Carlos Buil, Jose Gomez, Olaf Goerlitz, and Holger Lewen. Developing ontologies in owl: An observational study. In *OWL: Experiences and Directions 2006*, November 2006.
- [22] Syeda Sana e Zainab, Muhammad Saleem, Qaiser Mehmood, Durre Zehra, Stefan Decker, and Ali Hasnain. FedViz: A Visual Interface for SPARQL Queries Formulation and Execution. In *Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data*, pages 49–60, 2015.
- [23] Ivan Ermilov, Jens Lehmann, Michael Martin, and Sören Auer. LODStats: The Data Web Census Dataset. In *Proceedings of 15th International Semantic Web Conference - Resources Track (ISWC'2016)*, 2016.
- [24] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-38721-0.
- [25] Riccardo Falco, Aldo Gangemi, Silvio Peroni, David Shotton, and Fabio Vitali. Modelling OWL Ontologies with Graffoo. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 320–325. Springer International Publishing, 2014. doi: 10.1007/978-3-319-11955-7_42.
- [26] Florian Haag, Steffen Lohmann, Stephan Siek, and Thomas Ertl. QueryVOWL: Visual Composition of SPARQL Queries. In *The Semantic Web: ESWC 2015 Satellite Events*, pages 62–66. Springer International Publishing, 2015. doi: 10.1007/978-3-319-25639-9_12.
- [27] Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. RelFinder: Revealing relationships in RDF knowledge bases. In *Semantic Multimedia*, pages 182–187. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-10543-2_21.
- [28] Ivan Herman, Guy Melançon, and M Scott Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000. doi: 10.1109/2945.841119.
- [29] Pieter Heyvaert, Anastasia Dimou, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Towards a Uniform User Interface for Editing Mapping Definitions. In *Proceedings of the 4th International Workshop on Intelligent Exploration of Semantic Data (IESD 2015)*. CEUR-WS.org, 2015.
- [30] Pieter Heyvaert, Anastasia Dimou, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Towards Approaches for Generating RDF Mapping Definitions. In *Proceedings of the ISWC 2015 Posters & Demonstrations Track*. CEUR Workshop Proceedings, 2015.
- [31] Pieter Heyvaert, Anastasia Dimou, Aron-Levi Herregodts, Verborgh Ruben, Schuurman Dimitri, Mannens Erik, and Van de Walle Rik. RMLEditor: A Graph-Based Mapping Editor for Linked Data Mappings. In *The Semantic Web. Latest Advances and New Domains*, pages 709–723. Springer International Publishing, 2016. doi: 10.1007/978-3-319-34129-3_43.
- [32] Frederik Hogenboom, Viorel Milea, Flavius Frasin-car, and Uzay Kaymak. RDF-GL: A SPARQL-Based Graphical Query Language for RDF. In *Emergent Web Intelligence: Advanced Information Retrieval*, pages 87–116. Springer London, 2010. doi: 10.1007/978-1-84996-074-8_4.
- [33] Matthew Horridge. OWLViz, 2010. URL <http://protegewiki.stanford.edu/wiki/OWLViz>.
- [34] Ajaz Hussain, Khalid Latif, A Rextin, Amir Hayat, and Masoon Alam. Scalable Visualization of Semantic Nets using Power-Law Graphs. *Applied Mathematics & Information Sciences*, 8(1):355–367, 2014.
- [35] Muhammad Javed, Sandy Payette, Jim Blake, and Tim Worrall. VIZ-VIVO: Towards Visualizations-driven Linked Data Navigation. In *Proceedings of the Second International Workshop on Visualization and Interaction for Ontologies and Linked Data*, page 80, 2016.
- [36] Sergey Krivov, Richard Williams, and Ferdinando Villa. GrOWL: A tool for visualization and editing of OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):54–57, 2007. ISSN 1570-8268. doi: <http://dx.doi.org/10.1016/j.websem.2007.03.005>.
- [37] Jill H Larkin and Herbert A Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive science*, 11(1):65–100, 1987.
- [38] Steffen Lohmann, Vincent Link, Eduard Marbach, and Stefan Negru. WebVOWL: Web-based Visualization of Ontologies. In *Knowledge Engineering and Knowledge Management*, pages 154–158. Springer, 2014.
- [39] Steffen Lohmann, Stefan Negru, and David Bold. The ProtégéVOWL Plugin: Ontology Visualization for Everyone. In *The Semantic Web: ESWC 2014 Satellite Events*, pages 395–400. Springer International Publishing, 2014. doi: 10.1007/978-3-319-11955-7_55.
