IOPE: Interactive Ontology Population and Enrichment Guided by Ontological Constraints (revised version from WISE'21)

S Baghernezhad-Tabasi, L Druette, F Jouanot, C Meurger, M Rousset

IOPE: Interactive Ontology Population and Enrichment Guided by Ontological Constraints

S. Baghernezhad-Tabasi[1] L. Druette[2] F. Jouanot[1] C. Meurger[2] MC. Rousset[1,3]

[1] Université Grenoble Alpes CNRS, LIG
[2] Université Claude Bernard Lyon 1, SAMSEI
[3] Institut Universitaire de France Paris

Abstract

In this paper, we focus on the construction of specialized ontologies that capture skills of experienced experts in a particular domain with the goal to share them with a larger community of trainees or less experienced experts in the domain. Our main contribution is the automatic construction of a Graphical User Interface (GUI), named IOPE*, built from the ontological constraints of an input ontology, as the support of the controlled update process of the considered ontology. The resulting GUI functions as a guidance for the experts with no knowledge of OWL/RDFS, which enables them to easily explore and update their ontologies. We illustrate the functionality of* IOPE *on an ontology for simulation-based medical workshops called* ONTOSAMSEI*. In an extensive set of experiments, we discuss the effectiveness and efficiency of our proposed approach in a specialized medical domain.*

Keywords

Ontology Engineering, Knowledge Acquisition, Automation Form Generation, Simulation-based Training in Medecine.

1 Introduction

Ontologies are the backbone of many information systems that require access to structured knowledge. By their very nature, real world ontologies are dynamic artifacts that evolve both in their structure (the data model) and their content (instances). Keeping them up-to-date is a critical operation for most applications which rely on semantic Web technologies. Ontology updates encompass both *enrichment* and *population*. Ontology enrichment is the task of extending an existing data model of an ontology with additional concepts and semantic relations, while ontology population is the task of adding new instances of concepts to the ontology, using domain documentations. Ontology updates are typically performed in an exploratory and manual fashion, as the non-documented knowledge of the domain expert is required to be taken into consideration. However, these manual updates put burden on the experts and render the whole ontological ecosystem inefficient. In this paper, we advocate for an alternative and more effective approach, and propose to handle updates automatically through a few interactions with the expert, using a Graphical User Interface (GUI).
The challenges associated to interaction-based automatic updates are two-fold:

- While ontologies are typically represented in the form of graphs, it is inherently difficult and counterintuitive to provide a graphical graph-based representation of ontologies for the consumption of experts. While there exist several methods to visualize a graph structure [10, 8], the outcome is often hard to digest by domain experts.
- It is unclear how experts should perform ontology updates through the interactions, without the prior knowledge of the formal syntax and the semantics of ontology languages.

In this paper, we propose IOPE (Interactive Ontology Population and Enrichment), a framework for the automatic construction of a Graphical User Interface (GUI) consisting of *pre-filled Web pages*. We leverage Web pages as a natural interaction means to tackle the challenge of counterintuitive ontology representations. IOPE generates the Web pages from *ontological constraints*, which support the controlled update process of a given ontology, and prefills the generated pages. While IOPE is generic and can be applied to ontologies from a variety of domains, we employ an ontology called ONTOSAMSEI [4] as a use case, whose content helps the domain experts design teaching units for learning skills in simulation-based Medicine.
ONTOSAMSEI is a hierarchy of classes and properties enriched by ontological constraints on those classes and properties, that convey the constraints that will have to be fulfilled by their future sub-classes, sub-properties or instances. ONTOSAMSEI contains 30 different types of simulation sessions formalized with properties such as the target audience, the aimed objectives, the prerequisites, the resources required (human, consumable, simulator, material), as well as the evaluation mode of prerequisites and objectives, to name a few. All these sessions have been defined with the help of many formal documentations, expert and teacher interviews, and analyses on all information from a teaching ingineer. ONTOSAMSEI is availabe in Perscido dataset storage: *https://perscido.univ-grenoble-alpes.fr/datasets/DS352*. In this paper,

we show how to exploit these ontological constraints as a source of guidance for (possibly less experienced) educators willing to design their own simulation sessions, hence addressing the challenge of expert noviceship. ONTOSAMSEI's IOPE GUI is accessible via the following link: *http://iope.tabasi.info*.

The paper is organized as follows. Section 2 describes the formal background of the ontologies that we consider. Section 3 describes our methodology for the automatic construction of a GUI from an input ontology, and its usage for guiding its update (population and enrichment). Section 4 summarizes the evaluation conducted to assess the added value of the GUI for ontology updating. Section 5 is dedicated to related work, and Section 6 concludes the paper.

2 Formal Background

An ontology is a shared formalization of a domain of interest based on a structured vocabulary made of classes, properties and instances. Ontological constraints are declared on classes and properties to constrain their formal semantics to fit with their actual meaning in the domain of application. Then, factual statements can be added to describe specific entities as instances of classes with specific values for some properties. The ontological constraints are defined in RDFS[1] and OWL[2], and described as RDF graphs.

2.1 RDF Format

Let I, L and B be countably infinite pairwise disjoint sets representing respectively *IRIs*, *literals* and *blank nodes*. IRIs (Internationalized Resource Identifiers) are standard identifiers used for denoting any Web resource involved in RDF statements. A literal is a string that represents a specific value for some properties. A blank node represents an anonymous resource (either a literal or an IRI) that can have a local identifier such as *_:b1*.

An ontology in RDF (a.k.a. a knowledge graph) is a set of (factual or ontological) statements expressed as triples $(s, p, o) \in (I \cup B) \times I \times (I \cup L \cup B)$ that form a graph whose nodes are IRIs, blank nodes or literals.

2.2 Ontological Constraints

The ontological constraints that we consider are RDFS constraints and some OWL constraints (displayed in Table 1). These constraints form the main core for the most of domain ontologies and are sufficient for our application domains. Adding new ontological constraints will be studied in future works.

Figure 1 displays part of the specialization hierarchies of properties and classes resulting from RDFS ontological constraints declared in ONTOSAMSEI.

