How to Best Find a Partner? An Evaluation of Editing Approaches to Construct R2RML Mappings [⋆]

Christoph Pinkel[1], Carsten Binnig[2], Peter Haase[1], Clemens Martin[2], Kunal Sengupta[3], and Johannes Trame[1]

[1] fluid Operations AG, Walldorf, Germany
[2] Baden-Wuerttemberg Cooperative State University, Mannheim, Germany
[3] Wright State University, Dayton OH, USA

Abstract. R2RML defines a language to express mappings from relational data to RDF. That way, applications built on top of the W3C Semantic Technology stack can seamlessly integrate relational data. A major obstacle to using R2RML, though, is the effort for manually curating the mappings. In particular in scenarios that aim to map data from huge and complex relational schemata (e.g., [5]) to more abstract ontologies efficient ways to support the mapping creation are needed.
In previous work we presented a mapping editor that aims to reduce the human effort in mapping creation [12]. While assisting users in mapping construction the editor imposed a fixed editing approach, which turned out to be not optimal for all users and all kinds of mapping tasks. Most prominently, it is unclear on which of the two data models users should best start with the mapping construction.
In this paper, we present the results of a comprehensive user study that evaluates different alternative editing approaches for constructing R2RML mapping rules. The study measures the efficiency and quality of mapping construction to find out which approach works better for users with different background knowledge and for different types of tasks.

Keywords: #eswc2014Pinkel

1 Introduction

Motivation: The **R**DB to **R**DF **M**apping **L**anguage (R2RML[4]) has recently become a W3C standard for creating mappings from relational databases to RDF. This enables many semantic web applications to integrate easily with relational databases. Although very useful, we observe certain problems with the adoption of R2RML: (1) creating R2RML rules manually is a time consuming process, (2) even simple rules can be syntactically heavy in terms of the R2RML vocabulary, and (3) a steep learning curve is involved in gaining expertise of this new

[⋆] This work was supported by the 7th Framework Program of the EU Commission under grant agreement 318338 and by the US NSF under award 1017225.
[4] http://www.w3.org/TR/r2rml/

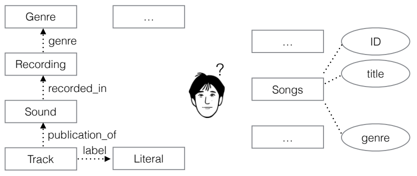


Fig. 1: Example Mapping Task (Left: Ontology, Right: Relational Schema)

language. All these issues essentially result in a high manual effort. In scenarios where data is mapped from huge and complex relational schemata to RDF, the manual effort is particularly high. In previous work [12] we demonstrated the initial version of an R2RML editor that aims to reduce the manual effort.

Problem Statement: The amount of effort that users invest for writing mapping rules depends on the mapping creation process. The usefulness of an editor therefore depends on how well it supports users in this process. Is the editing approach aligned with user expectations? Does the editor direct users in some specific direction and, if so, is this direction helpful? Or could users choose between procedural alternatives at their own discretion? Our editor initially imposed a strict editing approach, which turned out to be not optimal for many users and mapping tasks.

R2RML as a language, on the other side, leaves users a lot of of freedom about the order in which they compile different parts of a mapping rule. For example, you could start by first defining the RDF target of a mapping rule or you could start by selecting source tables from a relational database. R2RML also leaves it to the user whether to compose a separate rule for each mapping or to group many associated mappings into the same mapping rule.

Though there is a number of different approaches that a user may follow, two particular alternatives stand out: (1) the *database-driven* mapping approach where users work through the relational schema table by table and write mapping rules for all data in the tables that they find useful and (2) the *ontology-driven* mapping approach where users browse schematic aspects in an existing ontology (such as classes and properties) and then write mapping rules to add appropriate A-Box facts from the database. Essentially, these approaches start at opposite ends of a mapping. To build an efficient R2RML mapping editor we need to know which approach works better under which circumstances.

Example: Figure 1 depicts a mapping problem where instances of type *Track* should be constructed based on data in the table *Songs* of a relational database. Though both models describe the *genre* of a track, the genre is only indirectly connected to the *Track* class but directly connected to the *Songs* table.

Imagine a user trying to map the *Song* table to instances of type *Track* of the ontology, approaching the task from the database side: the user would start with exploring the table *Song*, easily identify the *ID* and the *title* attributes to construct the URIs for the *Track* instances and their labels. However, the user could have a hard time finding a mapping partner for the *genre* attribute of the table *Songs* in the ontology. In fact, what the user will have to do is (1) browse the ontology to find the corresponding partner (class *Genre*), (2) write a new R2RML rule to construct instances of class *Genre* in the ontology and (3) see how to construct triples all required triples to connect *Track* and *Genre*. This can prove to be a difficult task for a user. In this particular case, the opposite direction would appear more appealing.

