Ontology authoring with FORZA

C. Maria Keet
School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal
UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, South Africa
keet@ukzn.ac.za

Muhammad Tahir Khan
Fondazione Bruno Kessler
Trento, Italy
tahirkhan@fbk.eu

Chiara Ghidini
Fondazione Bruno Kessler
Trento, Italy
ghidini@fbk.eu

ABSTRACT

Generic, reusable ontology elements, such as a foundational ontology's categories and part-whole relations, are essential for good and interoperable knowledge representation. Ontology developers, which include domain experts and novices, face the challenge to figure out which category or relationship to choose for their ontology authoring task. To reduce this bottleneck, there is a need to have guidance to handle these Ontology-laden entities. We solve this with a generic approach and realize it with the *Foundational Ontology and Reasoner-enhanced axiomatiZAtion (FORZA)* method, containing DOLCE, a decision diagram for DOLCE categories, part-whole relations, and an automated reasoner that is used during the authoring process to propose feasible axioms. This fusion has been integrated in the MoKi ontology development tool to validate its implementability.

Categories and Subject Descriptors

M.8 [**Knowledge Reuse**]: Miscellaneous; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Knowledge base management; H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Ontology engineering, Foundational Ontology, Part-Whole relation, Q/A decision system, Ontology Alignment, Reasoner

1. INTRODUCTION

The need for effective support for ontology developers—which often include domain experts and novice modellers—in the process of *authoring* OWL ontologies, is increasingly recognised (see e.g., [2, 16, 18]) as a crucial step to make the construction of ontologies more agile and apt to the needs of organisations and business enterprises.

CIKM'13, October 27 - November 01 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505539.

Providing domain experts with this kind of support requires being able to face and solve important challenges that concern the formalisation of knowledge. Among the challenges that need to be faced, is providing assistance to ontology developers in answering a list of crucial questions that arise while writing an ontology: *where do I start?*, *what can I reuse from another ontology?*, *how/where do I add classes?*, and *and how do I relate them?*. These questions usually arise when capturing in a logic the output of an 'intermediate stage' or 'conceptualization', as suggested in METHONTOLOGY [3], and in scenarios 1-8 in the NeOn methodology for networked ontologies [17]. Regarding the latter question, concrete examples of ontology authoring concerns are, e.g.,:

- "Is a Theatre performance an object or some process?" (an event that unfolds in time),
- "should I add Tourist as a subclass of Person, or is Tourist a role that can be played by Person and thus relate the two with an object property expression instead of a subclass axiom?" (role/bearer is ontologically better defensible),
- "How is Mapungubwe National park a proper part of South Africa or is it proper located in South Africa?" (proper located in is more precise),
- "does a tourist participate in a sightseeing tour or is the tourist part of a tour?" (participation, not parthood),

which may be difficult to answer by domain experts and novice modellers without any extra support.

Generic, reusable ontology elements, such as a foundational ontology's categories (e.g., process, physical object, role), domain-independent relations (e.g., part-whole relations, participation), and ontology design patterns [14] in the case of small scenarios, can help ontology developers in answering such questions, improve the quality of the ontology, and also speed up the ontology development process, as shown in [6, 8] for the case of foundational ontologies and domain-independent relations. However, also reusing entities and selecting relations from existing ontologies is a complex task (e.g., [13]). In the case of foundational ontologies, having to delve into the documentation requires a massive effort, often without a clear sight upfront on return on investment. Moreover, assuming that a list of domain concepts (OWL classes) already exist, *how does an ontology developer link them to the foundational ontology?* and *how does an ontology developer relate them to each other using relations from a foundational ontology?*, be this for interoperability or ontology quality purposes. Finally, assuming

that manual informed guesses are performed in order to select and reuse entities and relations, these guesses are only checked for logical consistency after the addition step, making the whole process labour-intensive and highly iterative. Evaluation of OntoPartS tool [8] has shown it provides successful automated guidance in selecting part-whole relations, but also showed a bottleneck in the manual selection of categories from the DOLCE foundational ontology that reduced its effectiveness. Little methodological and tool-based support is given to ontology developers on how (where) to link a domain ontology to a foundational one and/or how to reuse any of the knowledge represented in the foundational ontology. Few exceptions are the paper-based OntoSpec [5], the not publicly available implementation of TMEO [12], and the publicly available OntoPartS tool [6, 8].

