RMLEditor: A Graph-Based Mapping Editor for Linked Data Mappings*

Pieter Heyvaert[1], Anastasia Dimou[1], Aron-Levi Herregodts[2]
Ruben Verborgh[1], Dimitri Schuurman[2]
Erik Mannens[1], and Rik Van de Walle[1]

[1]Data Science Lab (Ghent University - iMinds),
[2]Ghent University – iMinds – MICT
`pheyvaer.heyvaert@ugent.be`

Abstract. Although several tools have been implemented to generate Linked Data from raw data, users still need to be aware of the underlying technologies and Linked Data principles to use them. Mapping languages enable to detach the mapping definitions from the implementation that executes them. However, no thorough research has been conducted on how to facilitate the editing of mappings. We propose the RMLEditor, a visual graph-based user interface, which allows users to easily define the mappings that deliver the RDF representation of the corresponding raw data. Neither knowledge of the underlying mapping language nor the used technologies is required. The RMLEditor aims to facilitate the editing of mappings, and thereby lowers the barriers to create Linked Data. The RMLEditor is developed for use by data specialists who are partners of (i) a companies-driven pilot and (ii) a community group. The current version of the RMLEditor was validated: participants indicate that it is adequate for its purpose and the graph-based approach enables users to conceive the linked nature of the data.

1 Introduction

Semantic Web technologies rely on data which is interlinked and whose semantically enriched representation is available, the so-called Linked Data [1]. Most of the current Linked Data stems originally from (semi-)structured formats. Mappings specify in a declarative way how Linked Data is generated from such raw data. Nevertheless, defining and executing them still remains complicated, despite the significant number of tools implemented for this scope. At first, most approaches that map raw data to its RDF representation [2] incorporated the mappings in the implementation that executes them. Thus, not only knowledge of Semantic Web and Linked Data is required. However, also dedicated software development cycles for creating, updating and extending the implementations,

* The described research activities were funded by Ghent University, iMinds, the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research Flanders (FWO Flanders), and the European Union.

Pre-print of a paper accepted to the European Semantic Web Conference 2016 (ESWC 2016).The final publication is available at `link.springer.com`.

whenever new or updated semantic annotations are desired, are needed. Mapping languages, such as R2RML [3] and RML [4], enable *to detach the mapping definitions from the implementation that executes them.* Besides knowledge of the underlying mapping language that is required to define the mappings, manually editing and curating them requires a substantial amount of human effort [5]. Moreover, data specialists are not Semantic Web experts or developers. Thus, the task of editing mappings should be addressed independently, and disassociated from the corresponding mapping language and/or underlying technology used. Facilitating the editing of mappings further lowers the barriers of obtaining Linked Data and, thus stimulates the adoption of Semantic Web technologies. Nevertheless, dedicated environments that support users to intuitively edit mappings were not thoroughly investigated yet, as it occurred with applications that actually execute them and deliver its RDF representation. *Step-by-step* wizards prevailed, e.g., fluidOps editor [6], as an easy-to-reach solution. 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. We propose the RMLEditor[1], an editing environment for specifying mappings of raw data to their RDF representation based on graph visualizations, without requiring knowledge of the underlying mapping language. The RMLEditor is developed to support partners of (i) a companies-driven pilot for sharing and integrating the RDF representations of their data and co-develop third-party applications, and of (ii) a community-group-driven bootstrap for showcasing the advantages of Linked Data and Semantic Web technologies. The tool is available to interested parties under custom licensing conditions. We performed an exploratory user validation of our proposed solution, which showed that the RMLEditor achieves the goal it was implemented for. 82% of the participants found the use of graphs beneficial for editing mappings using the RMLEditor and 70% could better conceive that a relationship exists between multiple data sources. The remainder of the paper is structured as follows: Section 2 outlines existing mapping languages and mapping editors. Section 3 describes the mapping process without and with the use of a mapping editor. Section 4 presents our proposed solution, the RMLEditor. Section 5 outlines the use cases and explains the exploratory user validation and presents the results, Last, Section 6 discusses the results and presents the conclusions of our solution.

2 Related Work

In this section, we discuss existing mapping languages. Moreover, we elaborate on existing mapping editors, with a distinction between editors either supporting homogeneous or heterogeneous data sources.

[1] `http://rml.io/RMLeditor`

Pre-print of a paper accepted to the European Semantic Web Conference 2016 (ESWC 2016).The final publication is available at `link.springer.com`.