Contributions: In this paper, we present the results of a comprehensive user study that evaluates different alternative editing approaches to support the construction of R2RML mapping rules in our editor. We therefore extended the mapping editor to support different approaches.

We put our main focus on the ontology-driven and database-driven approaches. Consequently, one hypothesis that we tested in our user study is that both approaches accommodate the preferences of users with a different background knowledge. For example, database experts might prefer to proceed differently than ontology experts. In another hypothesis, we assumed that either mapping approach offers different advantages and disadvantages for different mapping tasks. For example, for a mapping task where a small ontology requires only a few facts out of a huge database, the ontology-driven approach might generally appear more reasonable.

Outline: The remainder of this paper is organized as follows: Section 2 discusses different approaches implied by R2RML for constructing mapping rules. Section 3 presents the existing R2RML editor that we have extended for this study. In Section 4, we present the design of our study along a set of key questions and discuss the results of the study. Section 5 describes related work. We conclude and discuss possible future work in Section 6.

2 R2RML Editing Approaches

R2RML as a language leaves the user much freedom about how to compose mapping rules. However, logical dependencies in many cases suggest a natural order of steps for writing rules.

2.1 Implied Editing Approaches in R2RML

Figure 2 shows the basic structure of R2RML mapping rules. Mapping rules are

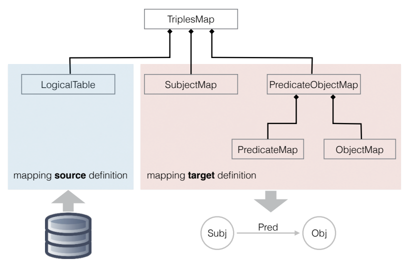


Fig. 2: Main Aspects of R2RML Mapping Rules

called *triples maps*. Each triples map consists of (1) a *logical table*, which defines a view on the source database, (2) a *subject map*, which defines target instance URIs and types, and (3) any number of *predicate/object maps*, adding triples to those instances.

For example, if you wish to add a mapping for songs and their titles from a table *Songs* to an ontology class *Track*, you might write a mapping rule with the following components: (1) a logical table that builds a view on table *Songs*, (2) a subject map that constructs a unique URI for each song tuple and types it as a *Track* and (3) one predicate/object map with predicate *dc:title* and object literals constructed from the *title* attribute in table *Songs*.

You may generally add those parts in any order. In the following we describe the choices a user can make based on R2RML as a language.

Mapping Direction: Most importantly, users may start by defining the logical table or by defining the subject map, i.e., they might:

1. Proceed *database-driven* by defining views over the source database.
2. Proceed *ontology-driven* by first specifying ontology classes.

Both approaches imply a different kind of thinking. Users can either think of existing database tables or of the target ontology and required information.

Subject Map Definition: Any triple depends on its subject, which is defined in a subject map. It would thus sound natural to specify the subject map before any predicate/object maps are defined. However, you may also consider the subject as implicitly given and specify it later.

Predicate/Object Map Definition: Predicate/object maps each contain a predicate map and an object map. The obvious order here is to first specify the predicate then the object but the other way around is also possible.

Predicate/Object Map Separation: The fact that each mapping rule can have only one subject map but any number of predicate/object maps suggests that as many predicate/object maps as possible should be added to the same rule. However, different properties may rely on different parts of the source database. Therefore, this assumption adds potentially heavy requirements on the source database view, i.e., on the logical table of the mapping rule. Depending on the required views this may be or may not be adequate in practice. Hence, in cases where the views would become too complex it can make sense to construct different mapping rules for each predicate/object map. Each of those rules would then reference the same subject map but a different logical table.

Incremental Rule Extensions: Finally, mapping rules may undergo many iterations, especially when working with complex data. For example, predicate/object maps may be added. Also, logical tables might be adjusted to cover a wider selection of data. The second case is particularly interesting. It basically represents the opposite strategy of predicate separation: add another rule for a new predicate or extend the logical table? Also, in some cases it may have implications on the correctness of previously added parts of a mapping.

2.2 Supported Approaches in R2RML Editors

Editors, while assisting users in various ways, may also restrict their freedom by forcing them to work with one specific approach.

In our search for the best approach we focus on mapping direction, i.e., on the choice between the database driven and ontology driven approach.