This paper aims at tackling these problems in ontology authoring by supporting an intelligent usage of foundational ontologies and other already represented knowledge to help ontology designers during this process. While the method is very general—catering for several combinations of manual design and automated support—the main idea for an intelligent automated support relies on making use of: (i) a question/answering (Q/A) system to support linking the domain ontology with reusable general or foundational ontologies, and (ii) an automated reasoner in supporting the selection of relations from general pre-existing ontologies.

More in detail, the contribution of this paper is presented as a three-step process from the generic to the specific. First, we present a novel general method—GENERATOR: *Guided ENtity reuse and class Expression geneRATOR*—for the reuse of already represented knowledge in such a way that it guides the modeller toward the comparatively best options of possible axioms to add. This general method contains three variable steps to cater for several scenarios: it can be realised manually or automated, with or without a foundational ontology, and with or without linking entities from a domain ontology to a foundational ontology. Second, we instantiate and provide automated support for GENERATOR in a particularly important and general scenario, where a domain ontology has to be linked to the foundational ontology DOLCE [10] in order to reuse a taxonomy of part-whole relations (called mereoTopoD, which is compatible with DOLCE). The resulting instantiation, called FORZA *(Foundational Ontology and Reasoner-enhanced axiomatiZAtion)*, includes automated support for the linking with DOLCE categories based on a novel decision tree to categorise a subject domain class as a subclass of a DOLCE class (named D3), and a novel algorithm that uses an automated reasoner to compute the applicable part-whole relation(s) between the selected classes (named OntoPartS-2), which avoids the common post-hoc checking and instead uses the reasoner to guide the 'trial' phase and reduce errors. FORZA is tailored to meet the characteristics of DOLCE and mereoTopoD, but does not depend upon the specific domain at hand. It can thus be used to foster the reuse of domain-independent elements and relations from DOLCE and mereoTopoD in any concrete setting. Third, we provide a proof-of-concept implementation of FORZA by means of an implementation of D3 and OntoPartS-2 that can be reused across ontology development environments. The FORZA method with D3 and OntoPartS-2 has been integrated into the MoKi ontology development environment [1], so that it is now available for use. The paper is structured accordingly.

2. AN EXPLANATORY USE CASE

The "tourism ontology" use case introduced in the following example is used throughout the paper to discuss and illustrate the ideas and the FORZA method we propose.

Example 1. *A tour operating agency aims at providing the best travel plan and recommendations to their customers, based on their interest and preferences, such as length of travel, mode of transportation, accommodation type, and activity theme. In order to do so, the agency has to retrieve a number of accurate information about touristic destinations and best possible routes available. The agency is proposed with an approach that makes use of semantic technologies (and in particular ontologies) to solve information integration problems that arise from the collection of the required information from different sources.*

Aided by a knowledge engineer, they start the development of the domain ontology by identifying five tourism sub domains of interest: Region*,* Event*,* Accommodation*,* Attractions*, and* Transportation*, and by building an initial taxonomy covering all these sub domains. After this initial stage, the tour operating agency decides to carry on building the domain ontology in-house without the support of the knowledge engineer in order to reduce costs.*

Along with refining the taxonomy, one of the key tasks that needs to be performed is the definition of relationships (object property expressions) between the entities present in the ontology. Since defining well founded relationships is a complex ontology engineering task, the experts of the tour operating agency decide to use the GENERATOR method (and its FORZA realisation) to facilitate them in selecting and reusing relations from general pre-existing ontologies, thus reducing the intervention of knowledge engineers, speeding up the ontology development process, and inserting high-quality knowledge in their ontology.

