Domain-Specific Customization of Schema.org Based on SHACL

Umutcan Şimşek(✉), Kevin Angele, Elias Kärle, Oleksandra Panasiuk, and Dieter Fensel

University of Innsbruck, Innsbruck, Austria
{umutcan.simsek,kevin.angele,elias.kaerle,
oleksandra.panasiuk,dieter.fensel}@sti2.at

Abstract. Schema.org is a widely adopted vocabulary for semantic annotation of web resources. However, its generic nature makes it complicated for publishers to pick the right types and properties for a specific domain. In this paper, we propose an approach, a domain specification process that generates domain-specific patterns by applying operators implemented in SHACL syntax to the schema.org vocabulary. These patterns can support annotation generation and verification processes for specific domains. We provide tooling for the generation of such patterns and evaluate the usability of both domain-specific patterns and the tools with use cases in the tourism domain.

Keywords: SHACL · Schema.org · Semantic annotation · Domain-specific patterns

1 Introduction

schema.org [7] is currently the de facto standard for annotating resources on the web. The vocabulary is maintained by the schema.org initiative, and it contains 821 types and 1328 properties[1]. The schema.org vocabulary has a highly generic nature. On the one hand, the vocabulary covers many domains (e.g., events, media, accommodation) superficially, on the other hand, it does not cover individual domains in detail.

The data model of schema.org vocabulary is quite flexible in terms of type hierarchy and inheritance of properties by specialized types (e.g., a Waterfall can have a telephone number). Moreover, there are multiple ways to represent the same information (e.g., the address of a Place can be represented with three different properties in two different ways). This flexibility and heterogeneity come with two side effects for data publishers: (a) The generic nature of the vocabulary can make the creation of correct, complete and concise annotations quite challenging, especially within a specific domain. The publishers may not know which types and properties to select for their domain or may use different

[1] See https://schema.org/docs/schemas.html Last accessed: 16.04.2020.

J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12507, pp. 585–600, 2020.
https://doi.org/10.1007/978-3-030-62466-8_36

properties to represent the same information, and (b) the schema.org vocabulary may not contain specific types and properties for their domain. These two issues may have harming implications on the overall quality of the annotations and consequently the applications built on top of them. The completeness would be harmed by missing important domain-relevant information. Similarly, the conciseness of the annotations would be harmed since publishers could be more prone to use different properties to represent same information on an annotation (see coverage and succinctness quality dimensions [9, Sect. 7.2 and 7.4]).

One way to guide data publishers to create semantically annotated data and content for a domain is to provide domain-specific patterns of the schema.org vocabulary. They would serve as an agreement between publishers and consumers to ease the annotation generation and verification processes. In this paper, we present an approach for generating such patterns. Figure 1 depicts the domain specification process. We apply domain specification operators implemented with Shapes Constraint Language (SHACL) [10] syntax to the schema.org vocabulary to have *an extended subset of schema.org* for a domain[2]. The main reason for utilizing (a subset of) SHACL is to benefit from its widespread adoption and tool support. It is a W3C recommendation that would have a substantial impact on the uptake of our approach.

The remainder of the paper is structured as follows: In Sect. 2, we give a brief introduction to schema.org and our usage of SHACL. Section 3 describes the domain specification process and gives a running example. Section 4 briefly introduces the tool support for domain specifications and their usage in annotation generation. We evaluate the usability and benefit of domain-specific patterns and its tool support in Sect. 5 with use cases in tourism domain. Section 6 gives an overview of the related work and Sect. 7 concludes the paper with final remarks and future directions.

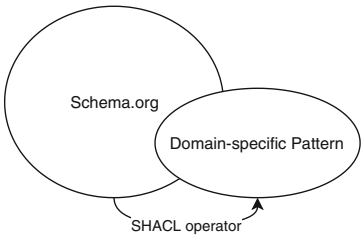

Fig. 1. The domain specification process

2 Preliminaries

In this section, we give an introduction to the data model of schema.org as well as SHACL, the syntax of which we use to define the domain specification operators.

[2] There is an automatically generated version of entire schema.org in SHACL maintained by TopQuadrant http://datashapes.org/schema.