Figure 2 shows the RDF graphs associated to two constraints declared in the ontology on the property

[1]Resource Description Framework Schema (RDFS): https://www.w3.org/TR/rdf-schema

[2]Web Ontology Language (OWL): https://www.w3.org/TR/owl-features

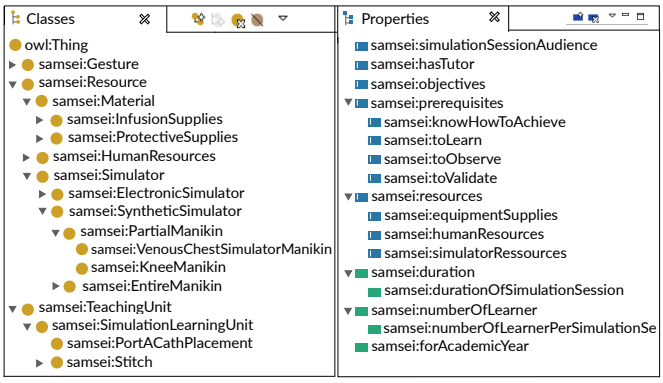

Figure 1: A part of hierarchy of properties and classes in ONTOSAMSEI

`samsei:equipmentSupplies` for the class `samsei:PortACathPlacement`, which is a particular type of simulation learning unit that trains students to place a port or a catheter. The constraint graph in Figure 2(a) expresses that `samsei:sterilecompress` (which is an instance of Bandage material) is declared in the ontology as a mandatory value of the property `samsei:equipmentSupplies`. The constraint graph depicted in Figure 2(b) expresses that at least one equipment of type `samsei:protectiveSupplies` is mandatory for simulating a placement of a port or a catheter.

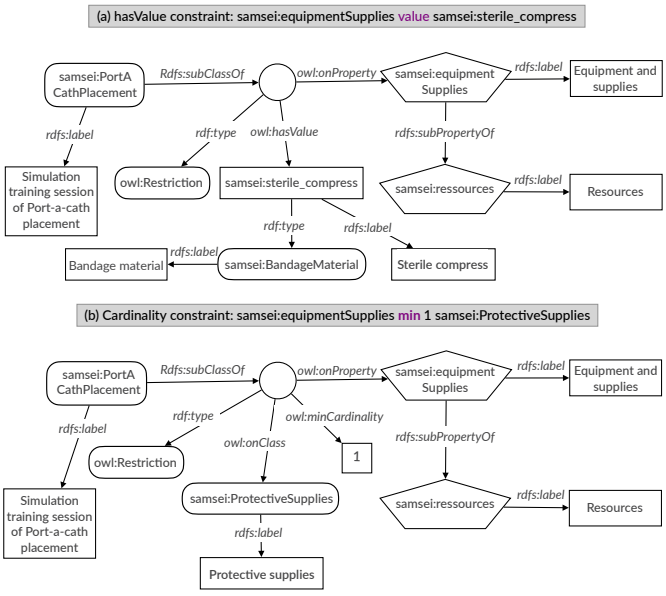

Figure 2: Two constraint graphs (a) and (b) on the property `equipmentSupplies` for the class `PortACathPlacement`

3 Interactive Ontology Update

Our approach consists of transposing the RDF data and the ontological constraints of a given domain ontology into a graphical user interface (GUI) named IOPE GUI.

Table 1: RDFS and OWL constraints considered in this paper

Type	Shortened syntax	Semantics
Class specialization	(C rdfs:subClassOf D)	$\forall$ i ((i rdf:type C) $\Rightarrow$ (i rdf:type D))
Property specialization	(p rdfs:subPropertyOf q)	$\forall$ i $\forall$ j ((i p j) $\Rightarrow$ (i q j))
Domain restriction	(p rdfs:domain C)	$\forall$ i $\forall$ j ((i p j) $\Rightarrow$ (i rdf:type C))
Range restriction	(p rdfs:range D)	$\forall$ i $\forall$ j ((i p j) $\Rightarrow$ (j rdf:type D))
Value restriction	(C p owl:hasValue v)	$\forall$ i ((i rdf:type C) $\Rightarrow$ (i p v))
Alternative values restriction	(C p owl:oneOf [v1, ..., vn])	$\forall$ i ((i rdf:type C) $\Rightarrow \bigvee_{k\in[1..n]}$ (i p vk))
Cardinality restriction	(C p min k D)	$\forall$ i ((i rdf:type C) $\Rightarrow \exists o_1, \ldots o_k (\bigwedge_{i,j\in[1..k]} o_i \neq o_j \wedge \bigwedge_{j\in[1..k]}$ (o_j rdf:type D) $\wedge$ (i p o_j))

It functions as a guidance for domain experts to easily explore the ontology and update it through interactive graphical widgets. The input entered by domain experts through the IOPE GUI is transformed into RDF triples that must be verified by a specialist in knowledge management, to maintain ontology correctness, before being added effectively in the domain ontology. Figure 3 is an overview of our interactive IOPE system.

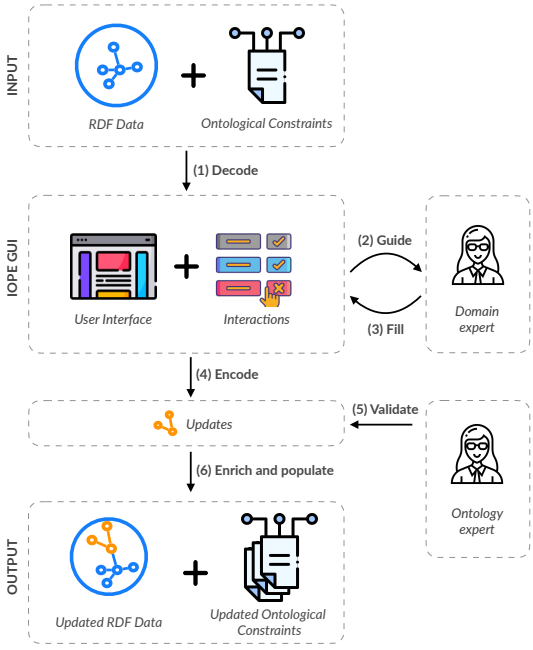

Figure 3: Overview of IOPE workflow

The *Web form templates* on which the IOPE GUI is built are described using a Web form ontology called IOPE_Web that we have developed by adapting RaUL [9].