3. THE GENERATOR METHOD

In this section, we describe the general method for logic-based and ontology-driven ontology authoring that we propose. The method is independent from whether its implementation is (semi-)automated and independent of the chosen ontology language. That is, it equally well can be realised manually with one or more ontologies formalised in a higher-order logic or semi-automated with OWL ontologies and OWL automated reasoners. For the sake of generality we envisage two scenarios, graphically illustrated in Figure 1: a scenario "A" in which the experts perform the authoring of their domain ontology with the help of one or more foundational ontologies, and a scenario "B" in which the experts perform the authoring of their domain ontology without the support of foundational ontologies. Thus, scenario A is envisaged especially for the enhancement of small ontologies or mere taxonomies, and for all the cases in which the users aim at maximise the reuse of already existing knowledge, while scenario B provides the freedom of using the method without committing to any pre-existing knowledge external to the domain ontology. We describe our method in two stages: the required materials and the method.

3.1 Materials

Regardless of the scenario, the application of the method requires the following materials, with the key aspects in italics font, which have to be set up only once (that is, not for

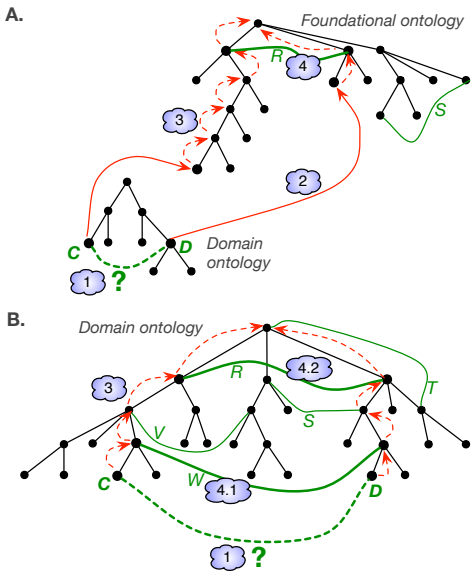

Figure 1: Depiction of the general idea of GENERATOR. Scenario A: a domain ontology and a foundational ontology; Scenario B: procedure when using only one ontology (see text for explanation).

each axiom one contemplates to add to the ontology to be enhanced):

- A *(top-)domain* or *reference ontology* O_d to be extended;
- In the case of scenario "A", one (or more) *foundational or high-level relational ontologies*, $O_{f1}, O_{f2}, \ldots, O_{fn}$.
- The ontology or ontologies (O_d and O_{fi}) should have one or more relationships (object properties) whose *domain and/or range have been declared* (and therewith have a language with which one declare this);
- Optionally, an ontology development environment for ontology authoring;
- Optionally, if partial automation is desired: an *automated reasoner* with the minimum capability of traversing the taxonomy upwards;

In order to tackle scenario "A", the domain ontology O_d should be aligned with the foundational ontologies O_{fi}. In the case where this alignment is not present, the method provides a capability to do this during its application either by manually selecting the category or through a *decision diagram* to guide the modeller to select the appropriate category in the foundational ontology O_{fi} for the selected domain and/or range classes of the domain ontology O_d. This alignment is either permanent or may be used just during the axiom selection process.

For our running example, we make the following choices.

Example 2. *In order to use the GENERATOR method to extend the tourism ontology (our O_d) presented in Example 1 we have to choose whether we want to make use of a foundational ontology or not. Since the ontology only consists of a taxonomy, it is more appropriate to do so. For the sake of the example we also assume that O_d is still to be linked permanently to a foundational ontology. Among the available foundational ontologies, we select DOLCE as O_{f1} (the selection and an explanation can be obtained from the ONSET recommender tool [9]). Moreover, upon inspection, the tourism ontology is lacking in part-whole and mereotopological (parthood and location) relations, and therefore we decide to include the mereoTopoD ontology (see [8]) as additional foundational ontology O_{f2}. The tourism ontology is represented in OWL, hence, we can use some Semantic Web tools, such as Protégé or MoKi, as ontology development environment.*