2.1 Schema.org

The data model of schema.org is quite simple and "very generic and derived from RDF Schema."[3]. It does not have a formal semantics[4]. Nevertheless, it provides informal definitions and guidelines. Similar to an RDFS vocabulary, the schema.org data model organizes types in a multiple inheritance hierarchy. There are two disjoint hierarchies, namely the item types that are more specific than s:Thing and data types that are more specific than s:DataType[5]. The vocabulary contains properties that have one or more types in their domains and one or more types in their range definitions. A significant deviation from RDFS comes with the properties, mainly how their domains and ranges are defined. schema.org defines the domains and ranges with s:domainIncludes and s:rangeIncludes properties, respectively. The semantics of these properties are not formally defined. However, the documentation indicates that domain and range definitions are disjunctive, which is not the case for domain and range definitions with RDFS.

The data model of schema.org allows global ranges, meaning each range is valid for each domain of a property. However, in many domain-specific cases, local ranges could be instrumental (cf. [12]). For instance, a SportsEvent is in the domain of the location property. The property has a global range of Place. In a domain-specific scenario, it is not hard to imagine that a sports event takes place in a sports activity location. Therefore, defining the range of the location property on SportsEvent type as SportsActivityLocation may be a better modeling practice. With the domain specification approach, we allow the definition of schema.org types with local ranges on their properties.[6]

When creating annotations on the web, the range of a property may need to be altered with the conjunction of multiple types. A prominent example for this is the so-called multi-typed entity (MTE) practice for the annotation of hotel rooms[7]. The s:HotelRoom type contains properties for describing beds and amenity features. However, to describe a daily room price, the hotel room must also be defined as an instance of s:Product, which allows the usage of schema:offers property. The MTE practice is an excellent example of why customization of schema.org for specific domains is needed. For data publishers, it may be complicated to find out which types should be used for an MTE. The domain-specific patterns created by domain experts can enforce the conjunction of multiple types as the range of a property to guide data publishers.

[3] https://schema.org/docs/datamodel.html - accessed on 16.04.2020.
[4] Patel-Schneider gives an analysis of the vocabulary and a possible formal semantics in [12].
[5] s is a prefix for the http://schema.org/ namespace.
[6] This is also why we ignore the property hierarchy of schema.org since this is already implemented via local properties.
[7] See MTE documentation: https://tinyurl.com/s2l3btw.

2.2 Our Take on SHACL

SHACL is a W3C recommendation for defining constraints over RDF graphs. For the domain specification process proposed in this paper, we use a subset of SHACL-CORE[8] elements. We adopt three main types of SHACL elements, namely shapes, target selectors, and constraint components [10]:

- *Class-based target selector (sh:targetClass)*, to specify the type on which the pattern is based
- *Node Shape with a target selector*, to specify the domain-specific pattern
- Shape-based Constraint Components to define local properties (sh:property) and range restrictions (sh:node)
- *Value Type Constraint Components (sh:class and sh:datatype)*, to define ranges on local properties
- *Logical Constraint Components (sh:or)* to define disjunctive ranges
- and various other constraint components to define constraints on local properties beyond range restrictions (see Sect. 3 for a complete list).

Despite adopting most of the SHACL-CORE components, the domain specification operators have stricter syntax rules. For instance, SHACL allows the usage of many constraint components on both node and property shapes, but we allow all constraint components only on property shapes. Moreover, we allow target definitions only on the node shapes that are not a value of sh:node property, in other words, only on the node shape that specifies the pattern. The property shapes are only allowed as a Shape based-constraint and not as standalone shapes[9]. In the next section, we explain how each SHACL element is used in the domain specification process.

3 Domain Specification

A domain expert creates a domain-specific pattern from schema.org and its extensions through the following actions:

1. Removing types and properties from schema.org that are not relevant for a specific domain.
2. Defining local properties and their ranges on the remaining types
3. Optionally, further restricting the types in the ranges of defined local properties
4. Optionally, extending the ranges with new types from schema.org and its extensions
5. Optionally, defining additional constraints on the local properties.

[8] *sh* prefix is used for the SHACL-Core namespace.

[9] An abstract syntax for our domain specification operators based on SHACL can be found online: https://tinyurl.com/qmkb3ln.

We explain the process mentioned above more concretely with a running example. A domain expert creates a domain specification operator that generates a domain-specific pattern for the accommodation domain. The process starts by eliminating all irrelevant types from schema.org. In this example, this action leaves us with the s:LodgingBusiness, s:Text, s:DateTime, schema:Place and s:PostalAddress and properties s:name, s:checkInTime, s:checkOutTime, s:containsPlace and s:location. Listing 1 shows the domain specification operator in the SHACL syntax. This operator defines five local properties and ranges on s:LodgingBusiness type[10]. The first two steps eliminate more than 100 properties that are allowed on s:LodgingBusiness by schema.org.

