Accepted (with major revisions) of: Audrey Sanctorum, Jonathan Riggio, Jan Maushagen, Sara Sepehri, Emma Arnesdotter, Mona Delagrange, Joery De Kock, Tamara Vanhaecke, Christophe Debruyne & Olga De Troyer (2022) End-user engineering of ontology-based knowledge bases, Behaviour & Information Technology, DOI: 10.1080/0144929X.2022.2092032

End-User Engineering of Ontology-Based Knowledge Bases*

Audrey Sanctorum[1 [0000−0002−4872−4687]], Jonathan Riggio[1 [0000−0001−7277−5356]], Jan Maushagen[1 [0000−0003−0197−2111]], Sara Sepehri[2 [0000−0003−1023−7987]], Emma Arnesdotter[2 [0000−0002−5891−5479]], Mona Delagrange[2 [0000−0002−8926−0444]], Joery De Kock[2 [0000−0002−4078−4896]], Tamara Vanhaecke[2 [0000−0002−6685−7299]], Christophe Debruyne[3 [0000−0003−4734−3847]], and Olga De Troyer[1 [0000−0002−8457−7143]]

[1] WISE Lab, Vrije Universiteit Brussel, Brussels, Belgium
[2] Research Group of *In Vitro* Toxicology and Dermato-Cosmetology (IVTD), Vrije Universiteit Brussel, Brussels, Belgium
[3] Data Representation and Engineering Group, Université de Liège, Liège, Belgium
{Audrey.Sanctorum, jonathan.riggio, jan.maushagen, sara.sepehri, emma.arnesdotter, Mona.Delagrange, Joery.de.kock, tamara.vanhaecke, Olga.DeTroyer}@vub.be, cdebruyne@uliege.be

Abstract. Knowledge bases store information on certain topics. Applying a well-structured and machine-readable format for a knowledge base is a prerequisite for any AI-based processing or reasoning. Semantic technologies (e.g., RDF) offer such a format and have the advantages that they make it possible to define the semantics of the information and support advanced querying. However, the disadvantage is that using such technologies is challenging for people not trained in IT, such as subject matter experts. This means that they need to rely on semantic technology experts to create, maintain, and query their knowledge bases. However, these experts are, in turn, not trained in the subject matter, while domain knowledge is essential for the construction of high-quality knowledge bases. In this paper, we present an end-user engineering approach for ontology-based knowledge bases. The goal is to allow subject matter experts to develop, maintain, and exploit the knowledge base themselves. We also present the supporting tools developed so far. The tools for the construction and the manual filling of the knowledge base are using the jigsaw metaphor to hide technicalities and guide the users. The end-user approach and the tools are demonstrated for building a knowledge base in the toxicology domain.

Keywords: Knowledge Base Engineering · End-User Development · Domain Ontology Development · Data lifting · Quality Assurance · Jigsaw Metaphor.

* Financially supported by Vrije Universiteit Brussel under Grant IRP19; and Cosmetics Europe and the European Chemical Industry Council (CEFIC).

1 Introduction

In many research domains like life-science, a huge amount of data and information is (publicly) available. This information may be needed for conducting new research, but may also be useful for consultation by practitioners in the field. However, in many domains, the information is often available in the form of documents and reports that may all have different formats and provide information of different levels of details. As a consequence, manual searching through the different documents and reports is needed when information is required. This is a time-consuming process and, in addition, it makes it hard to aggregate knowledge and reuse it in research.

Further, to stimulate data sharing and allow for intelligent data processing, there is an increasing interest in mechanisms that allow for a more uniform, flexible, and powerful way of storing data and information. Knowledge bases are suitable for this. A knowledge base can be used to collect and centralize information in a domain using a well-structured and machine-readable format. A knowledge base is different from a database in the sense that a database is mainly storing facts and its expressive power is limited. For instance, it is hard, if not impossible without additional tools, to deal with variability in data, to express the meaning of the data, or to perform intelligent reasoning over the data. Semantic technologies (e.g., RDF [16], OWL [3]), are more appropriate for storing knowledge, as they overcome the limitations of traditional database technology, i.e., information can be structured in a flexible way and the semantics of the information can be expressed, and they provide support for more advanced querying mechanisms and reasoning [24].

Knowledge bases can be constructed either manually or automatically. Work exists in the context of automated knowledge base construction, e.g., [2, 11, 14, 45]. However, the approaches proposed are usually domain-specific or cover only a certain aspect of the knowledge base creation, e.g., the automatic identification of concepts and their concept hierarchies, or populating a knowledge base from unstructured data. Therefore, the manual construction of the knowledge base is still a widespread practice.

Plain use of semantic technologies, such as RDF, is very challenging for those not ICT literate [46]. When applying semantic technologies for creating a knowledge base, usually tools such as Protégé [48] or OntoEdit [58] are used (see [55] for an overview of such tools and languages). These tools are quite technical. Consequently, subject matter experts, oftentimes not trained to use these tools, need to rely on semantic technology experts to construct and maintain their knowledge base. However, these experts are usually IT experts and they are, in turn, not trained in the subject matter, while the construction of knowledge bases requires extensive knowledge of the domain. The lack of knowledge about each other's domain results in a vast knowledge gap between the two groups of experts. In addition, each group of experts uses its own vocabulary and has its own concerns. Bridging this gap is a laborious and challenging process.

Different authors have studied and analyzed the gaps and barriers in interdisciplinary research [10, 40, 54] and proposed various approaches for bridging the

gap. For instance, in [40], the use of human translators or intermediaries, trained in both disciplines, is suggested to solve the communication problem in interdisciplinary collaboration. However, in the case of knowledge engineering, this only moves the problem of mastering two completely different disciplines from the IT expert or subject matter expert to the intermediaries. In the same publication, the use of technology to bridge collaboration barriers is mentioned. This is the approach we follow. Our goal is to provide subject matter experts tools that are easier to use than current semantic technology tools, and that hide the technicalities. This will facilitate constructing and maintaining knowledge bases without being completely dependent on semantic technology engineers.

This paper presents an end-user approach to engineering[4] domain-specific knowledge bases and a collection of tools developed to support this approach. The proposed end-user approach is based on the Abstract Reference Architecture (ARA), an existing framework for creating, maintaining and exploiting knowledge graphs [24]. Because we opt for an ontology-based knowledge base, which can be considered as a form of knowledge graph, we can apply this framework. Following that work, we distinguish three main activities: Knowledge Base Construction, Knowledge Storage, and Knowledge Consumption. We first explain these different phases and discuss whether and how end users can performed them. Next, we present the toolkit developed to support this end-user approach. Currently, the focus of the toolkit, called DIY-KR-KIT, is on providing support for the development of the knowledge base, including filling it with data. For this, we use the jigsaw metaphor, a metaphor that became popular with the programming language Scratch. The purpose of applying this metaphor is to hide the technicalities and terminology of the semantic technologies. Using this jigsaw metaphor, subject matter experts can create their own domain ontology, meaning that they can define the concepts and relationships used in their domain and needed to represent the available knowledge formally. Using the same metaphor, subject matter experts can also set up the knowledge base and fill it with data. The approach and tools are demonstrated to build a knowledge base in the toxicology domain. A first evaluation has been performed [53]. Furthermore, we present a tool to automatically import data from spreadsheets that were previously composed by subject matter experts to maintain their data. We also discuss directions to deal with quality assurance.

The paper is organized as follows: Section 2 explains the concept of an ontology-based knowledge base and the existing ARA framework for engineering ontology-based knowledge bases. Section 3 presents our end-user approach towards ontology-based knowledge base engineering, including a discussion on how the different activities can be made accessible for end users. In Section 4, the various tools developed so far are explained and demonstrated with a use case from the toxicology domain. Related work is discussed in Section 5. The paper ends with conclusions and future work (Section 6).

[4] The process of designing, building, and using a system

2 Background

In this section, we provide the background of the work. We start by explaining the concept of an ontology-based knowledge base. Thereafter, we discuss the engineering process of such a knowledge base.

2.1 Ontology-Based Knowledge Bases

To explain the concept of an ontology-based knowledge base, we will compare it with a classical database. A database organizes data according to a specific data schema, also called a data model. In this way, the database is an "instantiation" of the data model, as at each moment, the stored data can be considered as an instance of the data model. Similarly, a knowledge base (KB) also stores data (or rather, knowledge), but the data/knowledge is not stored according to a strict data schema but in general in the form of a knowledge graph, which is a collection of connected descriptions. An ontology can be used to specify which type of knowledge can be stored in the knowledge graph [33]. An ontology describes concepts in a domain and their properties, as well as relationships between the concepts and domain rules that apply to them. In this way, the ontology can be seen as the schema/model, and the knowledge stored can be considered as an instantiation of the ontology [33, 13]. Using an ontology to define the model of a knowledge base has the advantage of providing formal definitions of the knowledge that can be stored, its meaning, and possible restrictions on what can be stored. This not only provides an unambiguous description of the knowledge, it also allows humans, as well as computers, to process the information and infer new knowledge.

Note that in some applications, the instances of the concepts and relationships (i.e., the "real" data) are also considered as part of the ontology, removing the strict separation between model and data. However, we follow the approach proposed by Chasseray et al. in [13], where the distinction between model and data is kept: a knowledge base is composed of a *domain ontology* and an *instantiated ontology*. The domain ontology is used to specify the organizational structure of the knowledge base, and as the name indicates, the instantiated ontology is an instantiation of the domain ontology containing the actual instances (data). Chasseray et al. combine ontologies with the OMG's Model-Driven Engineering (MDE) approach [15] that defines four modeling levels: data level, model level, meta-model level, and meta-metamodel level. Following this MDE approach, the authors of [13] consider the domain ontology as an instantiation of an *upper ontology* that, in its turn, defines the concepts and relationships needed to define domain ontologies, and corresponds to the meta-model level from the MDE approach. Such an upper ontology contains general modeling concepts such as Concept, Relation, and Instance. We show this tree-level structure for an ontology-based knowledge base in Fig. 1.