3.2 The method

Let us consider the most complex case of scenario A of Figure 1, where a modeller has a domain ontology and a foundational ontology. She first selects the domain and range classes she wants to relate somehow (step (1) in Figure 1-A). Then (step 2), she has to check the alignment of the selected classes to the relevant categories in the foundational ontology. Here there are several cases: (i) the alignment already exists, so this step is completed automatically in the background, (ii) the alignment is carried out manually if the modeller knows the foundational ontology well, or (iii) the modeller chooses to be guided by the decision diagram that is specific to the selected foundational ontology. If the alignment does not exist yet, the modeller can choose to save it permanently or just use it for the duration of the application of the method. Step (3) consists in moving up in the taxonomy to (automatically) find the possible object properties. While step (3) can be done manually, or else with a quick script to retrieve a class's parent class in the machine-interpretable version of the ontology, there is a distinct advantage to using a reasoner besides saving oneself analysis and coding time: taxonomic classification, which offers the most up-to-date class hierarchy—including implicit subsumptions in the class hierarchy—and therewith avoiding spurious candidates. From a modeller's viewpoint and assuming automation, then one only has to select which classes to relate, and, optionally, align the ontology, and the software can handle the rest, as each time it finds a domain and range axiom of a relationship in which the parents of C and D participate, it is marked as a candidate property to be used in the class expression. Finally, the candidate properties are returned to the user (step 4). Not shown in the diagram but equally possible, is the use of a separate relation ontology, as in our Example 2, which typically is a variant of scenario A: the modeller uses a relation ontology to axiomatise the base ontology by adding relations to it. In this scenario, the base ontology is also traversed upwards and on each iteration, the base ontology class is matched against relational ontology to find relations where the (parent of the) class is defined in a domain and range axiom, also until the top is reached before returning candidate relations. Scenario B is similar to A regarding traversing the domain ontology upwards to find the candidate relations, but without the step of alignment to the foundational ontology.

Given the labels in Figure 1, the computed suggestion to relate C and D for scenario A is object property R. For scenario B, it suggests both W and its super property R. It is then up to the modeller to select among the available choices and save the selection into the ontology. Algorithmically, the process is summarised in Figure 2; the actual algorithm comprises two pages, which will be illustrated more in depth in Section 4.