2.1 Mapping Languages

Mapping languages specify in a declarative way how Linked Data is generated from raw data. At first, existing formalizations were considered as mapping languages, such as XPath [7] or XQuery language [8]. Nevertheless, there were also languages defined for this particular task. R2RML [3] is the W3C-recommended language to define mappings to generate RDF from data derived from relational databases. Besides R2RML, other *format-specific* languages were defined, such as X3ML[2] for XML data. There are also *query-oriented* languages such as XSPARQL [9], which combines XQuery and SPARQL to map XML data, and Tarql[3], for data in CSV. However, these languages only support homogeneous data sources. A number of tools were developed supporting mappings from heterogeneous data sources to RDF, such as Datalift[4], RDFizers[5] and Virtuoso Sponger[6]. However, those tools actually employ separate *source-centric* approaches for each format they support, which does not allow the interlinking between sources in different formats. The RDF Mapping Language (RML) [4] circumvents this, by enabling the generation of data in RDF representation based on multiple heterogeneous data sources, e.g., XML and JSON.

2.2 Mapping Editors

Despite the significant number of mapping languages, the number of corresponding editors that support users to define the mappings is not comparable. Similar to the mapping languages, a distinction can be made between tools supporting homogeneous data sources and tools supporting heterogeneous data sources.

Homogeneous Data Sources The fluidOps editor [6] is a browser application that provides an intuitive user interface for editing mappings. The underlying mapping language is R2RML. The fluidOps editor relies on a single *step-by-step* workflow. There are six successive steps, similar to actually creating a mapping document. Although its Graphical User Interface (GUI) aims to hide the R2RML vocabulary, it still strongly focuses on concepts and terminology introduced by R2RML (e.g., subject maps and object maps). Therefore, knowledge of the language is required to use the editor. This decreases its adoption by non-Semantic Web experts and lowers the GUI's reusability potential. Additionally, only once the users reach the final step, they are able to preview the mappings in R2RML syntax and identify possible inconsistencies. Consequently, they need to restart the workflow to update the definitions of the previous steps. Pinkel et al. [5] adapted the original fluidOps editor to overcome flexibility limitations imposed by the database-driven step-by-step workflow. Their extension supports

[2] `https://github.com/delving/x3ml/blob/master/docs/x3ml-language.md`

[3] `https://tarql.github.io/`

[4] `http://datalift.org/`

[5] `http://simile.mit.edu/wiki/RDFizers`

[6] `http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtSponger`

the *ontology-driven* approach. With the former approach creating the mappings starts with the data in the databases, semantic annotations are added afterwards. With the latter approach the mappings are created based on an existing ontology. Next, the mappings are complemented with data fractions from the databases. Sheet2RDF [10] is a platform that uses a PEARL [11] document to map data in spreadsheets to RDF. Its GUI allow users to view the source data, define the mappings by editing the PEARL document directly, and view the resulting RDF through a tabular-structure. However, the adoption of the tool decreases because users need knowledge about PEARL to edit the mappings. An alternative approach is proposed by Rodrıguez-Muro et al. [12] who introduced -ontopPro-[7], a plugin for Protégé [13]. It allows users to generate RDF based on data from database(s) and an ontology. Tabs are provided to manage the databases and the mappings. Users need to write SPARQL-like templates to define how to map the original data. However, data sources are limited to databases and users need to understand the template's custom syntax.

Heterogeneous Data Sources Karma[8] differentiates from aforementioned tools because it supports heterogeneous sources, such as databases, delimited text files (e.g., CSV files), JSON, XML, Microsoft Excel and web APIs. It uses Global-Local-As-View [14] rules to perform the mappings. These rules can be exported using R2RML or D2RQ [15]. When displaying the mappings to the users, Karma takes a data-centric approach: users can only view the input data. Users are not able to follow the ontology-driven approach, as it occurs with the fluidOps editor. DataOps [16] uses the latter to support heterogeneous data formats. However, users are still confronted with the syntax of the used languages. RDF123 [17] is similar to Sheet2RDF, however, it also supports CSV files, and it uses custom *map graphs* to represent the mappings, which are converted to a custom map function that produces RDF based on the input data. Consequently, users are not required to understand or know about the map function to define mappings, as it occurs with PEARL for Sheet2RDF. Additionally, they offer a web service that uses a link to a Google Spreadsheet or a CSV file, and generates RDF based on the mappings defined with the application. Therefore, the maps can be used outside the desktop application. However, their use is limited to that web service, as a custom map function is used and not a mapping language. TopBraid Composer[9] supports heterogeneous sources. It allows data integration from databases, XML, UML, RSS, spreadsheets and RDF data backends. The data can be reconciliated with DBpedia [18]. However, as in the case of Karma, interlinking data from different sources is not possible. To set up the mapping process, the GUI offers a *data-driven step-by-step* workflow. However, an ontology-driven approach is not possible. Similar to the original fluidOps editor, a more simplified wizard has been built on top of RML, instead of R2RML. This form-based