Additionally, some of the types in the property ranges are eliminated. A domain expert may want to describe the location of a lodging business with its address. The property has a disjunction of four types in its range, including the s:Place, s:PostalAddress, s:VirtualLocation and s:Text. The s:VirtualLocation type may not be desired for describing the location of an accommodation provider. s:Text is not expressive enough to make a granular description of an address. Both s:Place and s:PostalAddress types can be used to describe a postal address, but s:Place requires more properties than s:PostalAddress. Therefore a domain expert may eliminate the types other than s:PostalAddress from the range of s:location property.

A domain expert may choose to restrict further the types in the ranges of the local properties. Listing 2 shows such a restriction on two properties. The s:PostalAddress type in the range of the s:location property defined on s:LodgingBusiness is restricted further to a type that allows only s:addressLocality and s:addressCountry properties. Similarly, the range of s:containsPlace is restricted to a type that is a conjunction of s:HotelRoom (subtype of s:Place) and s:Product. The property s:containsPlace is defined by schema.org vocabulary very generically, to define a place that contains other places. In a specific domain like accommodation, domain experts may want to describe only hotel rooms and their offers. Therefore, they restrict the s:Place type in the range to s:HotelRoom and to allow the definition of offers, they create a conjunction with the type s:Product (see also the explanation of MTEs in Sect. 2.1).

A domain-specific pattern is an extended subset of schema.org; however, so far, we only defined a subset of the vocabulary. The schema.org vocabulary can be extended externally[11]. The extensions of schema.org are built following the same data model as schema.org and assumed to be hosted externally. The domain specification process can use external extensions of schema.org for:

- using a type from an extension to specify the pattern
- defining a property from an extension as a local property on a type
- adding types from an extension to the ranges of local properties

[10] Datatypes like s:Text and s:DateTime are mapped to xsd:string and xsd:datetime respectively for compatibility to SHACL syntax.

[11] See External Extensions: https://schema.org/docs/schemas.html.

```
@prefix sh: <http://www.w3.org/ns/shacl#>.
@prefix schema: <http://schema.org/>.
@prefix smtfy: <https://semantify.it/ds/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

smtfy:l49vQ318v a sh:NodeShape;
    sh:targetClass schema:LodgingBusiness;
    schema:name "LodgingBusiness";
    sh:property [
        sh:path schema:name;
        sh:datatype xsd:string;
    ];
    sh:property [
        sh:path schema:checkInTime;
        sh:datatype xsd:datetime;
    ];
    sh:property [
        sh:path schema:checkOutTime;
        sh:datatype xsd:datetime;
    ];
    sh:property [
        sh:path schema:containsPlace;
        sh:class schema:Place;
    ];
    sh:property [
        sh:path schema:location;
        sh:class schema:PostalAddress;
    ].
```

Listing 1: A domain specification operator for defining a domain-specific pattern in the accommodation domain

In Listing 3, we define a new local property n:totalNumberOfBeds[12] from an external schema.org extension. Similarly, the range of an existing property can be extended with a type from an extension. Here we add the n:Sauna type to the range of the s:containsPlace property.

A domain-specific pattern can apply constraints on properties beyond the type restrictions in their range definitions. A domain specification operator supports the definition of following types of constraint (based on the naming convention of SHACL constraint components):

- Cardinality constraints to enforce arbitrary cardinalities on property values (e.g., via sh:minCount)
- Value-range constraints to restrict the ranges of numerical property values and time values (e.g., via sh:minInclusive)
- String-based constraints to enforce length ranges, patterns or language tags on string literal values (e.g., via sh:pattern)
- Property pair constraints to enforce constraints involving the values of two properties (e.g., via sh:lessThan)
- Enumeration constraints to restrict the range of a property to a specific set of values (via sh:in parameters)

[12] *n* prefix is used for the namespace of a schema.org extension.

```
...
    sh:property [
        sh:path schema:containsPlace;
        sh:class schema:HotelRoom;
        sh:class schema:Product;
    ];
    sh:property [
        sh:path schema:location;
        sh:class schema:PostalAddress;
        sh:node [
            a sh:NodeShape;
            sh:property [
                sh:path schema:addressCountry;
                sh:datatype xsd:string;
            ];
            sh:property [
                sh:path schema:addressLocality;
                sh:datatype xsd:string;
            ];
        ]
    ].
```

Listing 2: The domain specification operator further restricts the local ranges

Listing 4 shows the addition of a property pair constraint that ensures that check-in time is always earlier than the check-out time. Additionally, it adds a minimum cardinality constraint to the s:checkinTime property to indicate that the property is required.