Handling subject definition is easy with editors as they can always construct a subject automatically. Users may later change these subject maps but there is no good reason to force them to edit subject maps at one specific point in time.

Predicate/object map definition plays no role in editors because a single dialog would be used to associate both. Therefore, users are always free to proceed either way. Predicate separation, i.e., the choice to construct several smaller rules instead of one single large rule may or may not be supported by editors. Similarly, incremental build-up may or may not be supported.

3 The R2RML Mapping Editor

In this paper, we use our R2RML editor [12] as a basis. For the purpose of the study we extended our editor to support different editing approaches. In the following, we first describe the original editor and then discuss our extensions.

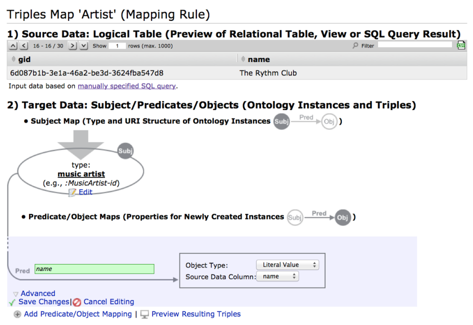
3.1 Basic Editor (Original Version)

In terms of individual features the editor provides a user interface that hides the R2RML vocabulary details, allows an easy access to schema meta-data of the relational database for which the mappings are to be created and gives feedback at each step, such as previewing triples. The original workflow supported by the editor follows a strict step-by-step pattern which can be described as follows:

1. **Datasource & Base URI Selection:** The user chooses data sources (i.e., databases and ontologies) to work with.
2. **R2RML Rule Selection:** At this point the user may choose to edit an existing rule or add a new one.
3. **Logical Table Selection:** Logical tables in R2RML are either database tables, existing views or custom SQL queries establishing an ad-hoc view. Consequently, the user can choose an existing table or view or write a query.
4. **Subject Map Creation:** The original version of the editor requires the user to define in detail how subject URIs are generated, usually using a template. An `rdf:type` can optionally be assigned.
5. **Predicate/Object Maps Creation:** Finally, for the selected subject any number of predicates and objects can be added. The editor offers the full expressiveness of R2RML predicate/object maps including advanced options.
6. **Textual Representation:** Finally, a summary is displayed. It is still possible to step back from this point in order to modify the rule.

3.2 Extensions and Modifications

While following the most obvious approach for creating mapping rules in R2RML, the original version of the editor showed little flexibility to deviate from this one


Triples Map 'Artist' (Mapping Rule)

1) Source Data: Logical Table (Preview of Relational Table, View or SQL Query Result)

16 - 16 / 30 Show 1 rows (max. 1000) Filter

gid	name
6d087b1b-3e1a-46a2-be3d-3624fba547d8	The Rythm Club

Input data based on manually specified SQL query.

2) Target Data: Subject/Predicates/Objects (Ontology Instances and Triples)

- Subject Map (Type and URI Structure of Ontology Instances)

type: music artist (e.g., *:MusicArtist-id*) Edit

- Predicate/Object Maps (Properties for Newly Created Instances)

Pred *name*

Object Type: Literal Value
Source Data Column: name

Fig. 3: A Mapping Rule in the Editor

approach that we required to test different hypotheses that we will explain in detail in Section 4.

To overcome those limitations, we modified the editor so that the user was free to construct mapping rules in almost any order. To this end, switching from one step to another is now possible by simply clicking the "Edit" button in the relevant part of the rule. When a user does so, a summary of all other parts of the rule still remains visible, so it is always possible to cross-check for implications of changes with those other parts at a glance. Finally, wherever possible, we also hid complex and rarely used language features behind an "Advanced" button, so that for most regular tasks the UI would not be obfuscated with a large number of extra knobs and options.

Before these changes users could not browse away from any wizard step (say, the predicate/object step) to explore relevant ontology or database aspects. It was neither possible to start exploring the ontology or database and then directly add a mapping rule for a relevant aspect just found. Instead, the users always had to get back to the editor's start page, add all mapping details in the expected order and manually enter every detail they found while exploring when requested.

Figure 3 depicts a rule in the new version of the editor where a predicate/object map is currently being edited while the summary of both the logical table and subject map are still visible.

Furthermore, we coupled the editor with ontology and database exploration: users can now add new mapping rules right from viewing the details of an ontology class, predicate, database table or column. If they do so, the rule will be initialized from the context, e.g., by automatically adding a logical table or rdf:type to the newly generated rule. This offers different entry points for adding rules and thus supports different editing approaches.