[7] `http://ontop.inf.unibz.it/components/sample-page/`

[8] `http://www.isi.edu/integration/karma/`

[9] `http://www.topquadrant.com/tools/modeling-topbraid-composer-standard-edition/`

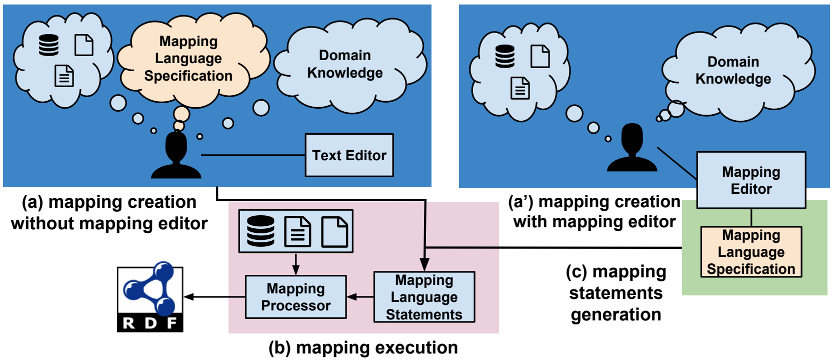

Mapping Language Specification
Domain Knowledge
Text Editor
(a) mapping creation without mapping editor
Domain Knowledge
Mapping Editor
(a') mapping creation with mapping editor
Mapping Language Specification
Mapping Processor
Mapping Language Statements
R D F
(c) mapping statements generation
(b) mapping execution

Fig. 1: The difference in the mapping process if during the mapping creation a mapping editor is used or not.

browser application[10] supports heterogeneous data sources, because it has RML as its underlying mapping language, compared to only homogeneous sources (i.e., databases) which are supported by the fluidOps editor. Last, OpenRefine[11] is a browser-based tool for cleansing raw data, changing the data format and incorporating external data using Web services. The RDF Refine extension[12] [19] allows users to export the data in RDF. A RDF graph is used to visualize the mappings. However, the RDF graph is forced in a hierarchy-layout, which weakens the advantages of using a graph representation. Additionally, this extension supports reconciliation services that offer HTTP interfaces. User intervention is needed to assess the quality of each reconciliation. A preview of candidate entities is available to help the user.

3 Mapping Process

The mapping process is a series of steps performed in order to generate Linked Data from raw data. There are two variations of the process, depending on whether a mapping editor is used or not.

Mappings without Editor Generating RDF from (semi-)structured data, using a mapping language without a mapping editor, consists of two consecutive steps: First, the mapping is created (see Figure 1a). In this step, the user needs to be aware of the input data and has to have knowledge about the domain and the *mapping language's specification*. The latter is, in principle, possessed by *Semantic Web experts*. In most cases, a text editor is used to create the mappings. Statements in the mapping language follows this step. In the second step the

[10] http://pebbie.org/mashup/rml
[11] http://openrefine.org/
[12] http://refine.deri.ie/

statements together with the input data sources are used by the mapping processor to generate the RDF triples (see Figure 1b). The mapping processor is a tool that knows how to interpret the mapping language's specification to generate the RDF representation of the corresponding data, taking into consideration the statements derived from the previous step.

Mappings with Editor When the mapping process is done with a mapping editor, the users do not longer need to be aware of the *mapping language's specification* (see Figure 1a'). This allows *non-Semantic Web experts* to define the mappings. Additionally, the text editor is replaced by the mapping editor. The mapping editor knows how to interpret the mappings created by the user using the editor based on the language's specification. Subsequently, it generates the mapping language statements (see Figure 1c). Important to note is that in this step no user knowledge about the specification is needed. Subsequently, the mapping is executed, and RDF triples are generated.