3.1 The IOPE_Web Ontology

The IOPE_Web ontology is shown in Figure 4. It is organized around 4 main classes `IOPE:Page`, `IOPE:PageLayout`, `IOPE:Container` and `IOPE:Widget` related by properties for modeling Web pages which are structured in containers with widgets and are associated with page layouts. The widgets act as a direct point of user interaction and provide access to the triples of the referenced RDF graph. The interaction with users can be done using several types of widgets such as LABEL, TREE-VIEW, LIST BOX, TEXT BOX and CHECK BOX, which leads to as many corresponding subclasses of the main class `IOPE:Widget`. They inherit of the standard properties for widgets that are described in IOPE_Web as datatype properties such as `IOPE:name`, `:placeholder`, `IOPE:label`, and others. In our setting, the `IOPE:dataSource` property is used to assign an input data from a domain ontology that can be of type *xsd:string*, simple or nested list *IOPE:list*, or *owl:Thing*. The `IOPE:value` property is filled by the output value of the widget provided by a user. The boolean properties `IOPE:hidden`, `IOPE:multiple`, `IOPE:readonly` and `IOPE:onclick` are similar to HTML form attributes. A widget with the `IOPE:required` property as "True" will be rendered by a red asterisk to specify that the widget must be filled in by the user.

Widgets can be grouped in a Web page within containers that can be nested using the `IOPE:partOf` property. In our setting, different types of specific containers are declared as subclasses of `IOPE:Container` to express that the different types of ontological constraints in our setting will be rendered in a specific manner in the IOPE GUI.

3.2 Ontology-Based GUI Construction

We have followed a declarative approach based on a set of *mapping rules* to generate automatically pre-filled Web pages, and on a set of *binding rules* to generate RDF graphs from entered values by users via widgets.

Input: The input of GUI construction is a domain ontology in which the ontological constraints have been automatically saturated by a reasoning algorithm as detailed in [5].

Initialization: The GUI construction is initiated with the choice of one class of interest in the ontology by the user, which is called the *focus class* F.

The set $Constraints(F)$ of the ontological constraints associated to the focus class F is decomposed in groups $Group(P, F)$ of all the constraints involving sub-properties of a given property P.

For the focus class F, and for each group of properties $Group(P, F)$, an instance of a Web page is created with the page layout depicted in Figure 5, which sets up the organization within the page of the specific containers dedicated to the different types of constraints on sub-properties p of P for which there exists constraints in $Group(P, F)$.

The following instances of the *IOPE:Container* class are

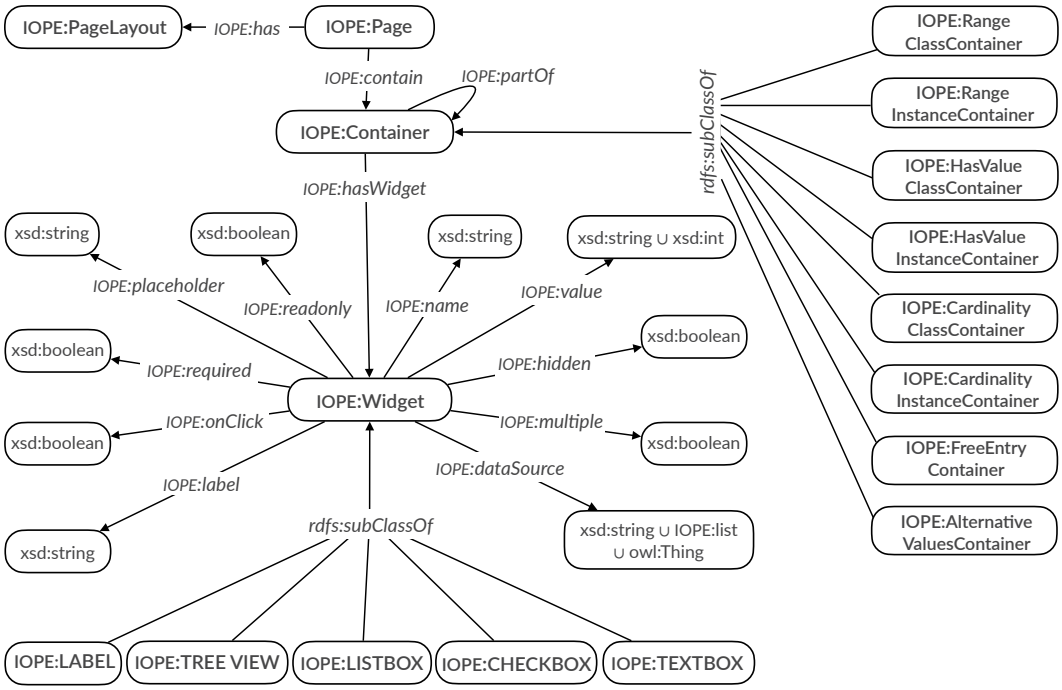

Figure 4: The IOPE Web form ontology

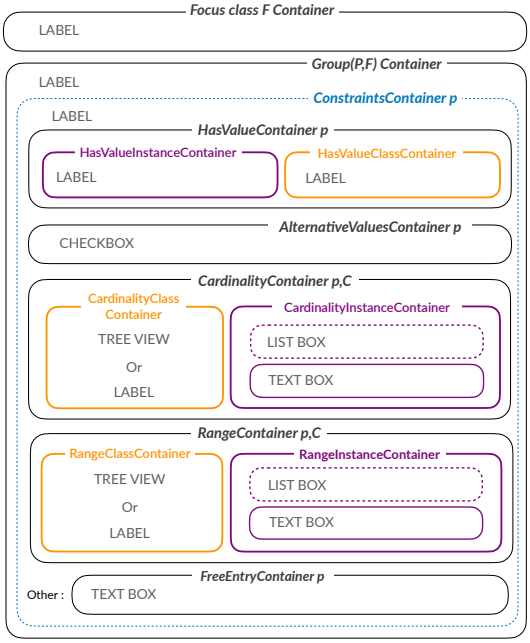

Figure 5: Empty web page prepared for the rendering of constraints of the focus class F for each property p which is a specialization of a same property P.