We illustrate the structure of such an ontology-based knowledge base with a use case from the toxicology domain. We will use this use case throughout the

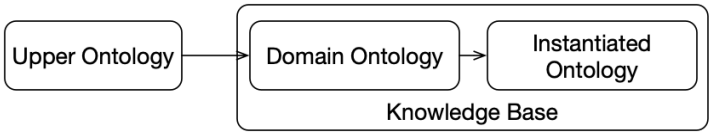

Fig. 1: Knowledge base structure based on the MDE approach (adapted from [13])

paper. In the toxicology domain, Safety Evaluation Opinions issued by the Scientific Committee on Consumer Safety (SCCS), provide collections of information on safety testing of cosmetic ingredients. These Safety Evaluation Opinions are text documents[5]. In order to create a knowledge base containing the information from such documents, and following the approach described above, first a domain ontology needs to be defined. That domain ontology will describe what type of information experts want to capture from these opinions in the knowledge base, e.g., *test species used* and *test reliability*, and how this information is related. Then the actual information from the opinions can be entered into the knowledge base, resulting into an instantiated ontology. The upper ontology is used to define the domain ontology for the safety evaluation opinions, i.e., the upper ontology should contain concepts and relationships (so-called meta-concepts, e.g., *Domain Concept*) needed to define the concepts and relationships required for specifying the knowledge contained in the opinions. For example, *Acute Toxicity* can be defined as an instance of *Domain Concept*.

2.2 Engineering Ontology-Based Knowledge Bases

For the engineering of ontology-based knowledge bases, we will follow (in Section 3) the process for creating, maintaining and exploiting knowledge graphs introduced in [24]. In that work, an Abstract Reference Architecture (ARA) is introduced to define the main phases and tasks required during the life cycle of knowledge graphs. Because an ontology-based knowledge base is one form of a knowledge graph, ARA is applicable for our purpose. ARA consists of three layers: *Knowledge Acquisition and Integration Layer*, *Knowledge Storage Layer*, and *Knowledge Consumption Layer*, which correspond to the three major tasks related to using a knowledge graph in organizations: construction, storage, and consumption. Each of these tasks can consist of sub-tasks:

1. The Knowledge Acquisition and Integration Layer deals with the knowledge life cycle, which consists, according to ARA, of the following tasks: *Ontology Development*, *Data Lifting*, *Data Annotation*, and *Quality Assurance*. These activities are shown in Fig. 2. From that figure, one can see that the outcome of each activity informs the other. For instance, when the schema evolves,

[5] Example Safety Evaluation Opinion: https://ec.europa.eu/health/scientific_committees/consumer_safety/docs/sccs_o_199.pdf

the transformation and integration of data into the knowledge base may need to be updated.
2. The Knowledge Storage Layer deals with the storage of the knowledge. Two main architectural options are mentioned in [24]: (1) reusing existing data storage and providing mappings between the ontology and the data schemes of the existing storage's; and (2) using a graph-based data store.
3. The Knowledge Consumption Layer provides tools for interested parties to access the knowledge. Examples include search and querying tools.

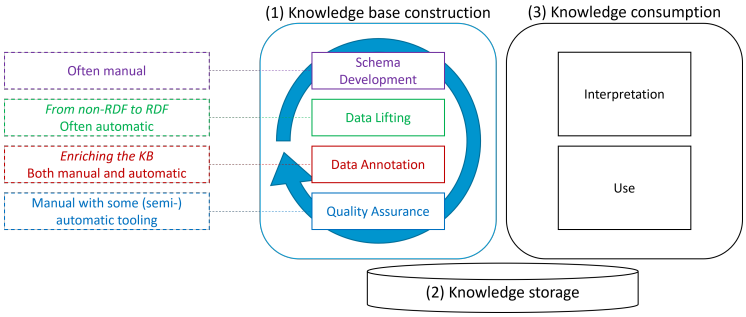

Fig. 2: The architecture of a knowledge base project depicting the various activities in knowledge base engineering, based on [24]

According to ARA, the first task to start with is the Ontology Development. Several methodologies defining a set of activities and techniques for developing an ontology have been suggested in the past, e.g., METHONTOLOGY [17], Diligent [51], HCOME [42], NeOn [57], and DOGMA [36], and dedicated tools have been developed, e.g., [34, 23]. As already mentioned in the introduction, ontologies are either created manually (by domain experts and/or knowledge engineers), or (semi-)automatically generated from non-ontological resources, such as classification schemes, thesauri, databases, XML, structured files, or unstructured documents. For unstructured documents, natural language processing may be needed to extract concepts and relationships automatically from the documents. It is also possible to learn (automatically or semi-automatically) domain definitions from existing data. This is called Ontology Learning [44].

In ARA, Data Lifting focuses on transforming raw data (e.g., stored in spreadsheets or classical databases) into semantic data, which involves converting the data to the appropriate format using the selected ontology. Data Annotation deals with linking and enriching the data with other relevant sources (e.g., other ontologies, knowledge bases, or even classical databases), resulting in interlinked and contextualized semantic data.

Finally, the Quality assurance phase in ARA is about ensuring that no mistakes are introduced into the knowledge base and its ontology. As mistakes can be introduced at multiple levels, different techniques need to be adopted to

detect and repair those. Quality assurance requires both domain expertise (to assess whether the knowledge base makes sense from an application domain's perspective) and knowledge base expertise (to assess whether there are no logical mistakes, for instance). Quality assurance can therefore be perceived as an interdisciplinary activity.

3 End-User Engineering of Ontology-Based Knowledge Bases

As already motivated in the introduction, direct use of semantic technology by subject matter experts who are not technologically skilled is often impossible. IT experts can be called in, but for specialized domains, like the domain of toxicology, it may take a long time before IT experts have familiarized themselves with the domain. In addition, the IT expert needs to stay available for the complete lifetime of the knowledge base as knowledge bases tend to evolve over time, e.g., new properties and concepts may be needed or new annotations may be required, and new data will be added. Furthermore, during the Knowledge Consumption phase, the assistance of IT experts may be needed, e.g., for the formulation of (new) queries or for (new) reasoning support. To avoid being largely or completely dependent on IT experts, we propose an end-user development approach for engineering ontology-based knowledge bases. According to [43], End-User Development (EUD) can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact.

Our proposed end-user development approach is based on the knowledge base engineering approach explained in Section 2.2 but targets subject matter experts who are not skilled in Computer Science as main developers. Fig. 3 provides an overview of the proposed end-user engineering process. Ideally, all tasks in the engineering process should be accessible to end users (i.e., subject matter experts); however, this seems not feasible for some tasks. In particular, tasks related to Data Lifting, Data Annotation, Quality Assurance, and Knowledge Storage may require IT experts' assistance, or could be handled (semi-)autonomously. In Fig. 3, three different icons indicate who can perform a task. If a subject matter expert can perform the task, the SM-expert icon is used; if the task is to be performed by an IT expert, the IT-expert icon is used; to indicate that the task can be done automatically, the service icon is used; and when a task can be done semi-automatically under the supervision of an IT expert, the IT-expert icon is combined with the service icon. When all three icons are mentioned, this means that some activities can be done by the subject matter expert and others semi-automatically or by an IT expert.

We elaborate on each of these activities in more detail in the following subsections. For subject matter expert activities, the main challenge is finding appropriate techniques to hide the technicalities of the Semantic Web technology and guide these users in the different tasks.

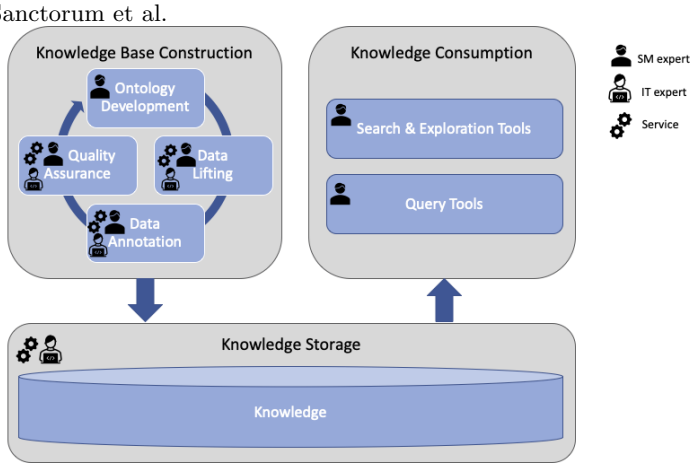

Fig. 3: End-User Engineering Process of Ontology-Based Knowledge Bases (based on [24])

3.1 Ontology Development

In the context of knowledge base engineering, Ontology Development is concerned with developing the knowledge base's schema. As already indicated, we follow [33] and [13] in the sense that a knowledge base is the combination of a schema and an instantiation of that schema (i.e., the data).

As the ontology describes (a part of) the domain of the end users, they need to be involved in some way. We already referred to ontology engineering methods involving domain experts, i.e., subject matter experts, and tools used by these stakeholders in Section 2.2. However, we can observe in other domains that learning how to use these tools can be quite difficult for end users [46]. In general, two approaches are used for creating an ontology, either domain experts contribute to the ontology creation, or knowledge engineers use the input of domain experts to create the ontology. The latter may require the use of knowledge representation techniques that are closer to the end user (e.g., a controlled natural language rather than logic-based formalisms).