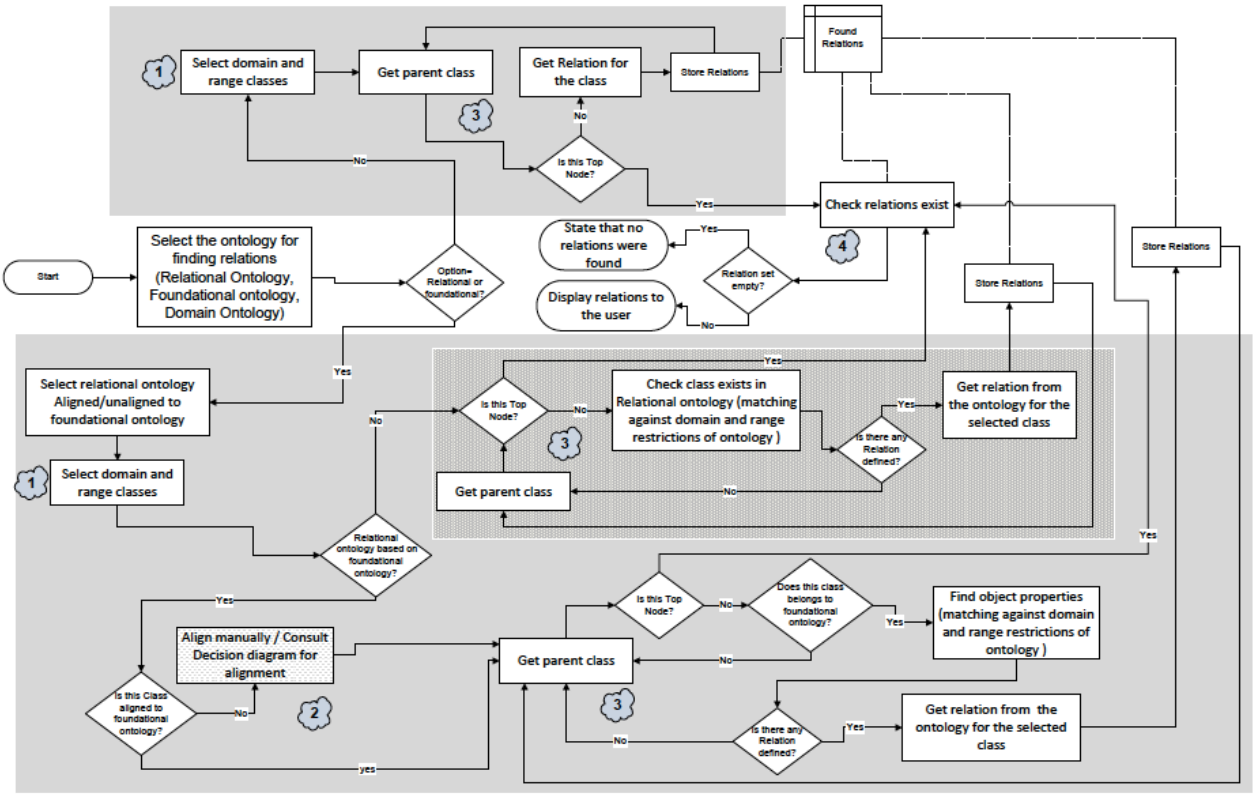

Figure 2: Summary of the selection algorithm, represented as a flow chart.

Thus, GENERATOR exploits the use of already declared knowledge in an ontology—be that its own or a borrowed one—and the use of manual or automated reasoning to compute the rest, thereby narrowing down the possible set of relations that can exist between two classes. Moreover, neither manual searching and assessment is strictly necessary anymore, nor will the selection result in an inconsistency based on that relation between the two selected classes alone[1]. Hence, it can greatly reduce the cognitive overload during ontology authoring of, especially, large ontologies, it fosters reuse of already well-researched knowledge, especially in scenario A, and by using a foundational ontology, it also anticipates easy mappings with other ontologies.

4. THE FORZA REALIZATION

Example 3. *Continuing with our use case, DOLCE and mereoTopoD are already aligned with each other. Therefore, we only have to link them and the tourism ontology. Since the experts of the tourist information office are not familiar with DOLCE, they choose to link their domain ontology by means of the decision diagram support provided by the method. The (subsumption axioms describing the) linking with DOLCE along with the taxonomy of part-whole and mereotopological relations are then saved in the OWL file.*

As illustrated in Example 3, in order to continue with their task of selecting and reusing relations from general pre-existing ontologies, the experts of the tourist information office now require a concrete instantiation of scenario A of the GENERATOR method described in Section 3 to a concrete foundational ontology and to a source of domain-independent part-whole relations; in fact, it requires the most comprehensive realisation. In this section we provide a description of such an instantiation where we aim at guiding ontology authors in linking their domain ontology with the DOLCE foundational ontology [10] and in the selection of part-whole relations taken from the relation ontology of part-whole relations (compatible with DOLCE) mereoTopoD [7, 8][2]. The specific instantiation of GENERATOR described here is called FORZA. It provides: (i) a decision tree to facilitate linking classes in the domain ontology to DOLCE categories, implemented as a tool called D3, and (ii) computation of the applicable part-whole relation(s) using a reasoner, which is implemented in the OntoPartS-2 tool that extends OntoPartS [8]. Although we illustrate it with a domain ontology in the tourism sector and with part-whole relations, the Q/A system illustrated in Section 4.1, and the selection algorithm illustrated in Section 4.2 and Figure 2 are general enough to cover a wide range of domains and relation types. Recall that while FORZA is tailored to DOLCE and mereoTopoD, it certainly does not depend on the subject domain of the ontology at hand, like the tourist domain in our running example. It can therefore be used to foster the reuse of domain independent elements and relations in any concrete setting.