3.3 Semi Automatic Match Suggestions

Besides general UI support and preview capabilities that form the main parts of our editor, another popular method for supporting users in editing mappings are *suggestions*. Those can come, for instance, in the shape of code auto-completion while editing rules or, more specifically in form of match suggestions.

As mapping rules are logically build on matches (or correspondences) between aspects in the relational schema and the ontology, respectively, such suggestions carry high potential in reducing the amount of work that users will have to invest to identify and name the corresponding aspects while writing rules. While suggestion quality is a key factor in the usefulness during mapping creation in general, the question how matches are created is orthogonal to our work since it can be carried out in a pre-processing step.

For this reason we extend the editor with an optional semi automatic suggestion mechanism that can be turned on and off for evaluation to study its impact. In our paper, we use the IncMap system [7] to generate suggestions in both directions (i.e., from database to ontology and vice versa).

4 User Study

We have designed our user study to shed light on the following *key questions* with regard to the hypothesis mentioned in Section 1:

1. How suitable are different editing approaches when creating mapping rules in general (i.e., starting with for *browsing the ontology or the database*? *Hypothesis:* In the general case, none of the main approaches outperforms the other.
2. Is some approach more suitable for users with a *different background* (i.e., different levels of expertise, background in databases vs. semantics technologies)? *Hypothesis:* The background of the user influences results.
3. Is there a approach that works better for *different task types*? *Hypothesis:* Task characteristics will have influence on which approach works better.
4. How much can be gained when *providing mapping suggestions* resulting from using semi-automatic matching tools for the different editing approaches? Does some approach gain more than others and could thus be more suitable whenever high-quality suggestions are available? *Hypothesis:* In general mapping suggestions will help users when constructing complex mapping rules (i.e., no one-to-one mappings). Since complex mapping rules in R2RML are only supported by constructing a SQL join query, we believe that the ontology-driven approach will benefit more since then SQL queries with joins can be suggested as a logical table which maps to a prior selected class of the ontology. The other way round complex mappings are only supported by constructing multiple rules (i.e., one for each artifact of an ontology).

4.1 Task Definitions

We have built our user study around the MusicBrainz database[5] and the Music Ontology [9]. This scenario is particularly appealing because some domain knowledge can then be taken for granted with any study participant without teaching her the basic classes and properties. Moreover, a series of hand-crafted mappings provided by the EUCLID project[6] already existed. We used those mappings as a basis to define relevant and realistic mapping tasks for the user

[5] http://musicbrainz.org/doc/MusicBrainz_Database/Schema
[6] http://www.euclid-project.eu

study and to make sure that our expectations towards the mapping semantics were reasonable.

For the study, we have defined three tasks around different concepts of the Music Ontology (i.e., artists, recordings, and tracks). Each task includes a similar number of rules that need be created whereas the rules cover different elements of the ontology (i.e., instances of classes and properties). Moreover, we have created the tasks such that each task comes with challenges of different complexity (e.g., defining SQL queries with and without joins for the mappings).

The high-level description of the three tasks of the user study is as follows:

1. **Artists:** We need to get some information about artists listed in the database: We need at least to be able to identify them uniquely as artists (typed) and know their names.
2. **Recordings:** We need to know about recordings listed in the database. At least, we need to have them uniquely identified and typed, and we also need to know their duration.
3. **Tracks:** We need to know about tracks listed in the database. At least, we need to have them uniquely identified and typed and we also need to know their position on the album.

We split each task in two steps: (1) The first step consisted of constructing a basic mapping between a source logical table and a target ontology class whereas the logical table had to be constructed by either choosing a plain table or writing a SQL query for joining multiple tables. (2) In the second step the user always had to add a predicate/object map in R2RML to map some attributes of a table to an appropriate ontology property. Table 1 lists all individual tasks of the user study, explains the two steps per task and discusses the associated problems that users had to solve in order to successfully complete the step.

4.2 Setup and Participants

To run the study, we extended the R2RML mapping editor [12] as described in Section 3 and provided a web front-end wrapping for the mapping editor in a specifically designed shell for the user study. Besides embedding the mapping editor and associated exploration and visualization features, the shell also implemented a questionnaire and a wizard-style series of briefing steps to prepare participants for the task. The editor implements the ontology approach as *ONTO mode* and the database approach as *DB mode*. For providing mapping suggestions, we used the IncMap system demonstrated in [8].

Participants could openly access the web front-end from the internet. Each participant was assigned an isolated study slot. Within each slot (i.e., for each participant) the user interface was restricted to provide only the functionality required for the part of the study to which the participant has progressed.