In this section, we described the domain specification process that applies a domain specification operator to schema.org vocabulary in order to create extended subsets of the vocabulary. As shown in the running example, the first two steps are enough to have the simplest form of a domain specification operator since it already creates a subset of schema.org. A full domain specification operator with further restrictions on ranges (e.g. further restrictions on the range of s:containsPlace and more cardinality constraints) can be found online[13].

The patterns can be used as machine-readable guidelines for annotation creation, but also as a specification to verify existing annotations (see also Sect. 4). In order to focus on the in-use aspects, we have left out the formal definitions in this section. A detailed abstract syntax for domain specification operators and formal semantics of the verification process can be found online[14].

4 Tools

We developed a set of tools to help domain experts with the creation of domain-specific patterns and to help data publishers with the usage of the patterns for creating annotations with schema.org. The tools are integrated with seman-

[13] https://semantify.it/ds/l49vQ318v.
[14] https://tinyurl.com/qmkb3ln.

```
...
    sh:property [
        sh:path n:totalNumberOfBeds;
        sh:datatype xsd:integer;
    ];
    sh:property [
        sh:path schema:containsPlace;
        sh:or ([sh:class schema:HotelRoom;
        sh:class schema:Product;] [sh:class n:Sauna])
    ];
...
```

Listing 3: A property from an external extension is defined on s:LodgingBusiness. The range of s:containsPlace is extended with a type from an extension

```
...
    sh:property [
        sh:path schema:checkInTime;
        sh:datatype xsd:datetime;
        sh:minCount 1;
        sh:lessThan schema:checkOutTime;
    ];
...
```

Listing 4: The domain specification operator extended with property pair and cardinality constraints

tify.it[15], a platform for creation, publication, and evaluation of semantic annotations. In this section, we will briefly introduce the domain specification tools, namely the domain specification editor and visualizer, the annotation verifier, and the annotation editors that utilize the domain specifications.

4.1 Domain Specification Editor and Visualizer

The domain specification editor provides a user interface for generating domain-specific patterns[16]. Figure 2 shows a part of the interface. A user first fills some metadata about the domain-specific pattern and selects a target type. Alternatively, an existing pattern can be used as a template. At this stage, extensions of schema.org can also be included in the editor via the advanced settings. After the target type is selected, the local properties and their ranges must be selected recursively. In Fig. 2, the user can click the blue pencil icon to restrict the range of the location property based on the PostalAddress type. Further constraints for a property can be specified by clicking the advanced settings button next to

[15] https://semantify.it.
[16] https://semantify.it/domainspecifications/create.

that property[17]. After a domain-specific pattern is created, it can be visualized in multiple forms, such as tabular, tree, and graph representation[18,19].

Using a subset of SHACL-CORE components helps us to keep the tool simple with a rather straightforward workflow. Unlike more generic shape editors (e.g. [4]), the domain specification editor supports a linear workflow. For the case of semantic annotations, this is more intuitive since the annotations are typically single sourced graphs with a main type[20].

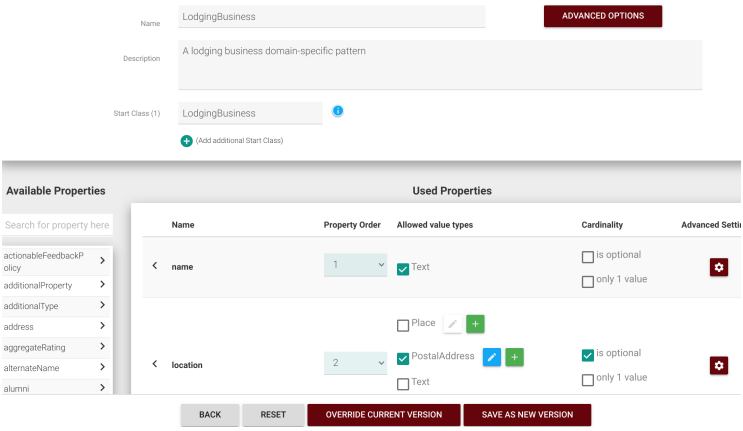

Fig. 2. A screenshot of the domain specification editor

4.2 Annotation Verifier

The semantify.it platform hosts a verifier implementation that extracts schema.org annotations from a web page and verifies them against domain-specific patterns[21,22]. An annotation is first syntactically verified against the schema.org vocabulary, then verified against the selected domain-specific pattern. A report is then produced for each annotation, describing the errors and their sources[23].