[1] it does not, however, prevent other possible inconsistencies; e.g., conflicting cardinalities is a typical pitfall.

[2] `http://www.meteck.org/files/ontopartssup/MereoTopoD.owl`

4.1 Using D3 to link the ontology to DOLCE

Here we illustrate step (2) in Figure 1 in the scope of the FORZA method, where classes of a domain ontology are linked with appropriate DOLCE categories. We have chosen DOLCE as it is a foundational ontology describing very general entities by means of a rich and precise axiomatisation and it can serve to support the construction of domain specific ontologies [15, 4]. In our example, moreover, DOLCE has been chosen as it fits well with the mereoTopoD ontology of part-whole relations [7, 8]. In other words, the selection of the appropriate DOLCE category triggers the selection of appropriate part-whole relations by using an extension of OntoPartS [8] (described in Section 4.2).

The selection of appropriate categories from a foundational ontology is known to be a difficult task. The problem can become even bigger when the ontology authors are domain experts or novices, who do not have the necessary background in formal ontologies to fully understand DOLCE. We solve this bottleneck by introducing the feature of a decision diagram, or 'Question and Answer (Q/A)' approach. The Q/A decision system, called *DOLCE Decision Diagram* (D3), asks the user one or more closed questions and depending on the answers given by the user, the system will propose the category. The one we propose for the DOLCE ontology is based on the fundamental ontological distinctions embedded in DOLCE itself and is shown in Figure 3. Thus, if we have to link a class "X" to DOLCE we start traversing the DOLCE taxonomy asking the ontology author whether "X" is wholly present, perceived, does not exist in space nor time, or is happening/occurring, which capture the fundamental ontological distinction between Endurant, Quality, Abstract and Perdurant. More precisely, we exploit DOLCE's disjointness axioms between subclasses, the descriptions of the categories augmented with the more informal WordNet descriptions, and the examples described in DOLCE's documentation. We illustrate the usage of D3 with our explanatory use case in the following example[3]

Example 4. *Let us assume that our domain experts have to link the concepts of their domain ontology, and in particular the concept* Carnival *(used to describe events such as the famous Venice Carnival), to a DOLCE category with the help of* D3. *An example of the interaction between the user and the system may be as follows:*

D3: *Is* Carnival *something that is a happening or an occurring?*

User: *Yes*

D3: *Are you able to be present or participate in a* Carnival*?*

User: *Yes*

D3: *Is* Carnival *Atomic, i.e., it has no subdivisions and has a definite end point?*

User: *No*

Here D3 *uses the answers to traverse the DOLCE taxonomy presented in Figure 3 and terminates by suggesting to link* Carnival *to the DOLCE category* Accomplishment. *Likewise, the experts of the tour operating agency can use* D3 *to link* CarnivalPartyGoer *as follows:*

D3: *Is* CarnivalPartyGoer *wholly present at any time of its existence?*

User: *Yes*

D3: *Is* CarnivalPartyGoer *a collection of things, regardless of their spatial other differences?*

User: *No*

D3: *Does* CarnivalPartyGoer *not take up space or does* CarnivalPartyGoer *need some other object to exist in, or is* CarnivalPartyGoer *a role played by some object? (e.g., a student)*

User: *Yes*

D3: *Is* CarnivalPartyGoer *dependent on a community of (>1) agents or embedded in some social setting?*

User: *Yes*

Following the answers in the tree in Figure 3 CarnivalPartyGoer *will be linked to the DOLCE category* Social Object.