We go one step further in our work and allow subject matter experts to create the ontology mainly by themselves. To make this possible, we use a metaphor to hide the technicalities of the semantic technology used. We decided to use the jigsaw metaphor [31], a metaphor that became popular with the programming language Scratch [52]. This metaphor is already used successfully in several domains (see Section 5). This section explains how end users can use the jigsaw metaphor to create a domain-specific ontology. In Section 4, we discuss the supporting tool and its implementation.

We first illustrate the principle of using the jigsaw metaphor for creating an ontology for the toxicological use case introduced in Section 2.1. The domain concepts that typically appear in the Safety Evaluation Opinions can be specified by means of jigsaw blocks (puzzle pieces) containing placeholders for property values and connection points for composing concepts; see Fig. 4 for an example block. This example block represents the domain concept *Acute Toxicity*. Its

main domain-specific property is *grading of lesion*. The two other properties, *additional information*, and *own comments* will be used to capture additional information provided in an opinion and comments that the subject matter experts want to add. Its composing concepts are *Test endpoints acute toxicity*, *Test method of acute toxicity* and *Reliability of test acute toxicity*.

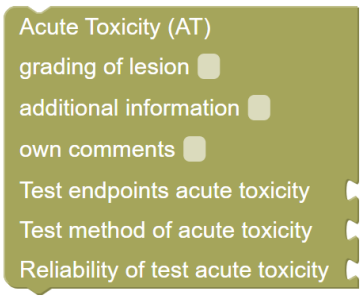

Fig. 4: Jigsaw Block for Domain Concept "Acute Toxicity"

Later on, when a subject matter expert wants to store the information of an opinion into the knowledge base, they compose a so-called *dossier* (representing the opinion) by connecting the relevant puzzle blocks and filling in the value fields in the blocks (see Section 3.2).

For defining these domain-specific jigsaw blocks, representing the domain concepts and their relationships in the ontology, the subject matter experts will use general jigsaw blocks representing the general modeling concepts (defined in the upper ontology). In Fig. 5, an example of such a general jigsaw block is given. This example block shows how to use a general jigsaw block to create the domain concept "Acute Toxicity" shown earlier on.

This ontology development process is illustrated in Fig. 6. The subject matter experts are using general jigsaw blocks to create the domain ontology, i.e., to define the concepts and their relationships in the specific domain. For each defined domain concept, a jigsaw block will be generated. These jigsaw blocks can then, in turn, be used later on by subject matter experts to compose and fill the knowledge base (see Section 3.2 and Fig. 8).

3.2 Data Lifting

Data Lifting comprises the activities related to filling the knowledge base. How this can be done depends on the data source. If the data is stored in some structured form, such as a database or spreadsheet, one can develop *mappings* to transform that data into the desired format. Those mappings can take on the form of dedicated software or be represented in a domain-specific language (e.g., R2RML [20]), which is interpreted by a processor that executes the transformations. If the data is captured in unstructured documents, manually entering the

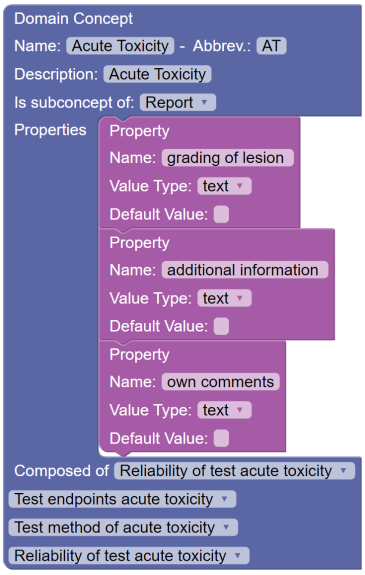

Fig. 5: Jigsaw Block for creating the Domain Concept "Acute Toxicity"

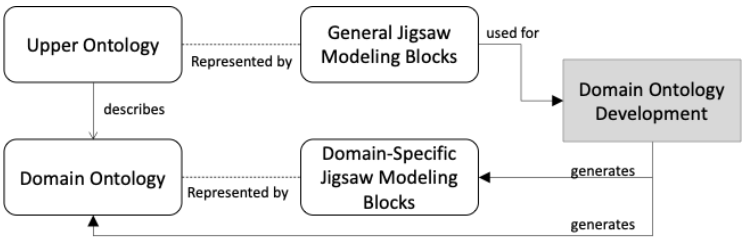

Fig. 6: Ontology Development Process

data is often the only option. Below, we discuss the possibilities in the context of an EUD approach.

Manual Data Lifting. To allow subject matter expert to enter their data manually into the knowledge base, we propose to use the same metaphor as for ontology development, i.e., the jigsaw metaphor. Again, we illustrate the principle of using the jigsaw metaphor for filling a knowledge base with the toxicological use case. When a subject matter expert wants to enter the information from a Safety Evaluation Opinion, they use the domain-specific jigsaw blocks generated during the Ontology Development Process (see Section 3.1) to compose a so-called *dossier* (representing the opinion) by connecting the relevant puzzle blocks and filling in the value fields in the blocks (see Fig. 7 for a (partial) example dossier). The jigsaw blocks can only be composed in a restricted way and validation for data fields can be provided. When the ontology is created with the terminology of the Safety Evaluation Opinions in mind, the names of the blocks and fields correspond with this terminology, and the subject matter experts can fill the knowledge base while reading the opinions. Based on the puzzle composition and its values, RDF can be generated that forms (a part of) the instantiated ontology. This data entering process is illustrated in Fig. 8. It is similar to the Ontology Development Process but the subject matter expert is now using domain-specific jigsaw blocks to instantiate the knowledge base.

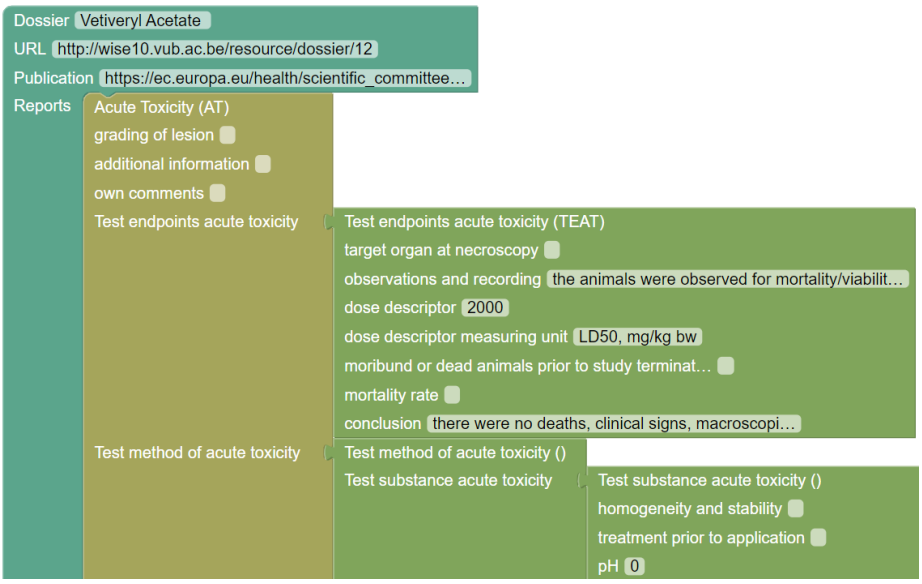

Fig. 7: Example Jigsaw Block for the dossier "Vetiveryl Acetate"

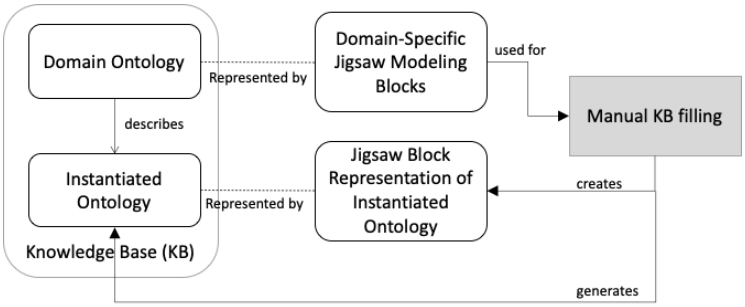

Fig. 8: Manual Knowledge Base Filling Process

Automatic Data Lifting. When data sources are structured, one can use mappings to transform data from non-RDF into RDF (a standard model for data interchange on the Web). One can either write tailor-made software (e.g., Python scripts) to transform the data or use domain-specific languages such as R2RML (a W3C Recommendation for transforming relational data into RDF) or RML [25], a non-standardized superset of R2RML that includes support for XML and JSON. There are quite a few tools that help one to transform data into RDF[6].

3.3 Data Annotation

Data Annotation comprises the activities of linking concepts and data with other relevant sources (e.g., other ontologies, knowledge bases, vocabularies, or even classical databases), resulting in interlinked semantic data. Similar to Data Lifting, some data annotation activities can or should be done manually, while tools can also be used for (semi-) automatic data annotation. In [24], different types of (semi-)automatic data annotation and examples of tools are discussed.

3.4 Knowledge Storage

In the context of end-user knowledge base engineering, the end users should not be concerned with knowledge storage. Given the approaches used for the Ontology Development and the Data Lifting, the resulting knowledge base storage (e.g., in the form of a triple store) can be automatically generated. Possibly, some intervention may be needed from an IT expert to deal with server issues.