We recruited participants different technical background (general computer science, databases, Semantic Web) and experience (professionals and computer science students). Among those asked to participate were data integration professionals, colleagues and Semantic Web experts, as well as a group of second year computer science students.

Task	Step#	Short Description	Challenge/Non-trivial Aspects
Artists	1	Map and type instances of class *mo:MusicArtist.*	Identify the correct table in the database with more than 10 similar tables.
Artists	2	Construct *foaf:name* triples for artists	Identify the unique ID and the name attribute spread over two tables and write a SQL query for joining two tables.
Recordings	1	Map and type instances of class *mo:Recording*	Besides *mo:recording* the ontology contains other similar concepts (e.g., *recording_session*), which can be disambiguated only when carefully reading of the description.
Recordings	2	Construct *mo:duration* triples for recordings	In the relational database the *duration* property is called *length.*
Tracks	1	Map and type instances of class *mo:Track*	Identify the correct ID attribute while most attributes show purely numeric sample data.
Tracks	2	Construct *mo:track_number* triples for recordings	Task description mentions *position* on an album (like database attribute), but not *number* (as used in ontology)

Table 1: Study Tasks and Associated Challenges

4.3 Structure of the Study

We structured the study in four parts: (1) initial questionnaire, (2) introduction and technical briefing, (3) mapping tasks (the key part of the study), and (4) a catamnestic questionnaire.

Initial Questionnaire: We asked participants to rate their technical knowledge in the relevant fields (*relational databases and SQL*, and *RDF and ontologies*) to address *key question 2* (i.e., the influence of the background knowledge). Users could rate their skills on a scale from 1 to 5 (with 1 indicating extremely low and 5 indicating very high expertise).

Study Introduction: After that, we introduced users to the study and the mapping editor in a wizard-style introduction of six subsequent web pages. Users were introduced to R2RML mappings in general (mapping from databases to ontologies), the application domain (music) and the general problems of finding correspondences without in-depth knowledge of the schema and ontology. They were then familiarized with the mapping editor by showing and explaining an example mapping rule in screen shots.

Mapping Tasks: In the main part of the study, we asked users to perform the different mapping tasks previously described in Section 4.1. For each task, users can see the high-level description of the task's information need as well as more detailed instructions for the current step they are working on. From the task description page, users can follow a link to the editor's main page, which shows all existing mapping rules created so far as well as entry points to browse the database schema and/or the ontology.

To make sure that the users follow different exploration strategies (i.e. first browse the ontology and then the database and then the other way round), we varied the availability of entry points for browsing from task to task. This helped us to discuss *k*ey question 1, which analyzes the influence of different approaches on the mapping results. Therefore half of the users would, for their first task, only be able to browse and explore the ontology (not the database) to create associated mapping rules. Once a mapping rule was created, users could enter matching database information by using standard editor tools, i.e., lists of available database tables, data previews, as-you-type auto-completion and, possibly, automatic suggestions. For the second task, those users would then only be able to initially browse the database schema (not the ontology) to create mapping rules. Matching ontology aspects could then only be entered in the editor itself. For the other half of the users browsing and exploration went the other other way around (i.e., they could only access the database schema in the first task and only the ontology in the second). For the last task, all users were free to try either way.

Moreover, we presented the three tasks in random order to all participants in order to compensate for a potential bias introduced by the learning curve of getting familiar with the schema, ontology and mapping editor. This should help us to better analyze *k*ey question 3 (i.e., the influence of different task types).

Finally, to discuss *k*ey question 4, half of the users were provided with automated mapping suggestions, while the others were not.

While browsing, exploring or editing mapping rules, participants could always mark the current step as completed, which would advance them to the next step, again showing the task/step description with current information. Participants could as well skip a step, which would also advance them to the next step or task. We kept a record of whether user marked steps a completed or skipped in order to compare the correctness of the mappings to the participants' confidence in their correctness. Also, participants could always follow a link back to task and step instructions and double-check what to do before proceeding to edit their mapping rules.

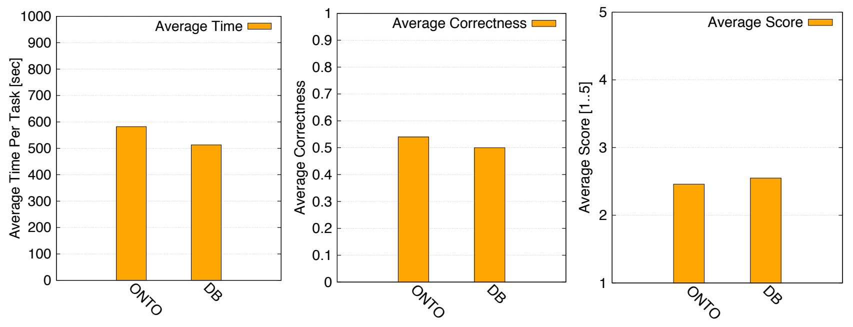
Catamnestic Questionnaire: After all tasks were completed (or skipped) we asked for each of the tasks how the user felt about solving them. While doing so, we reminded participants of the different browsing and exploration options they had in the different tasks to draw their attention to those different strategies. Participants could rate the options for each task on a scale from 1 to 5. This was to to inquire whether users would have a preference for one approach.