[17] Not all advanced constraints (e.g. property-pair consraints for dates) are supported by the editor at the submission of this paper.

[18] We use a customized version of VOWL library http://vowl.visualdataweb.org/webvowl.html.

[19] An example can be found in https://semantify.it/domainspecifications/public/l49vQ318v.

[20] See the formal definition: https://tinyurl.com/qmkb3ln.

[21] https://semantify.it/validator/.

[22] https://semantify.it/domainSpecifications Currently, there are 400 domain-specific patterns created by semantify.it users.

[23] Comparable to a SHACL validator.

4.3 Annotation Editors

The domain-specific patterns are used to guide the data publishers. We automatically generate editors to create annotations based on domain-specific patterns. An example of such generated editor is our Wordpress extension "Instant Annotation"[24]. The plugin uses a predefined set of domain-specific patterns to create minimal editors to annotate Wordpress content. These domain-specific patterns typically target Search Engine Optimization.

5 Use Cases and Evaluation

In this section, we first describe different use cases where the domain-specific patterns are being applied. The use cases are currently dominantly from the tourism domain. However, our approach can be applied to any domain since the syntax of domain specification operators (i.e., SHACL) and the approach itself are domain-independent[25]. We also provide a preliminary evaluation of the domain specification editor's usability and the benefit of domain-specific patterns.

5.1 Web Content Annotations in Tourism

Domain-specific patterns have been used in the tourism domain by Destination Management Organizations (DMOs). The work in [1] describes a use case regarding the generation of schema.org annotated tourism-related data such as events, accommodation, and infrastructure from raw data sources. The patterns have been used to guide the mapping process of the metadata of the IT solution provider of DMO Mayrhofen[26,27]. The domain-specific patterns are also used for manual generation of annotated data, through an editor that dynamically builds forms for annotation creation based on domain-specific patterns. The generated annotations have been used to annotate web pages for Search Engine Optimization and as a data source for a chatbot. The annotations created with the help of the domain-specific patterns are evaluated qualitatively by observing search engine results.

5.2 Schema-Tourism and DACH-KG Working Groups

The schema-tourism working group was founded as a place for the experts in the tourism domain and researchers to commonly work on a) a unified way to

[24] https://wordpress.org/plugins/instant-annotation/.
[25] See BioSchemas [6] for a potential use case in another domain.
[26] https://mayrhofen.at.
[27] DMOs are organizations that promote touristic services in a region. Similar annotation projects have been conducted with other DMOs such as DMO Fügen (https://best-of-zillertal.at), Seefeld (https://seefeld.com), Wilder Kaiser (https://wilderkaiser.info), and promotion agencies like Tirol Werbung (https://tirol.at).

use schema.org in the tourism domain and b) to identify the shortcomings of schema.org and extend the vocabulary when needed. The findings of the working group are published online as domain-specific patterns[28]. There are currently 81 patterns created by the working group. The documentation follows the schema.org documentation style and provides a list of all available domain-specific patterns, including a short description. By clicking on the title of a pattern, its description page is shown. This page lists all mandatory and optional properties with their respective ranges. If a type in the range is restricted in the pattern, then a link leads to its description. Otherwise, the link leads to schema.org's description of the type. Alongside the human-readable representation, the domain specification operator in SHACL syntax can also be obtained for each pattern.

The DACH-KG working group was founded with the primary goal of building a touristic knowledge graph for the region of Austria, Germany, South Tyrol, and Switzerland. Several stakeholders in the consortium[29], including but not limited to the German National Tourist Board[30], Austria Tourism Agency[31] and IDM South Tyrol[32] are currently working on a unified schema to represent their data in the knowledge graph. Hence the current focus of this working group lies on identifying the mappings from heterogeneous data sources to schema.org and defining best practices for further use of schema.org. The domain-specific patterns help the DACH-KG to formalize their findings and disseminate their best practice patterns to data and IT solution providers.

As a starting point, DACH-KG working group analyzes the existing patterns developed by the Schema-Tourism Working Group and develops additional patterns or suggests modifications. This process also includes the development of a schema.org extension, in order to increase the domain-specific coverage of schema.org for tourism. During this process, DACH-KG consortium uses schema.org extensions from different IT solution providers in tourism sector as an input. An example pattern developed by the working group can be found online[33].