3.5 Quality Assurance

Quality assurance is about ensuring that no mistakes are introduced into the knowledge base and its ontology. It is important to have high-quality knowledge bases, as one cannot rely on wrong or inconsistent data. In this context, two

[6] We refer to https://www.w3.org/wiki/ConverterToRdf for a non-exhaustive list.

different questions can be considered [59]: (1) has the knowledge base been built correctly, i.e., according to the requirements, and (2) has the right knowledge base been built, i.e., does the ontology correctly reflects the domain and does it contain correct data? Finding an answer to the first question corresponds with the notion of knowledge base *verification*, and finding an answer to the second one with the notion of knowledge base *validation* [30].

Assessing whether the knowledge base has been built correctly is arguably easier than assessing whether the right ontology has been built. For instance, one can quickly check whether a knowledge base contains contradictions. Evaluating whether the correct knowledge base has been built is much harder. One way to detect problems is to let subject matter experts use the knowledge base for a while. In [59], a comprehensive survey of the various dimensions of quality assurance with pointers to state of the art is provided. Important to note is that only a few of these dimensions can be assessed automatically. Sometimes dedicated software can be developed but often human input will be needed (i.e., semi-automatic approaches), usually from IT experts.

Furthermore, validating whether parts of the knowledge base adhere to the structure defined for it, is challenging to achieve with ontology languages due to the Open World Assumption (OWA)[7]. Validating the data in a knowledge base using ontology languages would not scale well; it would require the definition of many rules (e.g., to uniquely identify experiments mentioned in reports) and assertions (e.g., to explicitly state each pair of experiments in the knowledge base is disjoint).

An alternative is to use the Shapes Constraint Language (SHACL) [41]. With SHACL, a W3C Recommendation, one can validate RDF graphs. I.e., one can validate the structure of triples in a Closed World Setting, with little to no utilization of an ontology language's capability[8]. SHACL provides a set of "core" constructs for declaring rules (value- and data type checking, cardinality, value ranges, comparisons,... which can be combined with a set of logical operators). While the jigsaw metaphor can guide end users in entering information that is valid with respect to the jigsaw blocks, validating the knowledge base with SHACL is still valuable due to the fact that the integration of data via other means is often also possible or needed (see Section 3.2). One of SHACL's key concepts is the notion of a *shape*. A shape is a subset of an RDF graph, which can be declared, and to which one can add constraints. Shapes can be defined for both entities[9] (called node shapes) and properties (called property shapes).

One advantage of SHACL and RDF is that SHACL reports what errors RDF graphs contain and which entities violate the constraints. It does so by using the entities' Internationalized Resource Identifier (IRI). One can apply SHACL not only to validate the contents of a knowledge base, but also the validate data

[7] While an oversimplification, it suffices to state, for this article, that in an OWA setting, information that is not known is not necessarily false.

[8] Most often limited to inferring triples with a reasoning engine before applying SHACL.

[9] In a RDF graph, entities refer to the nodes in the graph.

prior to integration. Tools such as TopBraid's SHACL API[10] and PySHACL[11] can be easily integrated into a workflow allowing end users to identify and spot problems. The messages generated by SHACL are comprehensive enough for them: e.g., "123" is not of the type `xsd:integer`. SHACL's disadvantage is that it is not suitable for direct use by end users, i.e., we can not assume that end user can write SHACL-code themselves to validate a knowledge base. Therefore, either knowledge base specific applications need to be developed (by IT experts) or a method to generate such data validation patterns automatically is needed. In our tool support, we have adopted this second option (see Section 4.4).

3.6 Knowledge Consumption

The Knowledge Consumption refers to the activities dedicated to the knowledge base's use (or consumption). In an end-user approach, we should provide tools usable by people not schooled in IT [50]; we cannot expect that these end users are capable to (learn to) formulate SPARQL, for instance. For visualizing and browsing through the knowledge base, tools can be developed or existing tools (e.g., [6]) can be used if they are suitable for end users. However, when the subject matter expert needs precise answers to specific questions, the use of a dedicated query language may be required. Predefined queries can be provided by IT experts but not all possible queries can be foreseen and this makes the subject matter experts again dependent on the availability of IT experts. In that case, the challenge is to provide a layer on top of the query language to hide the technicalities of the language for the end user. See [56] for an elaborated discussion on this.

4 Tool Support: DIY-KR-KIT

This section explains the toolkit developed to support the end-user engineering of ontology-based knowledge bases. The toolkit, called DIY-KR-KIT (Do It Yourself Knowledge Representation Kit), currently provides support for the following tasks: Ontology Development, Data Lifting, Data Annotation, Quality Assurance, and Knowledge Storage. Support for Knowledge Consumption is under development. In the following sections, we describe the support for the different tasks and the status and limitations of the different tools.

4.1 Ontology Development Support

As mentioned before, Ontology Development uses the jigsaw metaphor. The tool is based on the tool described in [22]. In that tool, subject matter experts still had to rely on IT experts to define the domain ontology. Thus, that tool did not support end-user ontology development. Subject matter experts could only

[10] https://github.com/TopQuadrant/shacl
[11] https://github.com/RDFLib/pySHACL

use the tool to compose and fill the knowledge base by means of the domain concepts defined in the ontology (i.e., manual data lifting). We extended and adapted the tool in such a way that subject matter experts can also use it to create the domain ontology. A first version of this new tool was described in [53]. In the meantime and based on the user study performed (also described in [53]), the tool has been improved and functionality has been added.

The tool is a web-based application, built on top of Apache Jena, which provides the triplestore and SPARQL endpoint. The jigsaw metaphor is implemented via the Google Blockly JavaScript library.

For the ontology creation, the end user has to define the domain concepts. As described in Section 3.1 and illustrated in Fig. 5, for adding such a concept, a special block is provided that allows entering the name of the concept, an optional abbreviation and description of the concept, its properties, its sub-components, and optionally a link (URL) referring to a source providing information about the concept (such as a definition or explanation - see also Section 4.3). An already defined domain concept can also be adapted in this way.

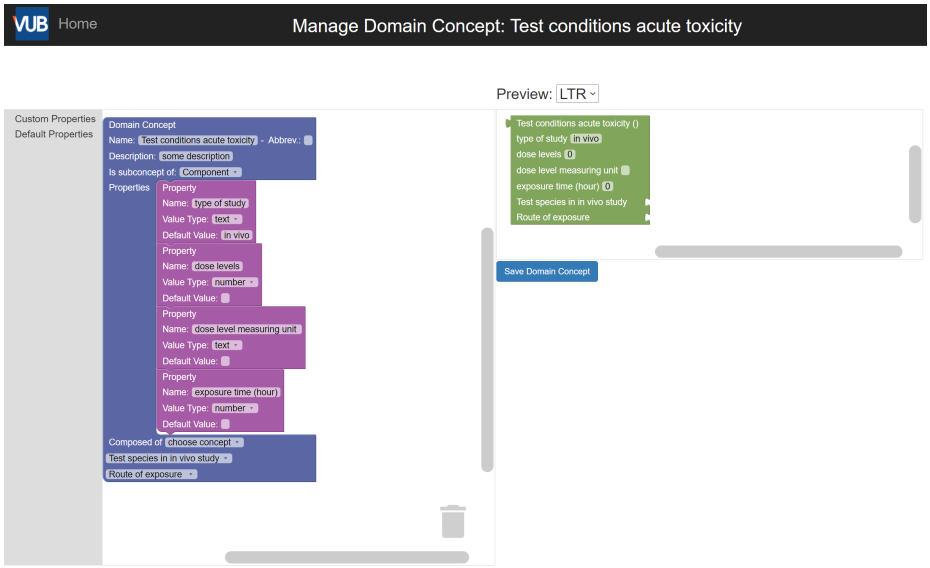

Fig. 9: Screenshot of the domain concept definition page

Fig. 9 shows the definition page for the domain concept *Test conditions acute toxicity*. On the left-hand-side of the page, one can see the top-level blue block "Domain Concept" that allows specifying the structure of the domain concept as previously described. Properties can be added to a domain concept block by dragging and dropping the empty *Property* block from the *Custom Properties* tab in the menu at the left. Properties have a name, a value type, and an optional

default value. In the example, the first property of the domain concept is given the name "type of study" with value type "text" and default value "in vivo" . Note that in the left sidebar menu we also have the *Default Properties* tab, which provides recurring property blocks, such as the "year" property block. By using these default properties blocks, users can save time.

A domain concept can be composed of other domain concepts. In Fig. 9, we see that the *Test conditions acute toxicity* concept is further composed of the *Test species in in vivo study* and the *Route of exposure* concept. This is shown in the blue Domain Concept block under the "Composed of" field. The "Composed of" dropdown allows adding other composing concepts.

On the right in Fig. 9, a preview of the generated jigsaw block for the domain concept defined at the left is given. In this case, the block contains two puzzle connectors on the right side, one for each composing concept given. Any changes made on the left side are reflected in real-time in the preview on the right.

Blockly defines blocks in XML format. However, when a domain concept is saved (using the "Save Domain Concept" button), this XML representation is transformed by our tool into RDF using an XSLT file so that it can be integrated into the domain ontology. Once the concept is created, its name will appear in the menu with the domain concepts.

4.2 Data Lifting Support

Our toolkit supports manual as well as some form of automatic data lifting:

Manual Data Lifting Support. As explained in Section 3.2, the same metaphor as for the ontology development, i.e., the jigsaw metaphor, is used for filling the knowledge base with data. For example, when a subject matter expert wants to enter the information from a particular Safety Evaluation Opinion, they use the domain-specific jigsaw blocks created during the Ontology Development Process (see Section 4.1) to compose a so-called *dossier* (representing the opinion) by connecting the relevant puzzle blocks and filling in the value fields in the blocks (see Fig. 7 for a (partial) example dossier). The jigsaw blocks can only be composed in a restricted way and validation for data fields is provided.