4 RMLEditor

The RMLEditor is a browser-based GUI with the goal to support users in production environments to define, in a uniform way, mappings that specify how to generate Linked Data represented using the prevalent RDF framework. In previous work [20], we listed 7 desired features of a GUI for uniform mapping editors. First, it should be independent of the underlying mapping language, so that users are able to create mappings without knowledge of the language's syntax. Second, it should allow users to execute the mappings outside of the editor, because it is only meant to create the mappings. Third, it should enable users to map multiple data sources at the same time, as it might occur that data is spread across multiple sources. Fourth, the editor should support data sources in different data formats, as the generation of Linked Data should be independent of the original format. Fifth, as multiple ontologies and vocabularies can be used to create a mapping, an editor should support the use of both existing and customs ontologies and vocabularies. Sixth, it should allow multiple alternative modeling approaches, as certain use cases might benefit from using a specific approach. Finally, by supporting non-linear workflows, users are able to keep an overview of the mapping model and its relationships. The GUI of the RMLEditor is designed to implement these features. The GUI uses graphs to visualize the mappings. Manipulation of these graphs results in creating, updating and extending the mappings, which can be done without any knowledge of the underlying mapping language or other used technologies. The graphs express how the raw data will be represented as RDF. However, this expressiveness is independent of the language's expressiveness. The RMLEditor triggers the mapping processor which executes the mappings exported by the RMLEditor and generates RDF statements. For the RMLEditor, we chose RML which can support mappings derived from a GUI that covers all of the aforementioned features. However, any other mapping language could be used instead, if it allows to implement the features. In Section 4.1 we discuss the RMLEditor's architecture. In Section 4.2, we elaborate on how the features are implemented in the GUI. In Section 4.3, we explain how the RMLProcessor, the mapping processor for RML, is used with the RMLEditor. In Section 4.4, we present two real-life use cases of the RMLEditor.

4.1 Architecture

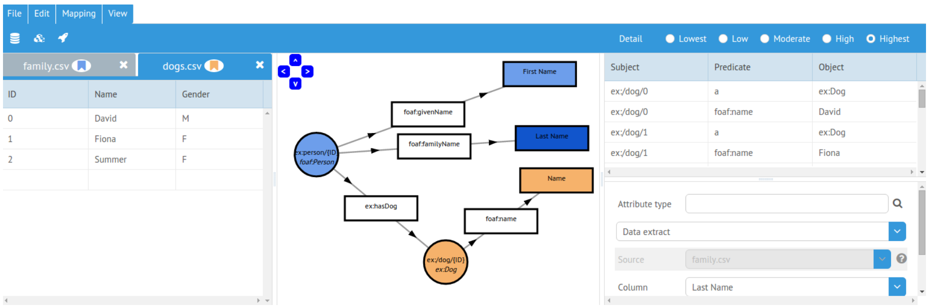
The RMLEditor's high-level architecture is based on the *multilayered architecture pattern* [21]. This allows to separate the presentation and the logic of the mappings, using the *presentation layer* and *application layer*, respectively. The loading of mappings and data sources is done using the *data access layer*. The latter only communicates with the application layer. Communication between the presentation layer and the data access layer is not possible, as the architecture prohibits communication between layers that are not directly under or above each other. For the presentation layer, the Webix JavaScript library[13] is used to build the GUI, in cooperation with the d3.js library [22] for the presentation of the graphs. The communication with application layer is facilitated by the *Model-View-Controller* pattern [23]. The Graph Markup Language (GraphML) [24] is used to represent the graph visualization of the mappings independently of the underlying mapping language. This allows users to export the graphs in an application-independent format. Additionally, the GraphML-version of the graphs are used to generate the corresponding RML statements. Users are able to load graphs by instructing the RMLEditor to load the corresponding GraphML document. When RML statements need to be loaded, the statements are first converted to a GraphML document, then interpreted as graph elements, and shown in the GUI.

4.2 Graphical User Interface

The graphical user interface of the RMLEditor allows users to define mappings on existing data. To implement the aforementioned features for the GUI, the RMLEditor offers three panels to the users: *Input Panel*, *Modeling Panel* and *Results Panel* (see Figure 2). They are aligned next to each other, however, when users want to focus on a specific panel, they are able to hide the other panels. The *Input Panel* shows the data sources to users (Feature 3). Each data source is assigned with a unique color. Depending on the data format, an adequate visualization is chosen (Feature 4). The *Modeling Panel* shows the mappings using a graph representation. The color of each node and edge depends on the data source that is used in that specific mapping, if any. It offers the means to manipulate the nodes and edges of the graphs in order to update the mappings. Semantic annotations can be added using multiple vocabularies and ontologies (Feature 5). The Linked Open Vocabularies[14] (LOV) can be consulted via the GUI to get

[13] `http://webix.com/`

[14] `http://lov.okfn.org/dataset/lov/`



ID	Name	Gender
0	David	M
1	Fiona	F
2	Summer	F

Subject	Predicate	Object
ex:/dog/0	a	ex:Dog
ex:/dog/0	foaf:name	David
ex:/dog/1	a	ex:Dog
ex:/dog/1	foaf:name	Fiona

Attribute type:

Data extract

Source: family.csv

Column: Last Name

Fig. 2: The RMLEditor with the *Input Panel* on the left, the *Modeling Panel* in the center and the *Results Panel* on the right

suggestions on which classes, properties and datatypes to use. As the graphs offer a generic representation of the mappings, because they do not depend on the underlying mapping language, this panel addresses Feature 1. Additionally, the graph representation and the RML statements can be exported (Feature 2), allowing the execution of the mappings outside the RMLEditor. The *Results Panel* shows the resulting RDF dataset when the mappings defined in the *Modeling Panel* are executed on the data in the *Input Panel*. For each RDF triple of the dataset it shows the subject, predicate and object. The functionality and the interaction between the panels supports the different mapping generation approaches, as we described in previous work [20, 25] (Feature 6). The *data-driven* approach uses the input data sources as the basis to construct the mappings. The classes, properties and datatypes of the schemas are then assigned to the mappings. When users start with the vocabularies and ontologies to generate the mappings, the *schema-driven* approach is followed. Next, data fractions from the data sources can be associated to the mappings. Additionally, by not restricting users in when to interact with which panels – as would be the case for linear workflows – the RMLEditor supports non-linear workflows (Feature 7).

4.3 RMLProcessor Server

As the RMLEditor uses RML, it needs the functionality of the RMLProcessor[15], a Java application which generates RDF based on provided RML mapping documents. However, the processor is not needed to define the mappings. For the RMLEditor's needs, the RMLProcessor's functionality is offered through a Web API. The Web API is developed using Node.js[16] and offers three functions: (i) executing a mapping document on a set of data sources; (ii) converting a GraphML document to RML to execute the mappings using the RMLProcessor, and (iii) converting RML statements to a GraphML document to visualize the mappings that are loaded in the RMLEditor.

[15] `https://github.com/RMLio/RML-Mapper`

[16] `https://nodejs.org/`

4.4 Real-Life Use Cases

The RMLEditor is developed for a pilot (COMBUST[17]), initiated by companies in Flanders who aim to build their collaboration network on top of their interlinked data. The RMLEditor covers their need to generate the RDF representation of their raw data to be further used in other third-party applications. Moreover, the RMLEditor supports the partners of the Open Tourism working group of the Belgian Chapter of Open Knowledge Foundation (OKFN)[18] to semantically annotate their data with the Open Standard for Tourism Ecosystems Data[19] in the frame of the *Sustainable Mobile Guides for Tourism (in Flanders)* bootstrap. In both cases, the partners of the project have the raw data and the required knowledge about the domain. However, they have no understanding of mapping languages, and without the use of the RMLEditor they need assistance of a Semantic Web expert. Additionally, for every update and execution of the mappings the expert is consulted, which introduces significant overhead. We collected feedback during the deployment of the RMLEditor and concluded that they were able to create mappings for their data using their own domain knowledge. This is confirmed during our exploratory user validation, which involved members of the aforementioned pilot (see Section 5). Furthermore, the RMLEditor is the topic of a number of workshops and tutorials for interested groups and users[20].

5 Exploratory User Validation

We performed an exploratory user validation to assess the RMLEditor's adequacy to support users in defining mappings that generate RDF datasets. In Section 5.1 we discuss the two use cases that the participants completed. In Section 5.2 we explain the two groups of participants, the apparatus and the procedure followed during the validation. In Sections 5.3 and 5.4 we elaborate on both the subjective and objective aspect of the validation. Finally, in Section 5.5 we discuss the results.

5.1 Use Cases

In this section, two use cases are outlined and it is explained in more detail how the RMLEditor is used to generate the mappings. Each use case covers a different mapping generation approach: *data-driven* or *schema-driven* [25]. The first use case involves data about employees and projects they work on, originally in two different data sources. The goal is to semantically annotate them, and link the employees to the projects they work on. The *data-driven* approach is recommended for editing the mappings to RDF. Users start by loading the two data sources into the RMLEditor. Next, mappings are generated based on data

[17] `http://www.iminds.be/en/projects/2015/03/11/combust`

[18] `http://www.openknowledge.be/`

[19] `http://tourism.openknowledge.be:8080/spec/`

[20] `http://rml.io/RMLevents`

fractions from the sources. Subsequently, users semantically annotate the mappings. Identifying suitable classes, properties and datatypes is supported by the LOV. The second use cases involves data about movies and their directors. The goal is to semantically annotate them, and to interlink each movie to the person that directed it. The *schema-driven* approach, supported by the RMLEditor, is recommended to the users for editing the mappings. Users start by generating mappings reusing movie concepts from the DBpedia Ontology[21] and person concepts from FOAF[22]. When the modeling is completed, the data sources are loaded. Subsequently, the mappings are updated with the data fractions to be used for generating the final RDF dataset.