The idea of a Q/A system for the selection of DOLCE categories is not new. Similar work has been done in the TMEO Methodology using a particular version of DOLCE called DOLCE-spray [12]. TMEO (Tutoring Methodology for the Enrichment of Ontologies) is a semi-automatic interactive Q/A system for guiding humans in the population of Italian semantic resources. The aim of TMEO is to select the most appropriate category from a reference ontology as a superclass for a given lexicalised concept. Compared to our work, TMEO is in Italian and it mainly focuses on DOLCE-spray, which is a specific, not freely downloadable, version of DOLCE that is specialised for Italian semantic resources. DOLCE-spray covers only 6 DOLCE categories, compared to the 24 core categories of our decision tree, which hampers its reuse, and it contrasts with the relatively widely used DOLCE-lite OWL file. Notably, D3 covers the entire core DOLCE taxonomy except the 3 sub categories of Region (Abstract, Physical, and Temporal Region), and the part-whole relations taxonomy uses, among others, DOLCE's Perdurant and Social Object that are not present in DOLCE-spray but are in DOLCE-lite. Note that, in addition to linking already existing domain concepts, D3 can also be used to create new domain concepts and immediately link them to DOLCE.

D3 is implemented on top of a tree (shown in Figure 3) where nodes can have binary or multiple branches. For binary selection points, the user will be asked one Yes/No question. In the case of more than two branches, the system will present a multiple choice question to the user (one item for each branch) so that she can select the best possible option. While having multiple questions sometimes can be necessary, single Yes/No questions are simpler to handle for users. For this reason, we have made an attempt to maximize them by adding, where appropriate, temporary nodes; an example is the node "Temp Entity" in Figure 3, compared to the original DOLCE taxonomy that has Arbitrary Sum, Physical Endurant and Non-Physical Endurant all directly subsumed by Endurant.

4.2 Reasoner-enhanced selection of part-whole relations

In this section we describe an instantiation of steps (3) and (4) of Figures 1 and 2, where candidate relations are

[3]For the example, we assume that the taxonomy produced by the domain experts is the simplified version of the travel ontology at `protege.cim3.net/file/pub/ontologies/travel/travel.owl`, which is available at `sourceforge.net/projects/cikmontology/files/tour.owl/download`

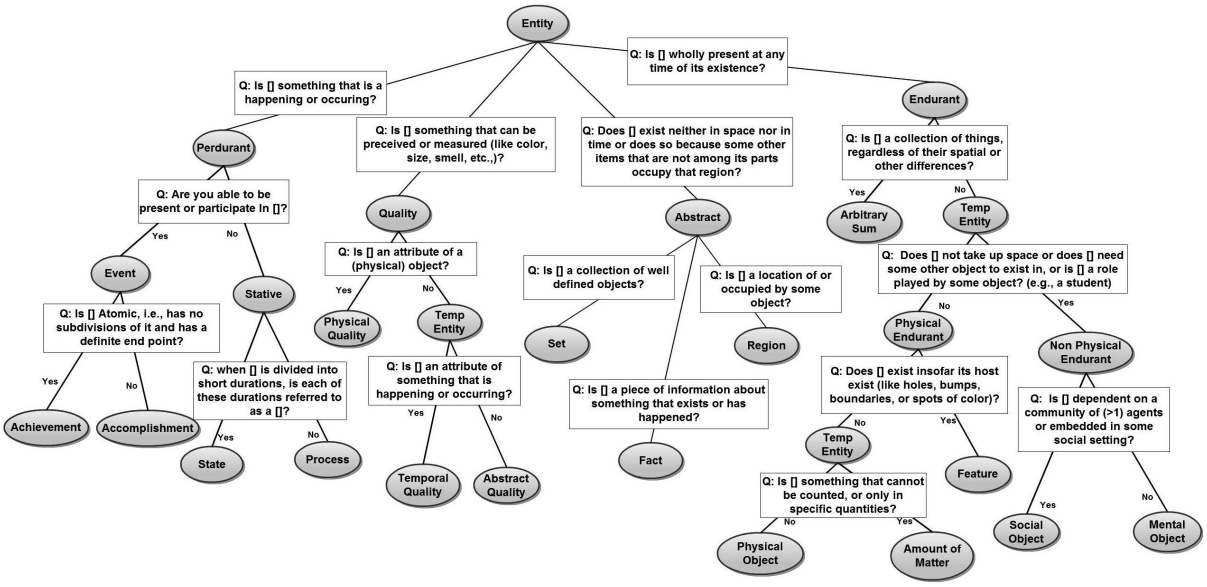

Figure 3: The Decision Tree of D3: the Perdurant branch is implemented as binary tree and the rest having more than two branches; a single branch can be selected at a time.