created, with their position in the empty Web page shown in Figure 5:

- *IOPE:FocusClass F Container* denotes the main container of the created Web page,
- *IOPE:$Group(P, F)$Container* denotes the container that will group all the containers corresponding to the constraints holding for F on sub-properties of P ,
- *IOPE:ConstraintContainer p* denotes the container that will display the restrictions of F on property p,
- *IOPE:HasValueContainer p* denotes the container that will display the hasValue restrictions of F on property p,
- *IOPE:AlternativeValuesContainer p* denotes the container that will display the alternative values restrictions of F on property p,
- *IOPE:CardinalityContainter p, C* denotes the container that will display the cardinality restrictions of F on property p and class C,
- *IOPE:RangeContainter p, C* denotes the container that will display the range restrictions of F on property p which is class C,
- *IOPE:FreeEntryContainer p* denotes the container for the user to add new classes involved in cardinality restrictions for the property p.

Then mapping rules are triggered for mapping components of each ontological constraints to the visual widgets in the prepared containers in order to fill each Web page guided by the ontology.

Mapping rules: Each mapping rule has a constraint graph pattern in its left-hand side and a IOPE_Web graph pattern in its right-hand side. The constraint graph pattern expresses a particular ontological constraint on the focus class and can be instantiated via the input data. The corresponding instantiation of IOPE_Web graph pattern in

the right-hand side is a specification using the vocabulary of the IOPE_Web ontology of how to fill the corresponding widgets and containers in the corresponding Web page.
The mapping rules can be triggered in a forward-chaining manner and in any order. The resulting IOPE_Web graph provides the full RDF specification of the pre-filled Web pages that have to be created for the focus class chosen by the user. The effective implementation of the mapping rules is implemented using RDFLib and JSON libraries in Python 2.7.16 language. Our code is publicly available in [1].
The set of mapping rules are given in a companion report [5]. Here, we just give two of them in their instantiated form for clarity purpose.
Figure 6 shows a mapping rule for a value restriction `(F p value v)`.
The specific container *IOPE:HasValueContainer* p is decomposed in two sub-containers defined as blank nodes whose types are *IOPE:HasValueInstanceContainer* and *IOPE:HasValueClassContainer* respectively. For these two sub-containers, widgets of type *IOPE:LABEL* are created as blank nodes with the property *IOPE:dataSource* filled by the corresponding labels of v and its class C in the domain ontology. The property *IOPE:required* is set to "True" for the first widget to refer that the value v is mandatory for the property p.
Figure 7 shows a mapping rule for a cardinality restriction `(F p min n C)` such that $n > 0$, where `C` has a hierarchy of sub-classes and a list of instances in the domain ontology.
The specific container *IOPE:CardinalyContainer* p, C is decomposed in two sub-containers defined as blank nodes whose respective types are:

- *IOPE:CardinalityClassContainer*
- *IOPE:CardinalityInstanceContainer*

For the former, a widget of type *IOPE:TREEVIEW* is created as a blank node with the property *IOPE:dataSource* filled by the tree view of $subClasses(C)$, which denotes the hierarchy of the sub-classes of C in the domain ontology enriched with an additional item $Other\_C$. The property *IOPE:required* and *IOPE:onClick* are set to "True" for this widget to indicate that entering at least one value is mandatory for the property p, and this widget supports the interaction with users to display interactively the sub-class hierarchy.
For the latter, a widget of type *IOPE:LISTBOX* is created as a blank node with the property *IOPE:dataSource* filled by the list $instances(C)$ of instances of the class C, the *IOPE:label* property set to "select existing item(s) or enter new item(s)" and the *IOPE:hidden* property set to "True" to make the widget invisible until the first interaction of the user through the widget of type *IOPE:TREEVIEW*. A widget of type *IOPE:TEXTBOX* is also created with the *IOPE:placeholder* property set to the value "Enter the new item(s) (separated by a comma)" in order to give users possibility to enter new instances.
Figure 8-left shows one of the resulting pre-filled Web pages generated by the HTML implementation of the IOPE_Web specification resulting from the application of the mapping rules to the ONTOSAMSEI's ontological constraints.
Figure 8-right shows the effect of a user interaction through the widget of type the *IOPE:TREEVIEW* to select the sub-class *DisposableDrape* from the *ProtectiveSupplies* sub-class hierachy. As a result of this interaction, the instance container corresponding to the selected sub-class becomes visible to let the user select an instance or enter a new one. User interaction are guided by constraints on properties based on the mapping rules, which allow to check cardinality, domain and codomain.
The input entered by the user must be bound to RDF data corresponding to new instances or new constraints submitted to populate or enrich the domain ontology. This binding mechanism is based on a set of *binding rules* that are triggered on IOPE_Web graphs to generate RDF graphs built on the domain ontology.
Binding rules: Each binding rule is defined as a diagram with a IOPE_Web graph on the left side and the corresponding generated triples in the form of RDF graph on the right side. The binding rule shown in Figure 9 is triggered when a focus class F is chosen. This rule simply creates an instance f of the focus class F.
The other binding rules are triggered when the `IOPE:value` property is filled by an input provided by the user through an interactive widget.
Figure 10 shows the binding rule for the textbox widget in the free entry container of a property p for the focus class F. Its application generates a new constraint graph expressing a new cardinality constraint for F on the property p and a new class.
The other binding rules are given in [5].

4 Evaluation

The objective of our study is to evaluate the *efficiency*, the *users' satisfaction* and the *effectiveness* of the IOPE interface with the purpose of populating and enriching the ONTOSAMSEI ontology. The users involved in our user study are a subgroup of 22 experts in simulation-based training in Medicine. They are domain experts but they are not familiar with RDF and OWL. The user study was organized in two steps for each expert. In the first step, the expert logs in the system with her credentials, picks one simulation training session, and begins to observe and update the information in the pre-filled Web pages. In the second step, she will be transferred to a survey form to evaluate some qualitative aspects of IOPE and ONTOSAMSEI ontology and reflect her viewpoint based on her interactions with the IOPE interface.