The domain-specific patterns are already being adopted by the individual participants of the DACH-KG group in their internal semantic annotation processes. For example, Thüringen Tourism is using semantify.it, particularly the domain specification editor to develop their own patterns[34]. Additionally, the German Center For Tourism announced an open call for a feasibility study for

[28] https://ds.sti2.org.
[29] Full list can be found at https://www.tourismuszukunft.de/2018/11/dach-kg-auf-dem-weg-zum-touristischen-knowledge-graph/.
[30] https://germany.travel.
[31] https://austriatourism.com.
[32] https://www.idm-suedtirol.com/.
[33] https://ds.sti2.org/TQyCYm-r5.
[34] see slide 5 at https://thueringen.tourismusnetzwerk.info/download/pdf-Veranstaltungen/ThueCAT.pdf.

the upcoming German Tourism Knowledge Graph and required applicants to follow the domain-specific patterns on https://ds.sti2.org[35].

5.3 Preliminary Evaluation

We conducted two user studies in order to evaluate the usability of the domain specification editor and the benefit of domain-specific patterns for creating semantic annotations with schema.org.

Domain Specification Editor Usability Study. The usability of the domain specification editor has been evaluated with a System Usability Scale (SUS) [3] survey. The survey contains ten questions with 5-point Likert Scale answers. We added the eleventh question, "Overall, I would rate the user-friendliness of this editor as:" with options such as "Worst Imaginable, Awful, Poor, Good, Excellent and Best Imaginable", in order to validate the quantitative SUS score with the qualitative perception of usability. This question has been adapted from the study in [2].

The survey has been conducted anonymously among domain experts and researchers from the DACH-KG working group, who are specialized in tourism (8 participants) and senior bachelor students of Economy, Health, and Sports Tourism (WSGT) program (29 participants). The members of each target group have a basic understanding of schema.org, domain-specific patterns and their purpose.

Table 1 shows the results of the study. The leftmost column represents different target groups. The main difference is that the DACH-KG members are more experienced in the tourism sector than the WSGT students. Judging from the mean ($\overline{x} = 55.27$) and median ($\tilde{x} = 55$) SUS values in comparison with the mean values of the SUS score for each qualitative perception class, the editor has overall "Good" ($\overline{x} = 49.86$) usability. The high difference between the median score of two groups may also indicate that experienced tourism experts appreciate the domain specification editor more than the tourism students[36].

The meaning of SUS scores has also been investigated in [2] with different adjective scales. The adjective scale between Worst Imaginable and Best Imaginable is mapped to minimum mean SUS scores with a study made over 1000 SUS surveys. This adjective scale in this survey also has a central value "OK", but it is observed that the intended meaning of the word OK (whether it is acceptable) is not clear; therefore this option is left out in our study. The minimum mean score for OK in [2] is 50.9, and Good is 71.4; therefore in the presence of the central value, overall usability can be seen as "OK"[37]. The more experienced DACH-KG users' qualitative judgment is more consistent with their SUS scores, while for inexperienced users, participants were tending towards more "Good" (58.6%) than "Poor" (17.2%), even though they have given a low SUS score.

[35] See page 4 at https://tinyurl.com/vlnxu76.

[36] $p = 0.01$. The difference in the median of SUS scores of two groups is significant at $p < 0.05$ based on Mann-Whitney U Test.

[37] Marginally acceptable in acceptability scale. See [2].

Table 1. SUS survey results

				Awful		Poor		Good		Excellent	
	$\overline{x}$	σ	$\tilde{x}$	$\overline{x}$	%	$\overline{x}$	%	$\overline{x}$	%	$\overline{x}$	%
DACH-KG	75	25	82.5	–	0	27.5	12.5	77.5	12.5	82.5	75
WSGT-students	49.82	16.5	50	20	3.4	36.5	17.2	48.23	58.6	70.41	20.7
ALL	55.27	21.24	55	20	2.7	35	16.2	49.86	48.6	76.45	32.4

Domain-Specific Pattern Usage Survey. This survey aims to see how the domain-specific patterns can help the data publishers and IT solution providers with creating semantic annotations with schema.org. We gave 14 computer science master students and software developers who had at least some experience in creating annotations with schema.org and the task of creating annotations with the help of domain-specific patterns hosted on https://ds.sti2.org.