5.2 Method

Participants 15 participants from both Ghent University and the COMBUST network took part. They were divided in two major groups: (i) *Semantic Web experts* with 10 participants and (ii) *non-Semantic Web experts* with 5 participants. A *Semantic Web expert* is a user who has knowledge about Semantic Web technologies and standards, including Linked Data and RDF. A *non-Semantic Web expert* is not aware of the Semantic Web technologies, including the editor's underlying language RML. The *Semantic Web experts* are further distinguished in two sub-categories: (i) 5 experts with experience in Linked Data publishing, and (ii) 5 experts with no previous experience in Linked Data publishing.

Apparatus The evaluation was carried out using the current version of the RMLEditor. Participants used their own computer with Chrome[23]. Both the RMLEditor and the RMLProcessor were hosted on the same physical server. Each participant participated in both use cases described in Section 5.1.

Procedure The following steps were followed during the evaluation:

Step 1 We gave a presentation[24] to the participants where basic concepts, such as Linked Data, RDF and schemas were explained. In order for them to understand the purpose of the RMLEditor. Additionally, the user interface's panels were described. However, no introductory tutorial regarding how to use the RMLEditor was given.

Step 2 We conducted a pre-assessment by asking the participants to fill in a questionnaire.

Step 3 Half of each group's participants started with Use Case A, and the other half with Use Case B. This was done to eliminate the influences of the use case-specific data and editing approach on the final results of the evaluation. Subsequently, they completed the other use case. During the execution of the

[21] `http://dbpedia.org/ontology/`

[22] `http://xmlns.com/foaf/0.1/`

[23] version 45.0.2454.101 or higher; `https://www.google.com/chrome/`

[24] `http://www.slideshare.net/secret/vI6AOOywqzyk1m`

use cases, the experts were able to ask questions when they had problems with the RMLEditor. It was recorded when intervention was needed. For the *non-Semantic Web experts*, closer observation was conducted and the think-aloud method [26] was taken into consideration. This allowed for the identification of specific usability issues related to the RMLEditor. After each use case, we conducted an assessment to inquire about the difficulty of the use case. **Step 4** (Experts with prior RML knowledge only) Half of the experts, with knowledge of the underlying mapping language RML, also had to create a mapping using a plain text editor, to observe differences and preferences between use and non-use of the RMLEditor. However, this could not be done by all participants, as they do not possess the required knowledge.

Step 5 At the end of each use case, we collected the created mappings. Additionally, a second and third questionnaire were filled in by the participants after their first and second use case, respectively.

Step 6 We conducted a post-assessment by providing the participants with a fourth questionnaire to fill in.

5.3 Subjective Validation

We conducted a subjective validation of the RMLEditor by presenting the participants with four questionnaires during the validation in step 2, 5 and 6. With the pre-assessment we assess the participants expectations before interacting with the RMLEditor. For the pre-assessment's questionnaire, we used the System Usability Scale (SUS) scale [27]. The SUS statements were translated to the use of the RMLEditor. The intermediate questionnaires of step 5 allowed for a subjective self-assessment of the previously completed use case, and more specific the approach of the specific use case. With the post-assessment we assess the participants' experience with the RMLEditor and determine what the positive and negative aspects are of the RMLEditor. For the intermediate questionnaires and the post-assessment's questionnaire, we used the 7-point Likert scale [28] from 'not difficult at all' to 'very difficult' to measure difficulty, from 'not useful at all' to 'very useful' to measure usefulness, and from 'not agree at all' to 'very much agree' to measure the degree of agreement to a given statement.

5.4 Objective Validation

After each completed use case, we collected each participant's mapping. We conducted a objective validation of the RMLEditor by comparing all mappings to the baseline mapping that we created ourselves. Additionally, if a participant also created a mapping using RML directly, it was compared with the participant's mapping created using the RMLEditor.

5.5 Results

During the pre-assessment we measured that both groups of participants had high expectations regarding the ease-of-use and the improvements the RMLEditor would bring to the mapping process. Interesting to note is that *Semantic Web experts* expected the tool to be hard to use if no introduction tutorial is provided, in contradiction to the *non-Semantic Web experts*.

Learning curve After having completed the first use case in row, we measured via the subjective validation that 60% of both the Semantic Web experts and non-experts found it difficult to use the RMLEditor to define the mappings that generate the RDF representation. After having completed the second use, all non-experts found it easier to use the RMLEditor. However, 30% of the experts still found it difficult to use the RMLEditor over all. This is partially due to the fact that they all had high expectations of the RMLEditor. All participants needed at least once help during the first in row use case. However, during the participants their second use case, 40% of the *Semantic Web experts* did not need any help at all anymore. However, all *non-Semantic Web experts* still needed help during their second use case. Nevertheless, over all, a significant lower level of intervention by us was needed. We deduced that the completeness and accuracy of the mappings increased when users performed their second use case, compared to their first one. This shows that there is a learning curve to use the RMLEditor, however, once users learn how to use it, the RMLEditor is adequate for its scope. The latter is verified via subjective validation by the fact that 47% of the participants found that the overall usage of the RMLEditor was not difficult. However, still 20% found it too difficult to use. Again, this is partially due to the fact that the participants had high expectations of the RMLEditor. Even though the RMLEditor aims to eliminate language and RDF-specific terminology, through the observation during the validation we found that 40% of the participants had trouble with the used terminology in the GUI, such as ‘child’ and ‘parent’ when data sources need to be interlinked, and ‘templates’ when the URIs of the entities are constructed based on a data fraction from the data source. However, once explained to the users, 90% of the users could use the terminology as intended.