From the user study performed (see [53]), it became clear that the manual filling of the knowledge base was rather time consuming. Several actions had to be repeated, such as defining some recurring blocks, which were needed to structure a dossier. In order to save time and also to ensure that those blocks are always defined in the same way, we now allow the end user to save such recurring block structures so that they can be reused in multiple dossiers as is. The Acute Toxicity block shown in Fig. 7 could, for example, be saved as a block structure. Then this block together with all its sub-components (i.e., blocks attached to its right) will appear in the tab "saved block structures" depicted in Fig. 10. These saved block structures can be dragged and dropped as a whole, which saves time for subject matter experts.

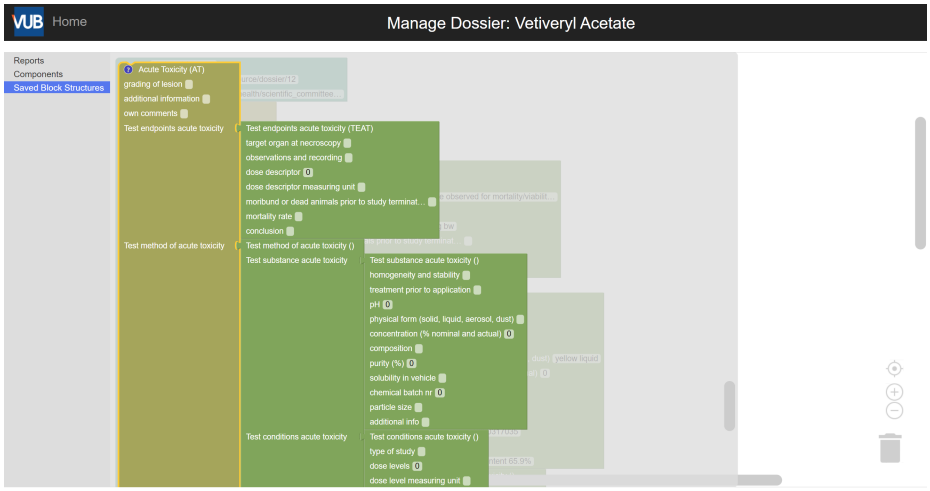

Fig. 10: Screenshot of the dossier creation page's saved block structure tab

Automatic Data Lifting Support. In the past, the subject matter experts from our toxicology use case used spreadsheets to structure and store the information from Safety Evaluation Opinions. Although the structuring of the information in the spreadsheets was very ad hoc, mostly based on a particular study or experiment, and limited, we decide to develop a tool to import the data into the knowledge base, as they spent a considerable amount of time in creating these spreadsheets. The tool works automatically given that the spreadsheet has a valid structure. However, we noticed recurrent encoding mistakes in some spreadsheets, such as merged cells that were not supposed to be merged or misplaced nodes in hierarchical tree structures drawn with cells. It therefore still requires to check manually that the spreadsheets are well-formed before parsing them. However, some errors are caught by the approach, which we will explain below. Automation of this process would be possible in a future version of the tool but would require reasoning capabilities in order to infer accurate corrections, similar to the solutions presented in [12] and [1].

We use R2RML to transform the spreadsheets into RDF. R2RML is a W3C Recommendation for transforming relational data into RDF. The spreadsheets are not relational databases, but each spreadsheet, once stored as comma separated values (CSV) file, can be considered as containing relational data (i.e., rows with attributes). R2RML engines (and dialects) such as RML [25] and R2RML-F [21] provide support for CSV files. We have chosen to adopt R2RML-F as this particular engine loads the CSV files into an in-memory relational database, which allows us to manipulate the data in the records with SQL prior to generating RDF.

As for data validation, R2RML-F produces RDF, but –as per R2RML specification– is not responsible for data validation such as checking data types.

Checking whether data types are correct are left to the graph database. Graph databases such as Apache Jena Fuseki (see Section 3.4) provide warnings when data types are invalid. Within our approach, we also avail of shape expressions (see Section 3.5) to see whether the generated RDF adheres to the structure of our domain ontology.

In Fig. 11, we show some of the RDF generated from the spreadsheets, in a visual manner with Ontodia [47]. Ontodia is a tool allowing one to visually explore a triple-based knowledge base using diagrams and faceted browsing. The RDF we show in this figure is, for sake of simplicity, limited to the concept of repeated toxicity. On the left, we have a list of classes and a list of instances tied to that class. We have dragged and dropped the report on "phenoxythenol" and started exploring by expanding its relationships. We see that there are 8 tests (w.r.t. repeated toxicity), 6 of which are OECD compliant tests and two that are not. We have shown the list of attributes of two tests. We can also navigate via other domain concepts, such as the species used in the tests. These visualizations can be used by the subject matter experts and by IT experts to validate whether the transformation was correctly performed, but also for quality assurance (see Section 4.4).

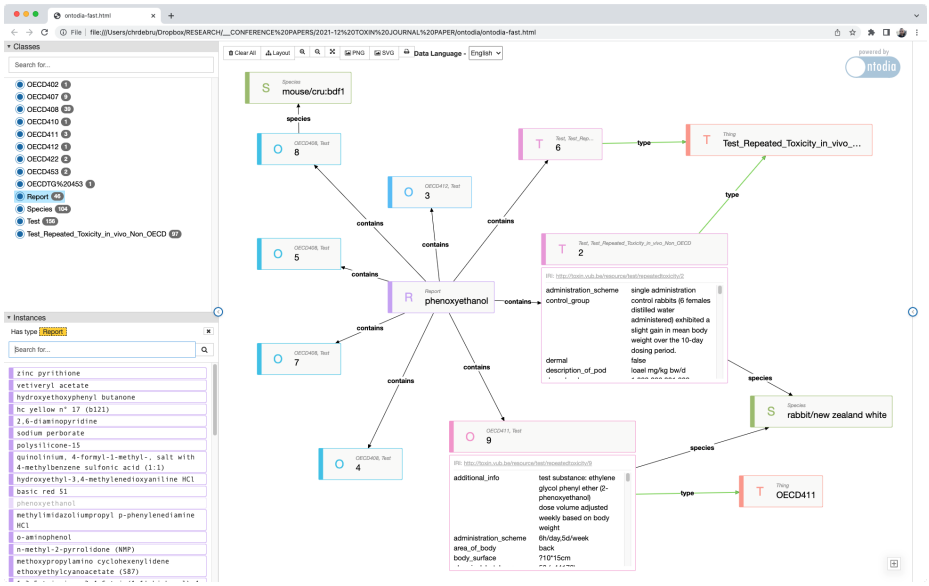

Fig. 11: Visually exploring the RDF generated from a spreadsheet with Ontodia.

4.3 Data Annotation Support

Currently, our own tool only supports manual data annotation. While defining the ontology, the subject matter expert can link domain concepts and data to

existing sources. While defining the ontology, the subject matter expert can link domain concepts and data to existing sources. Therefore, when creating a dossier or a domain concept, the subject matter expert has the possibility to add a link. For a dossier, this is a link to the Safety Evaluation Opinion file; for a domain concept it is a IRI (see Fig. 12). References to files indeed allow a user to consult the (original) sources from which the data is gathered. When they provide references to other ontologies and/or RDF datasets, the end users are effectively creating Linked Data [7]. Linked Data is an initiative in which one published RDF data according to specific best practices that result in interconnected data stored on different servers; a Web of data.

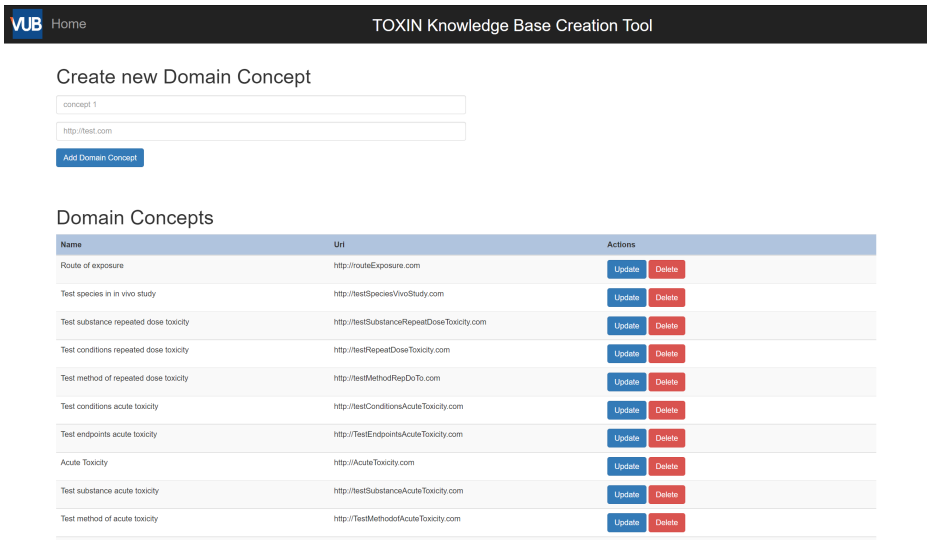

Fig. 12: Homepage for domain concept creation and modification

4.4 Quality Assurance Support

Currently, the support for quality assurance is limited in our tool to the validation of the structure of the data in the knowledge base by means of SHACL shapes (see Section 3.5 for SHACL). Because we cannot expect that our subject matter experts are able to write SHACL-code, we looked at an approach by which the SHACL-shapes and constraints can be generated.