4.4 Study Results

From a total of 47 participants we considered 31 result sets for evaluation. The remaining 16 users quit the study during the briefing or during the first task and produced too little usable data. Out of all 31 participants considered, 13 ranked themselves as experts in ontologies, while 11 ranked themselves as experts in databases (skill level of 4 or 5 on a scale from 1 to 5).

General Findings: A first look mostly confirms our expectations.

Figure 4 shows a comparison between the two main editing approaches from different angles. Neither the average time that users needed to complete a task (Fig. 4a), nor the correctness of results produced with each approach (Fig. 4b)

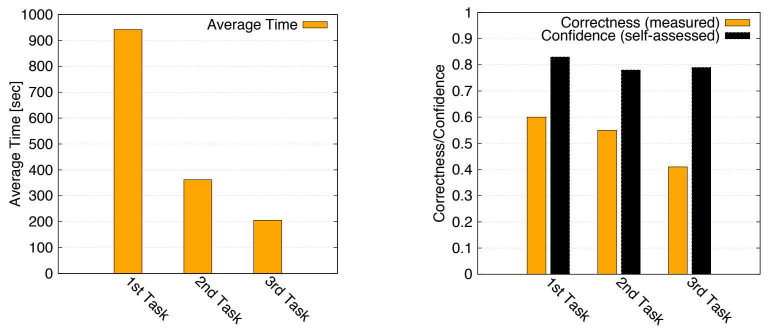


(a) Time Taken (b) Correctness (c) Average Rating

Fig. 4: Overall Per-approach Averages of Correctness, Time Taken, User Ratings

show significant differences. Our first hypothesis, namely that in the general case no approach outperforms the other, is therefore retained.

Also, participants felt about as comfortable with the ontology approach as they did with the database approach, as the catamnestic survey reveals (Fig. 4c). This is particularly interesting, as users overwhelmingly turned to the database-driven approach when given the choice in the last task. Additionally, we observed whether users produced more and smaller or fewer and larger mapping rules under different circumstances. We found that expert users tended to produce slightly fewer rules than non-experts.



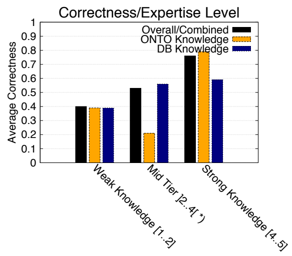
(a) Average Time (b) Correctness and User Confidence

Fig. 5: Influence of Task Number on Time Taken, Correctness, User Confidence

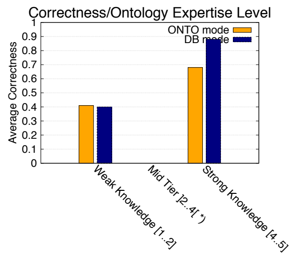
Little surprisingly, participants spent the longest time on the first task, worked faster on the second and again faster on the third (Fig. 5a). Somewhat less expected, however, the correctness of results also continuously declines with the task number (Fig. 5b). With result quality in mind this is a little unsettling because the correlation between correctness and the self-assessment of task success by the users is rather weak, as can also be seen in Figure 5b. On top of that, participants in general tend to overestimate the correctness of their mapping rules. This comes despite the fact that the editor offered preview data both for the relational source and for resulting target triples to allow for sanity checks.

Editing Approach per User Background: According to our second hypothesis, different approaches should work differently well for users with different background knowledge. As Figure 6 shows, this is in fact the case.

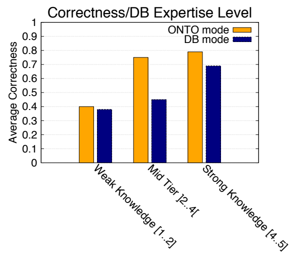
As was to be expected, users with a stronger background knowledge generally produce better results than those with poorer skill levels. Figure 6a shows the impact of different levels of background knowledge. Please note that, while the level of database skills was almost evenly distributed, only two users rated themselves into the middle tier of RDF and ontology skills; the drop of resulting correctness for mid-level ontology knowledge is dominated by an outlier.