Editing approaches 67% of the participants were able to start editing the mappings using the given approach with no assistance at all. Through oral feedback during the evaluation, participants stated their preferred approach. Because of that preference, when they were asked to follow the other approach, they stated it was counterintuitive. If users prefer the *data-driven* approach, the *schema-driven* approach is counterintuitive, because no data is loaded initially. If users prefer the *schema-driven* approach, the *data-driven* is counterintuitive, because no predefined schema to be used is given.

Graph visualizations During the post-assessment, participants were asked to what extend they agree with the statement that the use of graph is beneficial for editing mappings, and the statement that the graphs make the linked nature of the final RDF dataset clear. Through the subjective validation, we found that 82% of all participants found the use of graphs beneficial for editing mappings, and that graph-based visualizations are adequate for conceiving how the final RDF dataset will be. The objective validation showed that in the case of *Semantic*

Web experts, in only 15% of the total twenty use cases there were incomplete or inaccurate mappings, regarding the modeling of the domain. In 67% of the use cases they were able to create mappings as complete and accurate as expected.

Linking data sources In 33% of the use cases, participants missed the interlinking between multiple data sources when using RML, when we compared the mapping created with the RMLEditor and the mapping created directly using RML, via the objective validation. Through the post-assessment, we concluded that the links were missing because of the difficulties with the terminology of the language, or because the participants just forgot the interlinking. With the RMLEditor this only happened in 10% of the use cases. However, it still occurred because not all terminology was well covered in the GUI, as the participants made clear in the post-assessment.

6 Discussion & Conclusions

During the assessment both groups of participants were able to generate Linked Data through the RMLEditor's GUI, with the graph-based visualization as the most important contributor. Therefore, the RMLEditor fulfills its initial goal to supports users to define, in a uniform way, mappings that specify how to generate Linked Data. We present four main findings. First, non-Semantic Web experts are able to generate Linked Data when using the RMLEditor. Additionally, for experts, the quality of the Linked Data is better when using the RMLEditor instead of RML directly, by looking at the mapping created with the RMLEditor and the mapping created by using RML directly. Therefore, second, when Semantic Web experts generate Linked Data using the RMLEditor, the resulting RDF dataset is of at least the same quality as the dataset generated when using directly the RML language, if quality metrics, defined by the Semantic Web experts, are kept constant. Furthermore, for example, when a relationship between two data sources exists, however, not made explicit by the user, this is reflected in the graph: a link between the resource nodes belonging to the sources is not present. Using RML directly, finding the missing relationship is more difficult. Therefore, third, graph-based visualizations of mappings improve the interlinking between multiple data sources during the mapping creation, compared to the use of no visualizations. Nevertheless, improvements to the RMLEditor, and more specifically to the GUI can be made during the next development sprints. As the main aspect of the RMLEditor is the use of graphs, it is important that manipulations on them are made as easy as possible. Although the required functionality for the manipulations is available, improving even more its accessibility will benefit the mapping process. Additionally, allowing users to determine the size of the graphs allows them to select the desired detail-level of their mappings. As the participants where presented with two use cases, they were able to perceive a learning curve. Future evaluations need to be conducted to determine how steep this learning curve is. However, as the fourth finding, by providing a set of tutorials, where the features of the RMLEditor are discussed will already improve the initial use of the RMLEditor by new users. Topic of future research is the validation of these observations during further large-scale user testing. Moreover, the RMLEditor supports both the *data-driven* and *schema-driven* approach. Whether a user starts with the use case using the *data-driven* or the *schema-driven* approach, the second use case in row is less difficult. Therefore, there is no approach enforced on the user by the RMLEditor, and the preferred approach depends on the person and the circumstances [5]. Both Semantic Web experts and non-Semantic Web experts have different expectations from the RMLEditor. Non-Semantic Web experts expect that they can generate Linked Data without any knowledge about a mapping language or Semantic Web technology. Semantic Web experts can already generate Linked Data, as they can use RML directly. Therefore, they expect that they can at least do the same as with RML. Additionally, they have a set of requirements for their Linked Data, for which they look in the RMLEditor. These requirements improve, according to them, the quality of their generated Linked Data. In the long run, we want to see these expectations united. Non-Semantic Web experts do not only generate Linked Data. They are also concerned with the quality of their Linked Data, in order to further improve its adoption.