4.1 Evaluation of the IOPE GUI Efficiency

We first provide quantitative results on the time spent by users and their number of interactions with IOPE . Then, we compare these results with the time insight perceived by users and with the number of interactions required for

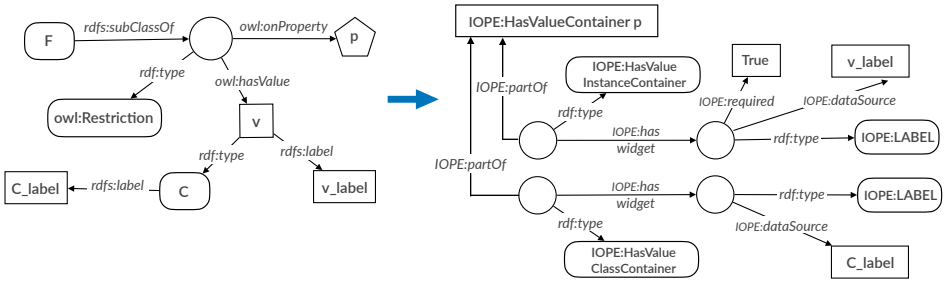

Figure 6: Mapping rule for a value restriction (`F p value v`) where `v rdf:type C`

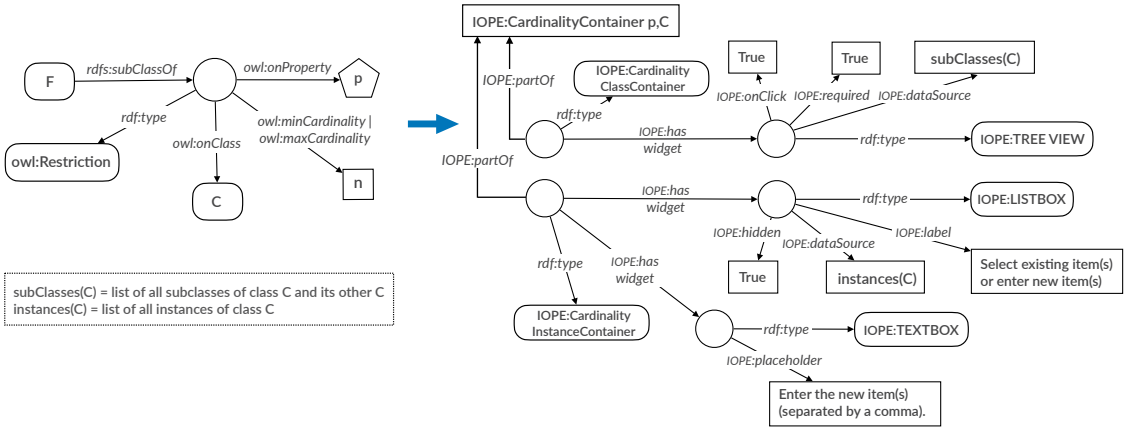

Figure 7: Mapping rule for a Cardinality constraint where subClasses(C) and instance(C) are not empty

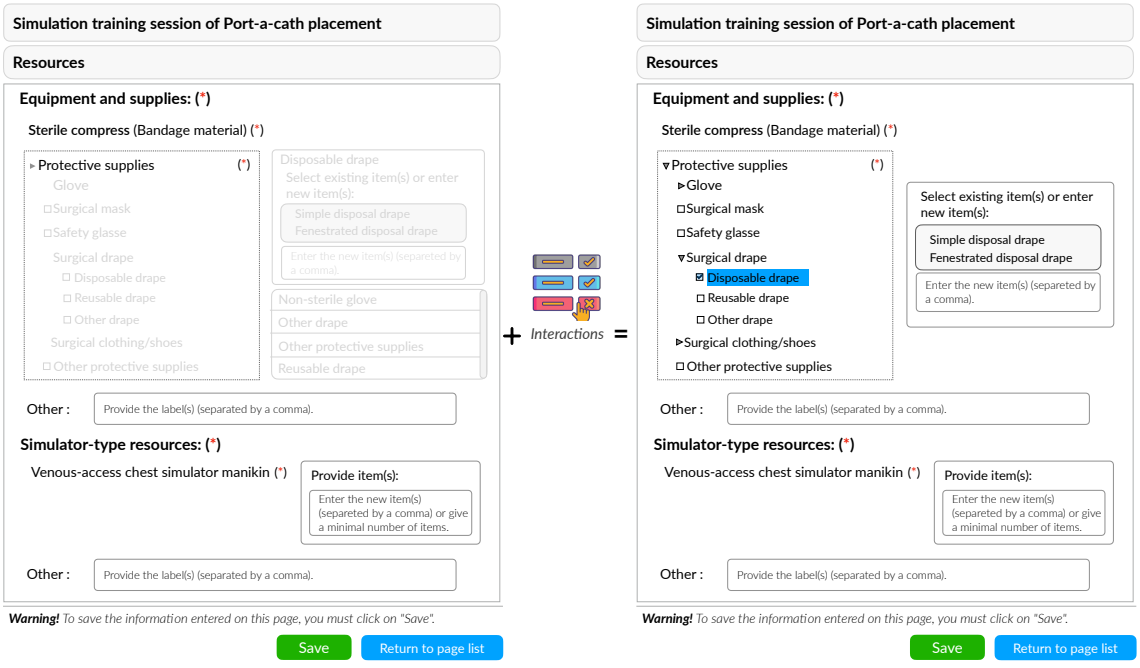

Figure 8: Left: HTML Web page generated from the outcome of the application of mapping rules on OntoSAMSEI ontological constraints. Right: HTML Web Page changes after user interaction by the widgets.

the same tasks when interacting with a standard ontology editor.