Note that in our approach, the knowledge base also contains the domain ontology, which not only defines the domain concepts, but also the way the data should be structured in the knowledge base. By analyzing this information, one can infer what predicates are allowed to appear for particular entities in the RDF graph. Thus, shapes can be generated in a declarative manner using SPARQL

```
PREFIX toxin: <http://ontologies.vub.be/oecd#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT {
  ?nodeshape
    a sh:NodeShape ;
    sh:targetClass ?s ;
    sh:property ?propertyshape ;
    sh:closed true ;
  .
}
WHERE {
  ?s rdfs:subClassOf toxin:Test .
  ?s toxin:attributeGroup*/toxin:attribute/toxin:predicate ?p .
  BIND( IRI(CONCAT(STR(?s), "Shape")) AS ?nodeshape )
  BIND( IRI(CONCAT(STR(?p), "Shape")) AS ?propertyshape )
}
```

Listing 1: Creating node shapes for each type of "test" in the TOXIN knowledge base

CONSTRUCT queries. For example, and for the knowledge base of our use case, the query in Listing 1 is responsible for generating the node shapes for each type of *test*[12] and linking each type of test with its *test* as defined by the domain ontology. The WHERE clause of Listing 1 looks for the different types of *tests* and their properties. These results are then used to create IRIs for both node and property shapes. In the CONSTRUCT clause, one can see how one shape for each type of *test* is created. The node shape is "closed" , which means that each entity of that type should not use properties beyond those that are declared in their property shapes. We can provide a list of exceptions, as shown in Listing 3. As `rdf:type` is not mentioned in a property shape, we have decided to ignore that. We could have created a property shape for `rdf:type` in Listing 1, but this approach is more elegant and more commonplace. In Listing 2, we finally generate property shapes for each property used. The query also inspects the ontology to find the data type of that property. Listing 4 provides an example of a node shape and of a property shape.

Using this approach, we can generate data validation patterns for the main concepts of the knowledge base. However, note that the validation that can be done in this way is limited. Examples of rules that cannot be generated in this way are either domain-specific rules (e.g., the accepted values for a particular property is depending on the type of test) or more generic rules (e.g., the relationship between start- and end dates). Notwithstanding this limitation, the

[12] *test* is a domain concept

```
# namespaces omitted for brevity
CONSTRUCT {
  ?propertyshape
    a sh:PropertyShape ;
    sh:path ?p ;
    sh:datatype ?r ;
  .
}
WHERE {
  [] toxin:attributeGroup*/toxin:attribute/toxin:predicate ?p .
  ?p rdfs:range ?r .
  FILTER REGEX(STR(?r), "http://www.w3.org/2001/XMLSchema#") .
  BIND( IRI(CONCAT(STR(?r), "Shape")) AS ?propertyshape )
}
```

Listing 2: Creating property shapes for each property used in blocks.

```
# namespaces omitted for brevity
CONSTRUCT {
  ?nodeshape sh:ignoredProperties ( rdf:type ) .
}
WHERE {
  ?s rdfs:subClassOf toxin:Test .
  BIND( IRI(CONCAT(STR(?s), "Shape")) AS ?nodeshape )
}
```

Listing 3: Node shapes are closed; entities belonging to this node shape may not use any properties beyond their property shapes. Here, however, we can declare a list of properties that may be ignored such as `rdf:type`.

```
toxin:Test_OECD_489Shape a shacl:NodeShape ;
    shacl:closed true ;
    shacl:ignoredProperties ( rdf:type ) ;
    shacl:property toxin:GLPShape,
        toxin:Ref_in_dossierShape,
        toxin:SCCS_comment_to_testShape,
        # omitted for brevity
        toxin:vehicleShape,
        toxin:yearShape ;
    shacl:targetClass toxin:Test_OECD_489 .

xsd:angleShape a shacl:PropertyShape ;
    shacl:datatype xsd:angle ;
    shacl:path toxin:Angle_new .
```

Listing 4: SHACL shapes resulting after the application of SPARQL CONSTRUCT queries in Listings 1, 2, and 3.

SHACL that is generated checks whether the data types are correct and whether the *tests* only use predicates that are stored in the ontology.

By combining the automatic data lifting with SHACL, we effectively create an ETL (**E**xtract, **T**ransform, and **L**oad) pipeline with *some* quality assurance support.

4.5 Knowledge Storage Support

Both the Blockly-tool for the ontology development and for the manual data lifting, automatically transform the XML representation used by Blockly into RDF using an XSLT file so that it can be integrated into the domain ontology.

While there are quite a few triplestores available (both free, commercial, and free for research purposes), we have adopted Apache Fuseki[13] for managing the triplestores and SPARQL endpoints. The ontology is stored in one named graph. The data which has been lifted are stored in another. This allows us to separate, at the level of the data, the ontology and the data.

5 Related Work

In this section, we discuss related work in the context of end-user development for ontology-based knowledge bases. Thereby, we focus on the aspects that we elaborated on in the paper: end-user approaches for the development of ontology-based KB development in general, end-user ontology development, and the use of the jigsaw metaphor to hide technicalities from end users.

5.1 End-User Approaches for Ontology-Based KB Development

In [28], the authors propose a framework, called DaCura, that provides a process and tools to harvest and curate Linked Datasets (encoded as an RDF graph and stored in a triple store). They claim that to ensure high-quality of the published data, there must be a provision for subject matter experts to review, assess, and correct harvested data. Therefore, subject matter experts (called Domain Experts in DaCura) also have a role in the overall process. The authors also recognize that producing high-quality datasets requires a variety of automated and manual processing tools. Compared to our work, the role of the subject matter expert is limited to interpreting reports created by data harvesters and the generation of facts from these reports using a service, which are then published for consumers. Note that defining how the facts are represented is done by data architects. The data architects are clearly IT-skilled persons, but the profile of a data harvester is not explicitly defined in the paper. In any case, in that approach, the role of the subject matter expert is limited to what we call manual data lifting and some aspects of quality assurance.

KawaWiki, described in [39], enables end users to provide Semantic Web descriptions and annotation on Wiki content. For this, RDF templates are used

[13] https://jena.apache.org/documentation/fuseki2/

to avoid that end users need to learn the RDF syntax. It allows end users to generate RDF data through simple forms in a browser, and, in addition, provides some validation checks. Also in this approach, the role of subject matter experts is limited to entering data, as the templates are defined by expert RDF users.

In [27], an end-user development approach based on model-driven development, visual programming, and wizard form-filling is proposed to develop intelligent systems for supporting the search and troubleshooting onboard aircrafts. The approach is rather domain-specific as it is based on the use of event trees for formalizing scenarios for problem situations. For, the knowledge base itself, Personal Knowledge Base Designer (PKBD) [60] is used but this is not an end-user tool.

5.2 End-User Ontology Development

In [9], it is proposed to derive an ontology from a conceptual map (Cmaps). Conceptual maps express concepts and their relationships in the form of concept-relation-concept. A set of heuristic rules to map a conceptual map into an OWL ontology is presented. A distinction is made between classification relations, composition relations, bidirectional relations, and other relations. A first implementation of the translation system was made using Prolog, but no evaluation with end users was reported. It remains an open question whether conceptual maps are easier to use for subject matter experts than for instance a graphical representation of RDF.

Some papers (e.g., [4]) propose to transform UML class diagrams into OWL. However, we are not convinced that UML class diagrams are suitable for people without a computer science background, as the ease to learn and understand them has already been questioned for computer analysts [26].

Another direction is the use of a natural language interface. For instance, GINO (Guided Input Natural language Ontology editor) [5] is using a natural language approach. However, to avoid the limitations of full natural language interfaces, a guided and controlled language akin to English is used. To add a new construct to the ontology, the user should start by typing "there is" or "there exists" after which a popup shows possible constructs, such as "class". After having selected the appropriate construct, the user is prompted to give a label and ends the sentence with a full stop. Next, this sentence, e.g., "there is a class Lake." , is translated into OWL triples and loaded into the ontology. Properties are also defined in this way. GINO has been evaluated for usability by six users without experience in ontology building and with no computer science background. The participants had to create one class, one subclass, one data type property, one object property. The participants also had to add one instance with values for two properties and change the value of a property. An average SUS score of 70,83 was obtained. However, it must be noted that the task was very small. Defining a large ontology in this way may be very time-consuming as the interface is quite verbose. Other works that follow a similar approach are CLOnE (Controlled Language for Ontology Editing) [29] which allows multiple

classes to be expressed in a single sentence, and Rabbit [32] which language is also somewhat richer than GINO's.

5.3 Jigsaw Metaphor

The jigsaw metaphor has been used in the Semantic Web community for the creation of Linked Data mappings [38] and the formulation of SPARQL queries [8]. Junior et al. [37] report on an experiment that indicates that users achieved higher performance and had a lower perceived mental workload when creating Linked Data mappings using the jigsaw metaphor.

Recently, Öztürk and Özacar [49] propose a block-based approach, based on Blockly, for instantiating a recipe ontology. Their approach is similar to the manual data lifting in ours as they use the blocks to populate the ontology. However, the blocks are predefined, which has the disadvantage that the blocks need to be adapted when the ontology evolves, and new blocks have to be programmed for each new ontology. We overcome this problem by rendering the necessary blocks from the meta level in which the end user defines the necessary blocks. The system proposed in [49] was evaluated for usability with 14 participants (students), however more than half of them had a background in ontology engineering.

Next to the use of the jigsaw metaphor for the programming language Scratch, intended for children between the ages of 8 and 16, the metaphor has also been used to support end-user development in other domains, e.g., for the development of IoT [35]; for the development of mobile applications [19]; and for debugging IF-THEN rules in an IoT context [18].