References

[1] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts*, 2009.

[2] Dan Brickley and Ramanathan Guha. RDF Schema 1.1. Working group recommendation, W3C, February 2014. URL `http://www.w3.org/TR/rdf-schema/`.

[3] 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/`.

[4] 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.

[5] Christoph Pinkel, Carsten Binnig, Peter Haase, Clemens Martin, Kunal Sengupta, and Johannes Trame. How to best find a partner? An evaluation of editing approaches to construct R2RML mappings. In *The Semantic Web: Trends and Challenges*. Springer, 2014.

[6] Kunal Sengupta, Peter Haase, Michael Schmidt, and Pascal Hitzler. Editing R2RML mappings made easy. In *Proceedings of the 12th International Semantic Web Conference*. 2013.

[7] Anders Berglund, Scott Boag, Don Chamberlin, Mary F Fernández, Michael Kay, Jonathan Robie, and Jérôme Siméon. XML path language (XPath). *World Wide Web Consortium (W3C)*, 2003.

[8] Scott Boag, Don Chamberlin, Mary F Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, and Mugur Stefanescu. XQuery 1.0: An XML query language, 2002.

[9] Stefan Bischof, Stefan Decker, Thomas Krennwallner, Nuno Lopes, and Axel Polleres. Mapping between rdf and xml with xsparql. *Journal on Data Semantics*, 1(3):147–185, 2012. ISSN 1861-2032.

[10] Manuel Fiorelli, Tiziano Lorenzetti, Maria Teresa Pazienza, Armando Stellato, and Andrea Turbati. Sheet2rdf: a Flexible and Dynamic Spreadsheet Import&Lifting Framework for RDF. In *Current Approaches in Applied Artificial Intelligence*. Springer, 2015.

[11] Maria Teresa Pazienza, Armando Stellato, and Andrea Turbati. Pearl: Projection of annotations rule language, a language for projecting (uima) annotations over rdf knowledge bases. In *LREC*, 2012.

[12] Mariano Rodrıguez-Muro, Josef Hardi, and Diego Calvanese. Quest: efficient SPARQL-to-SQL for RDF and OWL. In *11th International Semantic Web Conference ISWC 2012*, page 53. Citeseer, 2012.

[13] Natalya F Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W Fergerson, and Mark A Musen. Creating semantic web contents with protege-2000. *IEEE intelligent systems*, (2):60–71, 2001.

[14] Marc Friedman, Alon Y Levy, Todd D Millstein, et al. Navigational plans for data integration. *AAAI/IAAI*, 1999:67–73, 1999.

[15] Christian Bizer and Andy Seaborne. D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs. In *Proceedings of the 3rd international semantic web conference (ISWC2004)*, volume 2004. Citeseer Hiroshima, 2004.

[16] Christoph Pinkel, Andreas Schwarte, Johannes Trame, Andriy Nikolov, Ana Sasa Bastinos, and Tobias Zeuch. DataOps: Seamless End-to-End Anything-to-RDF Data Integration. In *The Semantic Web: ESWC 2015 Satellite Events*, pages 123–127. Springer, 2015.

[17] Lushan Han, Tim Finin, Cynthia Parr, Joel Sachs, and Anupam Joshi. *RDF123: from Spreadsheets to RDF*. Springer, 2008.

[18] 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), 2009.

[19] Fadi Maali, Richard Cyganiak, and Vassilios Peristeras. Re-using Cool URIs: Entity Reconciliation Against LOD Hubs. *LDOW*, 813, 2011.

[20] 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 Workshop on Intelligent Exploration of Semantic Data*, October 2015.

[21] Mark Richards. *Software Architecture Patterns*. O'Reilly, 2015.

[22] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12), 2011.

[23] Addy Osmani. *Learning JavaScript Design Patterns*. O'Reilly Media, 2012.

[24] Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. GraphML progress report structural layer proposal. In *Graph Drawing*. Springer, 2002.

[25] Pieter Heyvaert, Anastasia Dimou, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Approaches for Generating Mappings to RDF. In *Proceedings of the 14th International Semantic Web Conference: Posters and Demos*, October 2015.

[26] Elizabeth Charters. The use of think-aloud methods in qualitative research an introduction to think-aloud methods. *Brock Education Journal*, 12(2), 2003.

[27] John Brooke. SUS - A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[28] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.