They were first asked to create annotations without any domain-specific patterns; then, they created one with the help of a pattern. After they created their annotations, we have asked two main questions: (a) "Did you find the domain-specific patterns understandable?" (b) "Did domain-specific patterns help you with creating annotations?"

Only 21.4% of the participants found the domain-specific patterns difficult to understand, while 78.6% found it either easy or very easy. As for the second question, we provided answer options like "It made it more complicated", "It did not make any difference" or "It helped by saving time, reducing the complexity or enabling the usage of types and properties that do not exist in schema.org vocabulary"[38]. All participants reported that the domain-specific patterns helped them in some way. More than 50% of the participants reported that the patterns helped them to save time and reduce the complexity of schema.org. About 50% stated that they also helped them to discover and use new types and properties.

6 Related Work

Although there are not many tools and approaches specifically targeting customization of schema.org for semantic annotation, an informal definition of domain specification is made in [14]. This approach only takes a subset of schema.org by removing types and properties. We define the domain specification process formally and make a clear conceptual distinction between the process of domain specification and resulting domain-specific patterns. Additionally, we do not only restrict but also allow extension of the schema.org vocabulary.

A plethora of approaches for verifying RDF data has been proposed. RDFUnit [11] is a framework that mimics the unit tests in software engineering for RDF data by checking an RDF graph against test cases defined with SHACL

[38] Multiple answer selection was possible.

and SPARQL. Integrity Constraints in OWL [15] describes an integrity constraint semantics for OWL restrictions to enable verification. The constraints are then checked with SPARQL. Recent efforts on standardization of RDF verification has lead to approaches that revolve around the notion of *shapes*. A shape typically targets a set of nodes in an RDF graph and applies certain constraints on them. The shape specifications have already found various usage scenarios in the literature: They can be used to assess whether an RDF graph is valid with regards to a specification (e.g., set of shapes) or as an explicit knowledge to determine the concept of completeness for a knowledge graph, typically to be used for symbolic learning[39]. For specifying shapes, two languages, namely the W3C recommendation SHACL [10] and a W3C community group effort Shape Expressions (ShEx) [13] have emerged. SHACL facilitates the creation of data shapes for validating RDF data. Following the specification, SHACL shapes are informally converted into SPARQL queries. Similarly, the ShEx provides mechanisms to create shapes and has an abstract syntax and formal semantics.

Domain-specific patterns can also be seen from the perspective of Content Ontology Design Patterns (CPs). The CPs are used to solve recurrent content modelling problems [8]. A high-level CP can be taken as a reference to create a pattern with our approach for a specific domain. The CPs typically target OWL ontologies while domain-specific patterns are fit to schema.org characteristics (see Sect. 2.1). Therefore, they can be seen as a Content ODP language for semantic annotations on the web, given that the schema.org is the de facto vocabulary for this task.

Perhaps the most relevant work to the domain-specific patterns is BioSchemas[40] [6], which provides extensions to schema.org for the life sciences domain together with simple constraints for their usage such as cardinality. To the best of our knowledge, they currently provide only human-readable specifications publicly. We see BioSchemas as a good potential use case for the domain specification approach and will investigate the possible cooperation in the future work.

The domain specification approach proposed in this paper aims to bring a compact, formal solution for customizing a very large and heterogeneous vocabulary, the de facto content annotation standard schema.org, for specific domains. We utilize SHACL to benefit from its widespread adoption, but the portability of this approach to another language like ShEx is rather straightforward. Even though our approach is related to RDF verification, the main focus here is not to verify a graph but rather to create machine-readable patterns for content and data annotations. The domain-specific patterns can have further interesting usage scenarios. For example, they can be used as a template to determine which subgraph should be returned in case of node lookups, as they would represent the relevant data for a domain (cf. [9, Sect. 9.2.2]).

[39] We refer the reader to the survey by Hogan et al. [9].
[40] https://bioschemas.org.

7 Conclusion and Future Work

Schema.org is the de facto industrial standard for annotating content and data. Due to its design, it is not very straightforward for data publishers to pick the right types and properties for specific domains and tasks. In order to overcome this challenge, machine-understandable domain-specific patterns can guide the knowledge generation process.