4.1.1 Time Spent by Users and Number of Interactions.

Each expert spent 163 seconds (2.72 minutes) on average, maximum 320 seconds (5.33 minutes), minimum 67

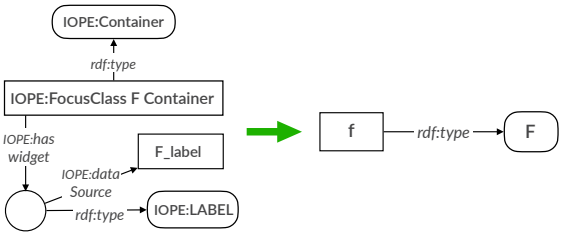

Figure 9: Binding rule for a focus class F

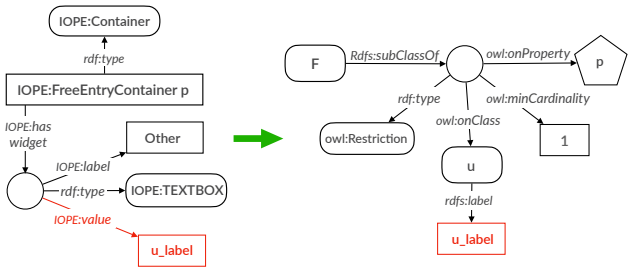

Figure 10: Binding rule for free entry container on property p and a focus class F

seconds (1.12 minutes). On average, their number of interactions with IOPE is 5.78, with a maximum of 14 and a minimum 3. The majority of interactions are with CHECK BOX widget (56.15%) followed by TEXT BOX widget (32.30%) and LIST BOX widget (11.53%). Table 2 shows the distribution of experts in two categories of groups. In terms of number of interactions, we have built the groups of "prolific" experts (having more than 6 interactions with IOPE), "active" experts (having between 3 and 6 interactions), and "moderate" experts (with less than 3 interactions). In terms of interaction duration, we have built the groups of experts spending "short-time" (less than 2 minutes), "medium-time" (between 2 and 4 minutes), and "long-time" (more than 4 minutes). Table 3 reports the distribution of time groups for each interaction activity groups. We notice that more interactions do not necessary yield to more time spent to interact. This shows that IOPE helps experts to fulfill their task in a reasonable amount of time, even for prolific experts.

4.1.2 Time-to-Insight Users's Evaluation.

After they are done with using the IOPE interface for fulfilling their task, we ask the experts the following question to estimate the time-to-insight for a future interaction with IOPE : *"how much time do you expect to take for setting up a new simulation training session with* IOPE*?"*. The response is in the form of a Likert scale from

Table 2: Distribution of expert groups

	moderate	active	prolific
Expert population	22.73%	50%	27.27%
	short-time	**medium-time**	**long-time**
Expert population	50%	31.82%	18.18%

Table 3: Distribution of interaction time groups for interaction volume groups.

		Interaction volume groups		
		moderate	**active**	**prolific**
Interaction time groups	**short-time**	0.80	0.46	0.33
	medium-time	0.00	0.27	0.67
	long-time	0.20	0.27	0.00

1 to 5 where "1" means "very short time" and "5" means "very long time".

Figure 11 shows the results. We observe that the majority of experts chose "short time" and "average time", i.e., options 2 and 3 in the Likert scale. Moreover, prolific experts and long-time perceive shorter expected time compared to the active and moderate experts. A possible interpretation is that more interactions and more time sent interacting with the system boosts the perception of faster delivery of required information.

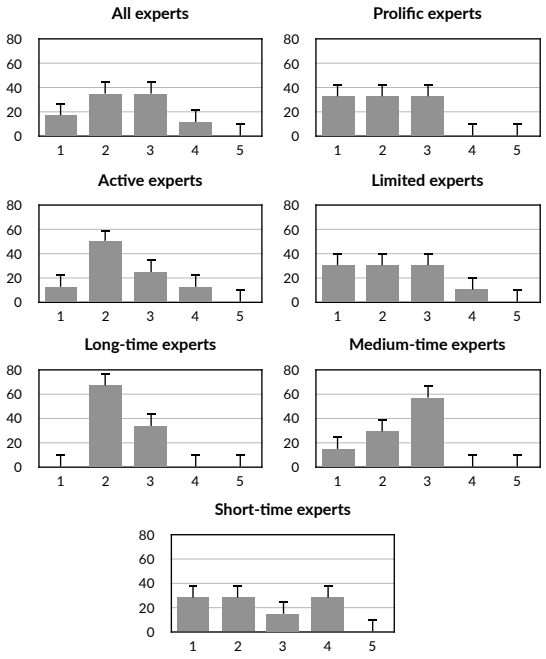

Figure 11: Time-to-insight results.

4.1.3 Comparative Efficiency of IOPE with a Standard Ontology Editor.

The goal of this experiment was to measure the added-value of IOPE compared to a standard ontology editor such as TOPBRAID [2], in terms of number of interactions required to fulfill edition tasks mentioned in Table 4. The tasks are categorized into three levels of difficulty based on [7]. TOPBRAID is a major IDE for knowledge graph management. It can play the role of a rich model, constraint, data and queries editing tool, and of a powerful querying and reasoning flexible. It is intensively used by many actors in the area of ONTOSAMSEI.

For the fairness of this experiment, since none of the

Table 4: Tasks descriptions

Task	**Description** (Given the simulation training session X ...)
Easy	Fill the number of trainees for X
Medium	Fill the target audience of X
Difficult	Fill the required resources for X

domain experts have ever used TOPBRAID, the different tasks were fulfilled by the five authors of the paper who have a sufficient knowledge about the domain, as well as a sufficient experience using both IOPE and TOPBRAID. Figure 12 shows the average number of interaction steps to fulfil those tasks in IOPE and TOPBRAID. We observe

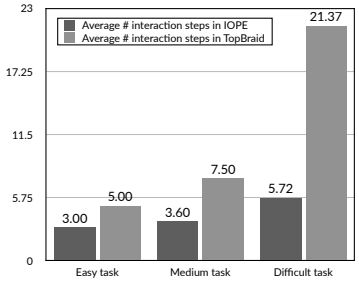

Figure 12: Comparative number of interactions between IOPE and TOPBRAID.

that for both tools, the number of interaction steps increases with the difficulty of the tasks. However, the IOPE's trend grows from average 3.00 steps for an easy task to average 5.72 steps for a difficult task, while using TOPBRAID grows from average 5.00 steps for an easy task to average 21.00 steps for a difficult task. This shows that IOPE , by weaving relevant information together using constraints, enables the experts to fulfill their tasks more rapidly than by using a standard editor.