- [40] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. Visualizing ontologies with VOWL. *Semantic Web*, 7(4):399–419, 2016.
- [41] Siné JP McDougall, Oscar de Bruijn, and Martin B Curry. Exploring the effects of icon characteristics on user performance: the role of icon concreteness, complexity, and distinctiveness. *Journal of Experimental Psychology: Applied*, 6(4):291, 2000.
- [42] Daniel Moody. The “physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009. doi: 10.1109/TSE.2009.67.
- [43] Enrico Motta, Silvio Peroni, José Manuel Gómez-Pérez, Mathieu d’Aquin, and Ning Li. Visualizing and navigating ontologies with KC-Viz. In *Ontology Engineering in a Networked World*, pages 343–362. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-24794-1_16.
- [44] Stefan Negru and Steffen Lohmann. A Visual Notation for the Integrated Representation of OWL Ontologies. In *Proceedings of the 9th International Conference on Web Information Systems and Technologies (WEBIST '13)*, pages 308–315. SciTePress, 2013.
- [45] Natalya F Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W Ferguson, and Mark A Musen. Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001. doi: 10.1109/5254.920601.
- [46] Erica L Olmsted-Hawala, Elizabeth D Murphy, Sam Hawala, and Kathleen T Ashenfelter. Think-aloud protocols: a comparison of three think-aloud protocols for use in testing data-dissemination web sites for usability. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2381–2390. ACM, 2010.
- [47] Addy Osmani. *Learning JavaScript Design Patterns*. O’Reilly Media, 2012.
- [48] Allan Paivio. *Mental Representations: A Dual Coding Approach*. Oxford University Press, 1990.

- [49] Emmanuel Pietriga. Semantic web data visualization with graph style sheets. In *Proceedings of the 2006 ACM Symposium on Software Visualization*, SoftVis '06, pages 177–178, New York, NY, USA, 2006. ACM. ISBN 1-59593-464-2. doi: 10.1145/1148493.1148532.
- [50] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. Technical report. URL <http://www.w3.org/TR/rdf-sparql-query/>.
- [51] Mark Richards. *Software Architecture Patterns*. O'Reilly Media, 2015.
- [52] Alistair Russell, Paul R. Smart, Dave Braines, and Nigel R. Shadbolt. NITELIGHT: A Graphical Tool for Semantic Query Construction. In *Proceedings of the Fifth International Workshop on Semantic Web User Interaction (SWUI 2008)*, 2008.
- [53] Simon Scheider, Auriol Degbelo, Rob Lemmens, Corné van Elzakker, Peter Zimmerhof, Nemanja Kostic, Jim Jones, and Gautam Banhatti. Exploratory querying of SPARQL endpoints in space and time. *Semantic Web*, 8(1):65–86, 2017.
- [54] Kunal Sengupta, Peter Haase, Michael Schmidt, and Pascal Hitzler. Editing R2RML Mappings Made Easy. In *Proceedings of the 2013th International Conference on Posters & Demonstrations Track*, pages 101–104. CEUR-WS.org, 2013.
- [55] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996. doi: 10.1109/VL.1996.545307.
- [56] Álvaro Sicilia, German Nemirovski, and Andreas Nolle. Map-On: A web-based editor for visual ontology mapping. *Semantic Web*, (Preprint):1–12, 2016.
- [57] Alexandra Similea, Niklas Petersen, Christoph Lange, and Steffen Lohmann. TurtleEditor 2.0: A Synchronized Visual and Text Editor for RDF Graphs. In *IEEE 11th International Conference on Semantic Computing*, 2017.
- [58] Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. LinkedGeoData: A Core for a Web of Spatial Open Data. *Semantic Web Journal*, 3(4):333–354, 2012.
- [59] Thomas S Tullis and Jacqueline N Stetson. A comparison of questionnaires for assessing website usability. In *Usability professional association conference*, pages 1–12, 2004.
- [60] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. Sig.ma: Live views on the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):355–364, 2010. ISSN 1570-8268. doi: <http://dx.doi.org/10.1016/j.websem.2010.08.003>.
- [61] Pierre-Yves Vandenbussche, Ghislain A Ateazing, María Poveda-Villalón, and Bernard Vatant. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web*, 8(3): 437–452, 2017.
- [62] Marc Weise, Steffen Lohmann, and Florian Haag. LD-VOWL: Extracting and Visualizing Schema Information for Linked Data. *Proceedings of the Second International Workshop on Visualization and Interaction for Ontologies and Linked Data*, pages 120–127, 2016.