6 Conclusion and Future Work

To support subject matter experts in the creation and exploitation (i.e., engineering) of ontology-based knowledge bases, we proposed an end-user development approach. We argue for such an approach because the direct use of Semantic Web technology such as RDF or OWL, is very challenging for people not schooled in IT. As a result, subject matter experts in need of a semantic technology-based knowledge base are dependent on IT experts for engineering such a knowledge base. This makes the engineering of such domain-specific knowledge bases a very tedious, expensive, and long process. There is not only the cost associated with involving IT experts, but, in general, there is also a vast knowledge gap between the experts in the field and IT experts. Such a knowledge gap may take a long time to be bridged and could lead to a lot of miscommunication and misunderstandings.

The proposed end-user approach is based on the process for creating, maintaining and exploiting knowledge graphs introduced in [24]. Following that work, we distinguish three main activities: Knowledge Base Construction, Knowledge Storage, and Knowledge Consumption. We first explained these different phases and discussed whether and how they can be performed by end users. For some tasks and given the current state of technology, it does not yet seem feasible

that they are (completely) performed by end users. In that case, we argued for tools that automate the task or semi-automatically perform the task with limited input from IT experts.

Next, we presented the tools developed to support this end-user approach. Currently, the main focus of the toolkit, called DIY-KR-KIT, is on providing support for the development of the ontology and filling the knowledge base with data. For the development of the ontology and the manual filling of the knowledge base, we used the jigsaw metaphor. The purpose of applying this metaphor is to hide the technicalities and terminology of the semantic technologies used for creating ontology-based knowledge bases. A first tool allows subject matter experts to create their own domain ontology, meaning that they can define the concepts and relationships used in their domain and needed to formally represent the available knowledge, and if possible link them to external sources defining them. Next, a similar tool allows the subject matter experts to set up the knowledge base and fill it with data. The approach, and the tools, are demonstrated and evaluated[14] for building a knowledge base in the toxicology domain. Furthermore, we developed a tool to automatically import data previously collected by subject matter experts in spreadsheets. We also discussed directions to deal with quality assurance.

Currently, we investigate how to provide end-user support for the Knowledge Consumption phase. The main challenge in this respect is to allow subject matter experts to create complex queries without the need for learning technical query languages such as SPARQL. Other future work includes the investigation for appropriate end-user methods and tools for knowledge base evolution, i.e., managing and propagating changes in the domain ontology to the knowledge base, and specifying reasoning capabilities.

Acknowledgements

The research of Audrey Sanctorum has been funded by an FWO Postdoc Fellowship (1276721N) of the Research Foundation Flanders.

References

1. Abraham, R., Erwig, M.: Ucheck: A spreadsheet type checker for end users. Journal of Visual Languages & Computing **18**(1), 71–95 (2007)
2. Alobaidi, M., Malik, K.M., Hussain, M.: Automated Ontology Generation Framework Powered by Linked Biomedical Ontologies for Disease-Drug Domain. Computer Methods and Programs in Biomedicine **165**, 117–128 (2018). https://doi.org/10.1016/j.cmpb.2018.08.010
3. Antoniou, G., Van Harmelen, F.: Web Ontology Language: Owl. In: Handbook on Ontologies, pp. 67–92. Springer (2004)
4. Belghiat, A., Bourahla, M.: An Approach based AToM3 for the Generation of OWL Ontologies from UML Diagrams. International Journal of Computer Applications **41**(3), 41–48 (2012)

[14] A user study to evaluate the tools was described in [53]

5. Bernstein, A., Kaufmann, E.: GINO - A Guided Input Natural Language Ontology Editor. In: Proceedings of ISWC 2006, International Conference on the Semantic Web. Athens, USA (November 2006). https://doi.org/10.1007/11926078_11
6. Bikakis, N., Sellis, T.K.: Exploration and Visualization in the Web of Big Linked Data: A Survey of the State of the Art. Computing Research Repository **abs/1601.08059** (2016), http://arxiv.org/abs/1601.08059
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. **5**(3), 1–22 (2009)
8. Bottoni, P., Ceriani, M.: SPARQL Playground: A Block Programming Tool to Experiment with SPARQL. In: Proceedings of ISWC 2015, International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data. Bethlehem, USA (October 2015)
9. Brilhante, V.V.B.B., Macedo, G.T., Macedo, S.F.: Heuristic Transformation of Well-Constructed Conceptual Maps into OWL Preliminary Domain Ontologies. In: Proceedings of IBERAMIA-SBIA-SBRN 2006, Workshop on Ontologies and their Applications. Ribeirao Preto, Brazil (October 2006)
10. Bruhn, J.G.: Beyond discipline: Creating a culture for interdisciplinary research. Integrative Physiological and Behavioral Science **30**(4), 331–341 (1995). https://doi.org/10.1007/BF02691605
11. Cahyani, D.E., Wasito, I.: Automatic Ontology Construction Using Text Corpora and Ontology Design Patterns (ODPs) in Alzheimer's Disease. Jurnal Ilmu Komputer dan Informasi **10**(2), 59–66 (2017). https://doi.org/10.21609/jiki.v10i2.374
12. Chambers, C., Erwig, M.: Automatic detection of dimension errors in spreadsheets. Journal of Visual Languages & Computing **20**(4), 269–283 (2009)
13. Chasseray, Y., Barthe-Delanoë, A.M., Négny, S., Le Lann, J.M.: A Generic Metamodel for Data Extraction and Generic Ontology Population. Journal of Information Science (2021). https://doi.org/10.1177/0165551521989641
14. Cimiano, P., Völker, J.: A Framework for Ontology Learning and Data-Driven Change Discovery. In: Proceedings of NLDB 2005, International Conference on Applications of Natural Language to Information Systems. Alicante, Spain (June 2005). https://doi.org/10.1007/11428817_21
15. Colomb, R.M., Raymond, K., Hart, L., Emery, P., Welty, C., Xie, G.T., Kendall, E.F.: The Object Management Group Ontology Definition Metamodel. In: Ontologies for Software Engineering and Software Technology, pp. 217–247. Springer (2006). https://doi.org/10.1007/3-540-34518-3_8, https://doi.org/10.1007/3-540-34518-3_8
16. Consortium, W.W.W., et al.: Resource Description Framework (RDF)¡ http://www. w3. org. RDF/[Last accessed 2008-06-04]
17. Corcho, Ó., Fernández-López, M., Gómez-Pérez, A., López-Cima, A.: Building legal ontologies with METHONTOLOGY and webode. In: Benjamins, V.R., Casanovas, P., Breuker, J., Gangemi, A. (eds.) Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications, vol. 3369, pp. 142–157 (2003). https://doi.org/10.1007/978-3-540-32253-5_9, https://doi.org/10.1007/978-3-540-32253-5_9
18. Corno, F., Russis, L.D., Roffarello, A.M.: My IoT Puzzle: Debugging IF-THEN Rules Through the Jigsaw Metaphor. In: Proceedings of IS-EUD, International Symposium on End-User Development. Hatfield, UK (July 2019). https://doi.org/10.1007/978-3-030-24781-2_2
19. Danado, J., Paternò, F.: Puzzle: A mobile application development environment using a jigsaw metaphor. Journal of Visual Languages and Computing **25**(4) (2014). https://doi.org/10.1016/j.jvlc.2014.03.005

20. Das, S., Cyganiak, R., Sundara, S.: R2RML: RDB to RDF Mapping Language. W3c recommendation, W3C (2012), https://www.w3.org/TR/r2rml/
21. Debruyne, C., O'Sullivan, D.: R2RML-F: towards sharing and executing domain logic in R2RML mappings. In: Auer, S., Berners-Lee, T., Bizer, C., Heath, T. (eds.) Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016). CEUR Workshop Proceedings, vol. 1593. CEUR-WS.org (2016), http://ceur-ws.org/Vol-1593/article-13.pdf
22. Debruyne, C., Riggio, J., Gustafson, E., O'Sullivan, D., Vinken, M., Vanhaecke, T., De Troyer, O.: Facilitating Data Curation: a Solution Developed in the Toxicology Domain. In: Proceedings of ICSC 2020, International Conference on Semantic Computing. pp. 315–320 (January 2020)
23. Debruyne, C., Tran, T., Meersman, R.: Grounding ontologies with social processes and natural language. J. Data Semant. **2**(2-3), 89–118 (2013). https://doi.org/10.1007/s13740-013-0023-3, https://doi.org/10.1007/s13740-013-0023-3
24. Denaux, R., Ren, Y., Villazón-Terrazas, B., Alexopoulos, P., Faraotti, A., Wu, H.: Knowledge architecture for organisations. In: Pan, J.Z., Vetere, G., Gómez-Pérez, J.M., Wu, H. (eds.) Exploiting Linked Data and Knowledge Graphs in Large Organisations, pp. 57–84. Springer (2017). https://doi.org/10.1007/978-3-319-45654-6_3, https://doi.org/10.1007/978-3-319-45654-6_3
25. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Bizer, C., Heath, T., Auer, S., Berners-Lee, T. (eds.) Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014. CEUR Workshop Proceedings, vol. 1184. CEUR-WS.org (2014), http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf
26. Dobing, B., Parsons, J.: How uml is used. Communications of the ACM **49**(5), 109–113 (2006)
27. Dorodnykh, N.O., Kotlov, Y., Nikolaychuk, O.A., Popov, V.M., Yurin, A.Y.: End-User Development of Knowledge Bases for Semi-Automated Formation of Task Cards. In: Bychkov, I.V., Tchernykh, A., Feoktistov, A.G. (eds.) Proceedings of ICCS-DE 2021, 3rd International Workshop on Information, Computation, and Control Systems for Distributed Environments. Irkutsk, Russia (July)
28. Feeney, K.C., O'Sullivan, D., Tai, W., Brennan, R.: Improving Curated Web-Data Quality with Structured Harvesting and Assessment. International Journal on Semantic Web and Information Systems **10**(2), 35–62 (2014). https://doi.org/10.4018/ijswis.2014040103, https://doi.org/10.4018/ijswis.2014040103
29. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: CLOnE: Controlled Language for Ontology Editing. In: Proceedings of ISWC 2007, International Semantic Web Conference (November 2007). https://doi.org/10.1007/978-3-540-76298-0_11
30. Gómez-Pérez, A.: Ontology Evaluation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 251–274. International Handbooks on Information Systems, Springer (2004)
31. Gozzi, R.: The Jigsaw Puzzle as a Metaphor for Knowledge. ETC: A Review of General Semantics **53**(4), 447–451 (1996)

32. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a Control Natural Language for Authoring Ontologies. In: Proceedings of ESWC 2008, European Semantic Web Conference. Tenerife, Spain (June 2008). https://doi.org/10.1007/978-3-540-68234-9_27
33. Hepp, M.: Ontologies: State of the art, business potential, and grand challenges. In: Hepp, M., Leenheer, P.D., de Moor, A., Sure, Y. (eds.) Ontology Management, Semantic Web, Semantic Web Services, and Business Applications, Semantic Web and Beyond: Computing for Human Experience, vol. 7, pp. 3–22. Springer (2008). https://doi.org/10.1007/978-0-387-69900-4_1, https://doi.org/10.1007/978-0-387-69900-4_1
34. Horridge, M., Gonçalves, R.S., Nyulas, C.I., Tudorache, T., Musen, M.A.: Webprotégé: A cloud-based ontology editor. In: Amer-Yahia, S., Mahdian, M., Goel, A., Houben, G., Lerman, K., McAuley, J.J., Baeza-Yates, R., Zia, L. (eds.) Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019. pp. 686–689. ACM (2019). https://doi.org/10.1145/3308560.3317707, https://doi.org/10.1145/3308560.3317707
35. Humble, J., Crabtree, A., Hemmings, T., Åkesson, K., Koleva, B., Rodden, T., Hansson, P.: "Playing with the Bits" User-Configuration of Ubiquitous Domestic Environments. In: Proceedings of UbiComp 2003, International Conference on Ubiquitous Computing. Seattle, USA (October 2003). https://doi.org/10.1007/978-3-540-39653-6_20
36. Jarrar, M., Meersman, R.: Ontology engineering - the DOGMA approach. In: Dillon, T.S., Chang, E., Meersman, R., Sycara, K.P. (eds.) Advances in Web Semantics I - Ontologies, Web Services and Applied Semantic Web, Lecture Notes in Computer Science, vol. 4891, pp. 7–34. Springer (2009). https://doi.org/10.1007/978-3-540-89784-2_2, https://doi.org/10.1007/978-3-540-89784-2_2
37. Junior, A.C., Debruyne, C., Longo, L., O'Sullivan, D.: On the Mental Workload Assessment of Uplift Mapping Representations in Linked Data. In: Proceedings of H-WORKLOAD 2018, International Conference on Human Mental Workload: Models and Applications. Amsterdam, The Netherlands (September 2018). https://doi.org/10.1007/978-3-030-14273-5_10
38. Junior, A.C., Debruyne, C., O'Sullivan, D.: Juma Uplift: Using a Block Metaphor for Representing Uplift Mappings (January - February 2018). https://doi.org/10.1109/ICSC.2018.00037
39. Kawamoto, K., Kitamura, Y., Tijerino, Y.A.: KawaWiki: A Semantic Wiki Based on RDF Templates. In: Proceedings of IEEE/WIC/ACM 2006, International Conference on Intelligent Agent Technology
40. Kertcher, Z.: Gaps and Bridges in Interdisciplinary Knowledge Integration. In: e-Research Collaboration, pp. 49–64 (2010). https://doi.org/10.1007/978-3-642-12257-6_4
41. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL) (2017), https://www.w3.org/TR/shacl/
42. Kotis, K., Vouros, G.A.: Human-centered ontology engineering: The HCOME methodology. Knowl. Inf. Syst. **10**(1), 109–131 (2006). https://doi.org/10.1007/s10115-005-0227-4, https://doi.org/10.1007/s10115-005-0227-4
43. Lieberman, H., Paternò, F., Wulf, V. (eds.): End User Development: An Emerging Paradigm. Human-Computer Interaction Series, Springer (2006). https://doi.org/10.1007/1-4020-5386-X, https://doi.org/10.1007/1-4020-5386-X

44. Maedche, A., Staab, S.: Ontology learning. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 173–190. International Handbooks on Information Systems, Springer (2004)
45. Maynard, D., Funk, A., Peters, W.: Using Lexico-Syntactic Ontology Design Patterns for Ontology Creation and Population. In: Proceedings of ISWC 2009, Workshop on Ontology Patterns. Washington, USA (October 2009)
46. McKenna, L., Debruyne, C., O'Sullivan, D.: Understanding the position of information professionals with regards to linked data: A survey of libraries, archives and museums. In: Chen, J., Gonçalves, M.A., Allen, J.M., Fox, E.A., Kan, M., Petras, V. (eds.) Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL 2018, Fort Worth, TX, USA, June 03-07, 2018. pp. 7–16. ACM (2018). https://doi.org/10.1145/3197026.3197041, https://doi.org/10.1145/3197026.3197041
47. Mouromtsev, D., Pavlov, D.S., Emelyanov, Y., Morozov, A.V., Razdyakonov, D.S., Galkin, M.: The simple web-based tool for visualization and sharing of semantic data and ontologies. In: Villata, S., Pan, J.Z., Dragoni, M. (eds.) Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015. CEUR Workshop Proceedings, vol. 1486. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1486/paper_77.pdf
48. Noy, N.F., Crubézy, M., Fergerson, R.W., Knublauch, H., Tu, S.W., Vendetti, J., Musen, M.A.: Protégé-2000: An Open-Source Ontology-Development and Knowledge-Acquisition Environment: AMIA 2003 Open Source Expo. In: Proceedings of AMIA 2003, Annual Symposium on American Medical Informatics. Washington, USA (November 2003)
49. Öztürk, Ö., Özacar, T.: A Case Study for Block-based Linked Data Generation: Recipes as Jigsaw Puzzles. Journal of Information Science **46**(3) (2020). https://doi.org/10.1177/0165551519849518
50. Pantazos, K., Lauesen, S., Vatrapu, R.K.: End-User Development of Information Visualization. In: Dittrich, Y., Burnett, M.M., Mørch, A.I., Redmiles, D.F. (eds.) Proceedings of IS-EUD 2013, 4th International Symposium on End-User Development. Lecture Notes in Computer Science, vol. 7897, pp. 104–119. Springer, Copenhagen, Denmark (June 2013). https://doi.org/10.1007/978-3-642-38706-7_9, https://doi.org/10.1007/978-3-642-38706-7_9
51. Pinto, H.S., Tempich, C., Staab, S.: Ontology Engineering and Evolution in a Distributed World Using DILIGENT. In: Handbook on Ontologies, pp. 153–176. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92673-3_7, http://link.springer.com/10.1007/978-3-540-92673-3_7
52. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J.S., Silverman, B., Kafai, Y.B.: Scratch: programming for all. Communications of the ACM **52**(11), 60–67 (2009). https://doi.org/10.1145/1592761.1592779
53. Sanctorum, A., Riggio, J., Sepehri, S., Arnesdotter, E., Vanhaecke, T., Troyer, O.D.: A Jigsaw-Based End-User Tool for the Development of Ontology-Based Knowledge Bases. In: Fogli, D., Tetteroo, D., Barricelli, B.R., Borsci, S., Markopoulos, P., Papadopoulos, G.A. (eds.) Proceedings of IS-EUD 2021,8th International Symposium on End-User Development. Lecture Notes in Computer Science, vol. 12724, pp. 169–184. Springer (July 2021). https://doi.org/10.1007/978-3-030-79840-6_11, https://doi.org/10.1007/978-3-030-79840-6_11

54. Siedlok, F., Hibbert, P.: The Organization of Interdisciplinary Research: Modes, Drivers and Barriers. International Journal of Manangement Reviews **16**(3), 194–210 (2014)
55. Slimani, T.: Ontology Development: A Comparing Study on Tools, Languages and Formalisms. Indian Journal of Science and Technology **8**(24), 1–12 (2015). https://doi.org/10.17485/ijst/2015/v8i1/54249
56. Soylu, A., Giese, M., Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I.: Ontology-based End-User Visual Query Formulation: Why, what, who, how, and which? Universal Access in the Information Society **16**(2), 435–467 (2017). https://doi.org/10.1007/s10209-016-0465-0, https://doi.org/10.1007/s10209-016-0465-0
57. Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: The neon methodology framework: A scenario-based methodology for ontology development. Appl. Ontology **10**(2), 107–145 (2015). https://doi.org/10.3233/AO-150145, https://doi.org/10.3233/AO-150145
58. Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D.: OntoEdit: Collaborative Ontology Development for the Semantic Web. In: Proceedings of ISWC 2002, International Semantic Web Conference. Sardinia, Italy (June 2002). https://doi.org/10.1007/3-540-48005-6_18
59. Vrandecic, D.: Ontology evaluation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 293–313. International Handbooks on Information Systems, Springer (2009). https://doi.org/10.1007/978-3-540-92673-3_13, https://doi.org/10.1007/978-3-540-92673-3_13
60. Yurin, A.Y., Dorodnykh, N.O.: Personal Knowledge Base Designer: Software for Expert Systems Prototyping. SoftwareX **11**, 100411 (2020). https://doi.org/10.1016/j.softx.2020.100411, https://doi.org/10.1016/j.softx.2020.100411