In this paper, we proposed an approach, the domain specification process for generating domain-specific patterns. The domain specification process applies an operator implemented in SHACL to schema.org vocabulary, in order to tackle the issues that come with its generic nature and make it more suitable for specific domains. We presented our approach by defining the domain specification process and with a running example of domain specification operators based on SHACL. We demonstrated the utility of our approach via various use cases. The common benefit of machine-understandable domain-specific patterns observed in all use cases is to have a rather standardized way of encoding the domain expert's knowledge for publishing annotations. The patterns can support the generation of user interfaces and mappings for annotation creation, as well as verification of annotations against a specification. Thanks to the adoption of (de-facto) standardized semantic technologies, the patterns are reusable and interoperable, which is very important, especially for providing common schemas for a domain such as tourism, where many different data providers and application developers are involved. This brings a considerable advantage over non-semantic approaches such as ad-hoc excel sheets and vendors-specific non-reusable solutions. Given the concrete scenarios and the results of the preliminary[41] user studies, the domain-specific patterns and their tools address an important challenge standing in front of the publication of semantic data by bridging the gap between data publishers and domain experts, as well as the application developers.

For the future work, we will work on expanding the application of domain-specific pattern patterns on verification of instances in knowledge graphs [5]. We will work on the formalization of different types of relationships (e.g., subsumption) between domain-specific patterns. Additionally, we will address a second dimension alongside domain, namely the task dimension with the patterns of schema.org. We will use such patterns to make restrictions and extensions of schema.org to make it suitable for certain tasks such as Web API annotation. Another interesting direction to go would be extracting domain-specific patterns from Knowledge Graphs by using the graph summarization techniques (see [9, Sect. 3.1.3] for details).

References

1. Akbar, Z., Kärle, E., Panasiuk, O., Şimşek, U., Toma, I., Fensel, D.: Complete semantics to empower touristic service providers. In: Panetto, H., et al. (eds.) OTM 2017. LNCS, vol. 10574, pp. 353–370. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69459-7_24

[41] Larger studies will be conducted in the future work.

2. Bangor, A., Kortum, P., Miller, J.: Determining what individual SUS scores mean: adding an adjective rating scale. J. Usability Stud. **4**(3), 114–123 (2009)
3. Brooke, J.: SUS-a quick and dirty usability scale. Usability Eval. Ind. **189**(194), 4–7 (1996)
4. De Meester, B., Heyvaert, P., Dimou, A., Verborgh, R.: Towards a uniform user interface for editing data shapes. In: Ivanova, V., Lambrix, P., Lohmann, S., Pesquita, C. (eds.) Proceedings of the 4th International Workshop on Visualization and Interaction for Ontologies and Linked Data. CEUR Workshop Proceedings, vol. 2187, October 2018. http://ceur-ws.org/Vol-2187/paper2.pdf
5. Fensel, D., et al.: Knowledge Graphs - Methodology, Tools and Selected Use Cases. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-37439-6
6. Gray, A.J., Goble, C.A., Jimenez, R., et al.: Bioschemas: from potato salad to protein annotation. In: International Semantic Web Conference (Posters, Demos & Industry Tracks), Vienna, Austria (2017)
7. Guha, R.V., Brickley, D., Macbeth, S.: Schema.org: evolution of structured data on the web. Commun. ACM **59**(2), 44–51 (2016). http://dblp.uni-trier.de/db/journals/cacm/cacm59.html#GuhaBM16
8. Hammar, K.: Content Ontology Design Patterns: Qualities, Methods, and Tools, vol. 1879. Linköping University Electronic Press, Linköping (2017)
9. Hogan, A., et al.: Knowledge graphs, March 2020. http://arxiv.org/abs/2003.02320
10. Knublauch, H., Kontokostas, D. (eds.): Shapes Constraint Language (SHACL). World Wide Web Consortium, July 2017. https://www.w3.org/TR/shacl/
11. Kontokostas, D., et al.: Test-driven evaluation of linked data quality. In: WWW 2014 - Proceedings of the 23rd International Conference on World Wide Web, pp. 747–757 (2014)
12. Patel-Schneider, P.F.: Analyzing schema.org. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 261–276. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_17
13. Prud'hommeaux, E., Labra Gayo, J.E., Solbrig, H.: Shape expressions: an RDF validation and transformation language. In: Proceedings of the 10th International Conference on Semantic Systems, pp. 32–40. ACM (2014)
14. Şimşek, U., Kärle, E., Holzknecht, O., Fensel, D.: Domain specific semantic validation of schema.org annotations. In: Petrenko, A.K., Voronkov, A. (eds.) PSI 2017. LNCS, vol. 10742, pp. 417–429. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74313-4_31
15. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Fox, M., Poole, D. (eds.) Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, 11–15 July 2010. AAAI Press (2010)