(a) By User Expertise



(b) By Ontology Skills



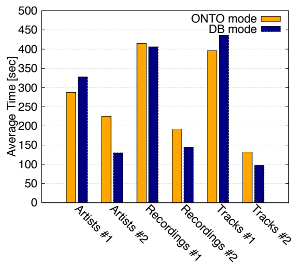
(c) By DB Skills

Fig. 6: Influence of Different Background Knowledge on Correctness

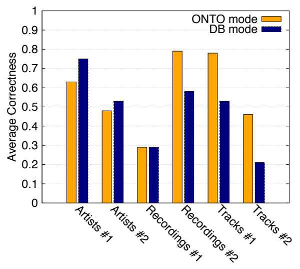
Interestingly, though, it is not the case that database experts work better with the database approach while ontology experts improve when working with the ontology approach, but the other way around. At second glance, however, this makes perfect sense: in the ontology mode, users have the opportunity to browse and explore the ontology first, then they need to identify the corresponding database table(s) in the editor, which offers less exploration and visualization possibilities (for database mode it is the other way around). Thus, users who are proficient in databases were more successful when the tougher part – finding a mapping partner – could be handled in the database world that they are familiar with, and vice-versa.

Browsing and Exploration Methodology per Task: As a third hypothesis we assumed that different approaches would work differently well on different task types and associated challenges.

Figure 7 clearly shows that this is the case, with each approach in the lead on correctness for some of the tasks. Observations on which approach works better for which task also largely match exceptions, when considering the specific challenges for each task and step described in Section 4.2. For instance, the second step of the Recording task (Recording#2) required to solve a lexical mismatch (i.e., duration in the ontology and task description vs. length in the database schema), which we expected to work better if users work in an ontology-driven approach since it is harder to pick an attribute in the database schema



(a) Average Time

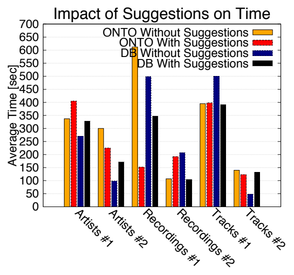


(b) Average Correctness

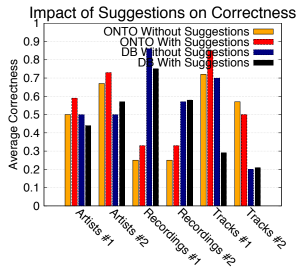
Fig. 7: Correctness and Time Taken per Task, Step, and Editing Approach

with a totally different name (i.e., length) without any context and assistance. For this task step the ontology-driven approach is on a clear lead in correctness.

Influence of Automatic Suggestions: Finally, our fourth hypothesis says that suggestions should have a stronger positive impact with the ontology approach. Figure 8 depicts the impact of automatic suggestions. We provided suggestions for logical tables in ontology mode, for ontology classes in database mode and for predicate/attribute correspondences in both cases.



(a) Average Time



(b) Average Correctness

Fig. 8: Influence of Automatic Suggestions for Different Strategies

Figure 8a shows that the presence of suggestions can save working time, though not in all cases. When analyzing the influence of suggestions on the correctness of R2RML mapping rules (Fig. 8b), our hypothesis is partially confirmed. As expected, the ontology-driven approach gains in most cases in correctness (i.e., in five out of six steps) when providing logical tables as mapping suggestions (since this involves writing potentially complex SQL queries). In particular in the artist task, where users had to manually write a SQL join, the gain of the appropriate logical table suggestion is noticeable high and also clearly puts the ontology mode in lead before the database mode for this task. For the database-driven approach, however, mappings suggestions have a negative impact for many task steps (i.e., in three out of six steps). We cannot really explain this observation for the database-driven approach. Instead, we speculate that some users deliberately have chosen absurd suggestions at the end of the study to finish their last tasks quicker, which would also explain the general drop in correctness for the later tasks in Figure 5b.

5 Related Work

Data integration is a well studied research problem [2]. However, studying different editing approaches for mapping construction that involve user interactions for data integration tasks has gained relatively less attention so far. Most research in the field of data integration has been focusing on automatic approaches for schema alignment, record linkage, data fusion etc. We believe that with the growth of schema complexity as well as with the increasing need to integrate more and more data sources, new interactive approaches for constructing complex mappings are getting necessary.