4.2 Evaluation of IOPE Users' Satisfaction

We have measured on a Likert scale in the range 1 to 5 the assessment by users of three aspects of satisfaction, namely *utility*, *usability*, and *adoption*, through the questions of the three first rows of Table 5.

The aggregated results are shown in the three first column of Figure 13.

Utility. 82.35% of the participants have a positive view on the utility of IOPE. However, the prolific experts appreciate the utility more than active experts. This shows that more interactions increases the perception of utility, which is also confirmed by long-time experts who are entirely on the positive spectrum.

Usability. Overall, the experts perceived usability positively. However, there is a vivid contrast between moderate experts versus active and prolific experts, where the former group seems to not enjoy the usability of IOPE. We conjecture that moderate experts got lost early in the process, and abandoned their task. There is also a subset of long-time experts who assessed low usability. They probably spent too much time to fulfil their tasks and got lost in the process also.

Adoption. The choice over adoption is from 1 to 5, where 1 means "never" and 5 means "always". Most of the experts voted to adopt IOPE in the future.

4.3 Effectiveness of IOPE for Enriching the ONTOSAMSEI Ontology

In this part of the experiment, we measure the expert's assessment of *accuracy* and *completeness* of the ONTOSAMSEI ontology through its presentation to the experts by IOPE GUI. We do it by asking the experts the questions in the two last rows of the Table 5. The aggregated results (on the Likert scale from 1 to 5) are shown in the two last columns of Figure 13.

Accuracy. The majority of the participants are positive on accuracy, while 11.76% are negative. Short-time and moderate experts express more negative votes on accuracy compared to long-time and prolific experts, respectively. This is presumably because less investigations in the former groups did not enable them a precise view of the ontology.

Completeness. 76.46% of the participants find ONTOSAMSEI complete enough. However, prolific experts appreciate completeness less than the overall population. We found out that they prominently interact with text-boxes, which shows that they use IOPE to effectively enrich the ontology. The entire long-time expert group votes positively, which means that spending more time to go into the details of the simulation training sessions convinces them of their completeness.

5 Related Work

In the literature, ontological updates are often performed using ontology editing tools, such as PROTÉGÉ [14], TOPBRAID [2], and ONTODIA [13]. However, these systems require a basic understanding of the RDF notation and of the OWL semantics to edit the ontology consistently. Graph-based editing approaches alleviate this limitation by leveraging shapes graphs in the form of SHACL standard[3] [20, 18]. While shapes graphs are well adapted for editing complex data, they require the definition of such graphs for each ontology. In contrast, IOPE abstracts all RDF/OWL technicalities and seamlessly enforces the ontological constraints as a strong guidance for the experts to update the ontology, using the pre-filled forms.

WebVOWL [19] is a web application for the interactive graph-based visualization of ontologies which employs the Visual Notation for OWL Ontologies (VOWL) [11]. However, WebVOWL does not visualize the instances but only the OWL part of a (possibly populated) ontology. Also, the graphs displayed by the tool tend to become quickly illegible when their size increases. In IOPE, we employ Web forms as a more widespread medium for visualizing information, and we support the update of instances and of ontological constraints.

Forms are also used in [12] in a nested structure to capture

[3] *Shapes Constraint Language (SHACL): https://www.w3.org/TR/shacl/*

Table 5: Measure definitions and corresponding questions asked in the survey.

Measures	Definition	Question asked in the survey
utility [17, 3]	The usefulness of the method to fulfil a given task.	How do you evaluate the utility of IOPE for setting up simulation training sessions?
usability [16, 3]	The easiness of interactions with the method	To which degree do you find IOPE easy-to-use?
adoption [17]	The usefulness of the method for future similar tasks	How often will you employ IOPE for setting up and describing a new simulation training session in the future?
accuracy [15, 16]	The precision of information based on expert's prior knowledge.	How do you evaluate the accuracy of IOPE's pre-filled information for describing simulation training sessions?
completeness [16]	The retrieval exhaustiveness of the necessary and required information.	How do you evaluate the sufficiency of IOPE's pre-filled information for describing simulation training sessions?

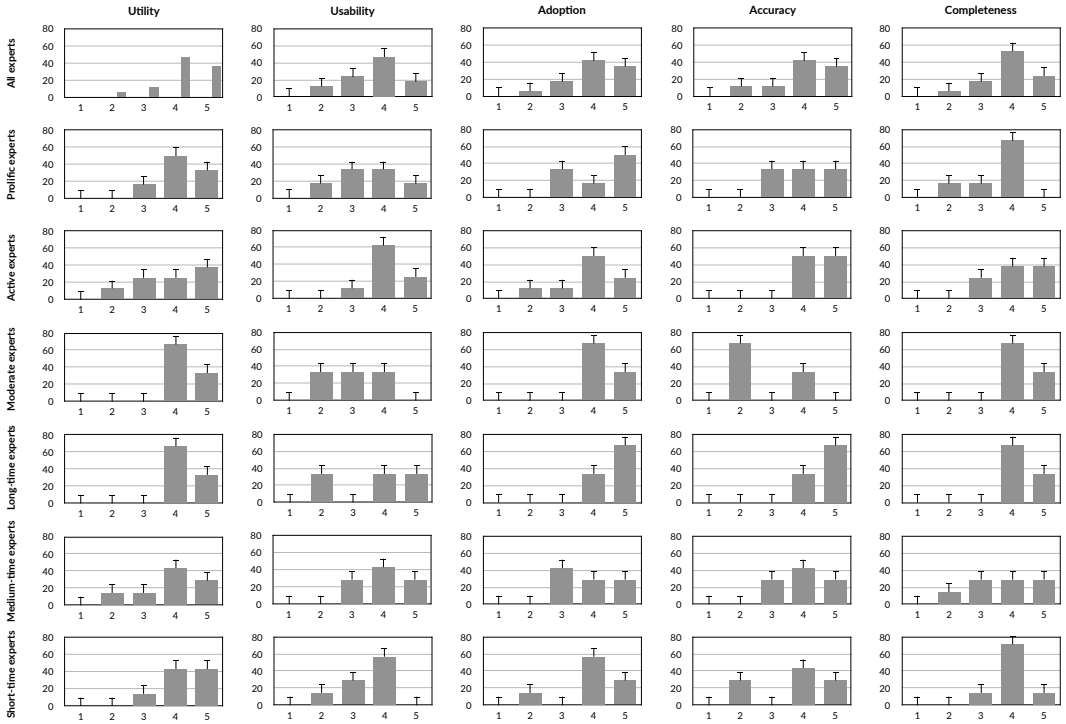

Figure 13: Satisfaction and effectiveness metrics results.

relational aspects of knowledge graphs and update RDF data. However, the nested structure introduces increasing complexity and hence lacks intuitiveness. Moreover, the focus in [12] is solely on the population part and the approach does not extend to OWL constraints.

In [6], Web forms are generated from ontologies (using a User Interface ontology, called RaUL) by interpreting ontology assertions as rules. While the approach only incorporates individual assertions (ontology population), IOPE serves both ontology enrichment and population, through interactions with the experts. IOPE stresses on ontological constraints as first-class citizens and renders pre-filled forms to provide a more aggregated view for the experts, which is, to the best of our knowledge, nonexistent in the literature.

In [21], Web forms generation are very close to ActiveRaul approaches. Constraints are expressed using SHACL language as SHACL shapes. The SchÍmato application manages the web form generation, the end-user interaction and the ontology engineer enrichement. While end-user interactions are limited to add and update instances of a knowledge graph, IOPE allows end-users to enrich the knowledge graph by updating classes and some constraints on data. IOPE has been applied and evaluated on a complex domain and is able to generate a sequence of web pages that structures the knowledge graph for the end-users, and not only a simple web form generator.

6 Conclusion

In this paper, we have presented the interactive IOPE framework for enrichment and population of specialized ontologies.Given any input ontology, IOPE exploits the ontological constraints and a set of mapping rules to generate a set of user-friendly Web pages which assist the experts in editing the ontology. Binding rules are then

used to derive the RDF graphs corresponding to the updates entered by the experts. We have conducted an extensive set of experiments on the domain of simulation-based medical education, for measuring IOPE's efficiency, effectiveness, as well as the experts' satisfaction in fulfilling their tasks using IOPE . In the future, we plan to improve the *explainability* of IOPE to reduce the number of abandoned editing tasks and increase its usability by domain experts not familiar with ontology engineering.

References

[1] IOPE implementation. https://github.com/shadi-tabasi/IOPE.git.

[2] Topquadrant topbraid composer. https://www.topquadrant.com/products/topbraid-composer/. Accessed: 2021-01-15.

[3] William Albert and Thomas Tullis. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.

[4] Shadi Baghernezhad-Tabasi, Loic Druette, Fabrice Jouanot, Celine Meurger, and Marie-Christine Rousset. OntoSAMSEI: Interactive ontology engineering for supporting simulation-based training in medicine. In *WETICE*, 2021.

[5] Shadi Baghernezhad-Tabasi, Marie-Christine Rousset, Loïc Druette, Fabrice Jouanot, and Celine Meurger. IOPE: Interactive Ontology Population and Enrichment. Research report, https://hal.archives-ouvertes.fr/hal-03177176, LIG, 2021.

[6] AS Butt, A Haller, S Liu, and L Xie. Activeraul: Automatically generated web interfaces for creating rdf data. *Semantic Web*, 2013.

[7] Evanthia Dimara, Steven Franconeri, Catherine Plaisant, Anastasia Bezerianos, and Pierre Dragicevic. A task-based taxonomy of cognitive biases for information visualization. *IEEE Trans. Vis. Comput. Graph.*, 26:1413–1432, 2020.

[8] Yixiang Fang, Reynold Cheng, Siqiang Luo, Jiafeng Hu, and Kai Huang. C-Explorer: Browsing communities in large graphs. *VLDB*, 2017.

[9] Armin Haller, Jürgen Umbrich, and Michael Hausenblas. Raul: Rdfa user interface language - A data processing model for web applications. In *WISE*, 2010.

[10] Nathalie Henry, Jean-Daniel Fekete, and Michael J. McGuffin. Nodetrix: a hybrid visualization of social networks. *TVCG*, 2007.

[11] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. Visualizing ontologies with VOWL. *Semantic Web*, 7(4):399–419, 2016.

[12] Pierre Maillot, Sébastien Ferré, Peggy Cellier, Mireille Ducassé, and Franck Partouche. Nested forms with dynamic suggestions for quality RDF authoring. In *DEXA*. Springer, 2017.

[13] D. Mouromtsev, D. Pavlov, Yury Emelyanov, A. Morozov, Daniil Razdyakonov, and M. Galkin. The simple web-based tool for visualization and sharing of semantic data and ontologies. In *International Semantic Web Conference*, 2015.

[14] Natalya Fridman Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen. Creating semantic web contents with protégé-2000. *IEEE Intell. Syst.*, 16(2):60–71, 2001.

[15] Behrooz Omidvar-Tehrani and Sihem Amer-Yahia. Data pipelines for user group analytics. In *SIGMOD Conference*, pages 2048–2053. ACM, 2019.

[16] Protiva Rahman, Lilong Jiang, and Arnab Nandi. Evaluating interactive data systems. *VLDB J.*, 29(1):119–146, 2020.

[17] James J Thomas. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2005.

[18] Andre Valdestilhas, Gustavo Publio, Andrea Cimmino Arriaga, and Thomas Riechert. Voceditor an integrated environment to visually edit, validate and versioning rdf vocabularies. 2020.

[19] Vitalis Wiens, Steffen Lohmann, and Sören Auer. Webvowl editor: Device-independent visual ontology modeling. In *ISWC 2018 Posters & Demonstrations*, 2018.

[20] Jesse Wright, Sergio José Rodríguez Méndez, Armin Haller, Kerry Taylor, and Pouya Ghiasnezhad Omran. Schímatos: A shacl-based web-form generator for knowledge graph editing. In *ISWC*, 2020.

[21] Jesse Wright, Sergio José Rodríguez Méndez, Armin Haller, Kerry Taylor, and Pouya Ghiasnezhad Omran. Schímatos: A shacl-based web-form generator for knowledge graph editing. In Jeff Z. Pan, Valentina A. M. Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal, editors, *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II*, volume 12507 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2020.