Some RDB2RDF editors exist that offer advanced and more or less visual user interfaces, e.g., [1,10]. However, these are based on domain specific languages predating R2RML. To the best of our knowledge, no other editors to date expose R2RML semantics through a visual interface. [11] describes an Eclipse plugin that supports R2RML execution as well as mapping generation with custom algorithms. Neto et al. have demonstrated a mapping editor with a highly visual interface that eventually generates R2RML [6]. However, they not expose R2RML semantics but only simple correspondences used as assertions. Arguably, the expressiveness of these assertions is only a subset of R2RML.

Only a few recent papers exist that include user studies that analyze approaches for user-centric data integration [13,3,4]. Both [3] and [13] contain user studies that analyze the effectiveness and efficiency of existing visual tools for data integration that follow a similar cognitive support model. This cognitive support model represents a very strict approach for creating mapping rules: first, a set of mapping rules is created automatically, then these rules are applied to some data and finally users verify the individual rules by marking the results as correct or incorrect. Compared to this very strict approach, [4] introduces a new interactive data transformation language that leaves much freedom to the user which approach the user actually will apply. Basically, the user can apply a set of data transformation primitives in any order and is supported by interactive data visualization tools to preview results, histories to undo changes, etc.

Our editor is in-between these two extremes and proposes two general approaches that support users to curate mapping rules by either selecting schema elements from the source schema or the target schema that are then mapped to the other side. These approaches are analyzed in a comprehensive user study with more than 31 usable data sets of 47 participants, which is a higher number than reported in the other user studies (which range from 4-22 participants).

6 Conclusions

We presented the results of a comprehensive user study that evaluates alternative mapping editing approaches (ontology-driven vs. database-driven) to construct

R2RML mapping rules in an editor. Consequently, we tested different hypotheses and measured the time and correctness for different mapping tasks.

We found out that neither approach is at a significant advantage of the other in the general case. However, we have seen that the ontology-driven approach works better for users with a background in databases and vice-versa, which was initially counter-intuitive for us. We also found a strong influence of task characteristics on the resulting mappings. Finally, it showed that automatic suggestions tend to have more impact on the ontology driven approach. It needs to be noted that, when given the choice, all users overwhelmingly tend to follow the database-driven approach, not the one that statistically works better for them.

As a result of our observations, we can make the following recommendations for building R2RML editors:

1. It is desirable to support both basic approaches, ontology-driven and database-driven, as each works better under different circumstances.
2. We cannot expect users to choose the best approach. Instead, an editor should try to learn about their background and, if possible, about the mapping tasks and then actively propose the adequate approach.
3. Prominent validation mechanisms should be offered, as users largely overestimate the quality of their mapping rules, even with data previews available.
4. When working with automatic match suggestions, the ontology-driven approach is somewhat more promising.

References

1. Bizer, C., Seaborne, A.: D2RQ-treating non-RDF databases as virtual RDF graphs. In: ISWC (2004)
2. Dong, X.L., Srivastava, D.: Big Data Integration. PVLDB 6(11), 1188–1189 (2013)
3. Falconer, S.M., Noy, N.F.: Interactive Techniques to Support Ontology Matching. In: Schema Matching and Mapping (2011)
4. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J.: Wrangler: interactive visual specification of data transformation scripts. In: CHI (2011)
5. Kharlamov, E. et al.: Optique 1.0: Semantic Access to Big Data. In: ISWC (Posters & Demos) (2013)
6. Neto, L.E.T. et al.: R2RML by Assertion: A Semi-automatic Tool for Generating Customised R2RML Mappings. In: ESWC (Satellite Events) (2013)
7. Pinkel, C., Binnig, C., Kharlamov, E., Haase, P.: IncMap: Pay-as-you-go Matching of Relational Schemata to OWL Ontologies. In: OM (2013)
8. Pinkel, C. et al.: Pay as you go Matching of Relational Schemata to OWL Ontologies with IncMap. In: ISWC (Posters & Demos) (2013)
9. Raimond, Y., Giasson, F., (eds): Music Ontology, www.musicontology.com (2012)
10. Rodriguez-Muro, M., Calvanese, D.: -ontop- framework (2012), `http://obda.inf.unibz.it/protege-plugin/`
11. Salas, P.E., Marx, E., Mera, A., Breitman, K.K.: RDB2RDF Plugin: Relational Databases to RDF plugin for Eclipse. In: TOPI (2011)
12. Sengupta, K., Haase, P., Schmidt, M., Hitzler, P.: Editing R2RML Mappings Made Easy. In: ISWC (Posters & Demos) (2013)
13. Stuckenschmidt, H., Noessner, J., Fallahi, F.: A Study in User-centric Data Integration. In: ICEIS (3) (2012)