returned to the ontology authors. Because we chose the most comprehensive scenario with an additional relation ontology of part-whole relations, we first describe the taxonomy of part-whole relations we use, and then the reasoner-enhanced algorithm used to propose relations.

4.2.1 *Part-whole relations*

Part-whole relations have been investigated in conceptual data modelling, ontologies, and Ontology, and many relations have been been proposed as a kind of part-whole relation. We use a taxonomy of part-whole relations that is based on an ontological investigation and adapted to the Semantic Web setting as an OWLized object property hierarchy aligned with DOLCE (i.e., mereoTopoD). This taxonomy is based on [7, 8] and contains 23 part-whole relations. Summarising the taxonomy, a first main distinction is made between merely meronymic part-whole relations, such as member-of and constituted-of, and mereological parthood with its basic refinements to cater for different domain and ranges; e.g., a structural parthood for objects versus an involved-in to represent processes and part-processes. The latter main branch is refined further with mereotopological relations, such as non-tangential-proper-part-of and non-tangential-proper-located-in, which are required for certain subject domain; e.g., to derive that country Lesotho is landlocked and wholly enclosed in another country (South Africa), for image annotation and analyses, and anatomy. All relations in this extended taxonomy of part-whole relations have domain and range restrictions using DOLCE categories, which facilitated greatly the design and development of the OntoPartS tool [8] and this feature is exploited further in OntoPartS-2. For the current purpose, also inverses of the part-whole relations have been declared (except for the mereotopological equal relations) which may increase its usage. Informal, yet to be investigated, feedback and preliminary domain ontology analysis revealed that modellers seem to prefer explicit inverses when the 'direction' of the relation—i.e., the so-called "all-some" pattern—goes from whole to part. For instance, in OWL 2 (in DL notation), the class expressions $\mathsf{Computer} \sqsubseteq \exists \mathsf{hasProperPart}.\mathsf{CPU}$ and $\mathsf{Computer} \sqsubseteq \exists \mathsf{properPartOf}^{-}.\mathsf{CPU}$ are the same with respect to the subject domain semantics and correct with respect to reality (and it is not the case that each CPU in existence is a proper part of a computer), but the former way of axiomatising the knowledge appears more often. This brings the total amount of part-whole relations to 44.

4.2.2 *Selecting part-whole relations with* OntoPartS-2

Here we describe the workings of OntoPartS-2 using the bottom half of Figure 2. Once the domain and range classes from the domain ontology that play the part and whole have been selected in step (1), there are three options: (i) manually align them to DOLCE, (ii) use D3 to align them, and (iii) the domain ontology is already aligned; OntoPartS-2 caters for each of them. The alignment, if not already present, is stored in the ontology. This brings us to the reasoner-enhanced component, where we traverse upwards in the class taxonomy (step 3). A call is issued to the reasoner to get the parent class of the classes selected in step (1) and it checks whether that class is in DOLCE, which is repeated upward in the class hierarchy until the DOLCE class is found. Once a DOLCE OWL class is reached, it makes a call to the function to check whether that class is a domain (respectively, range) in one of the domain (resp. range) axioms from the taxonomy of part-whole relations; e.g., involved-in has as domain and range DOLCE's Perdurant. As a minor optimization step, the range class—i.e., a parent of D in Figure 1—is checked only if the domain matches. If there is a match, i.e., the domain and range of the part-whole relation are parents of both the domain ontology's part and whole classes that were selected in step (1), then the matched relation is retrieved and stored in a temporary in-memory storage. This holds for automated 'discovery' of both the relation and its declared inverse, where

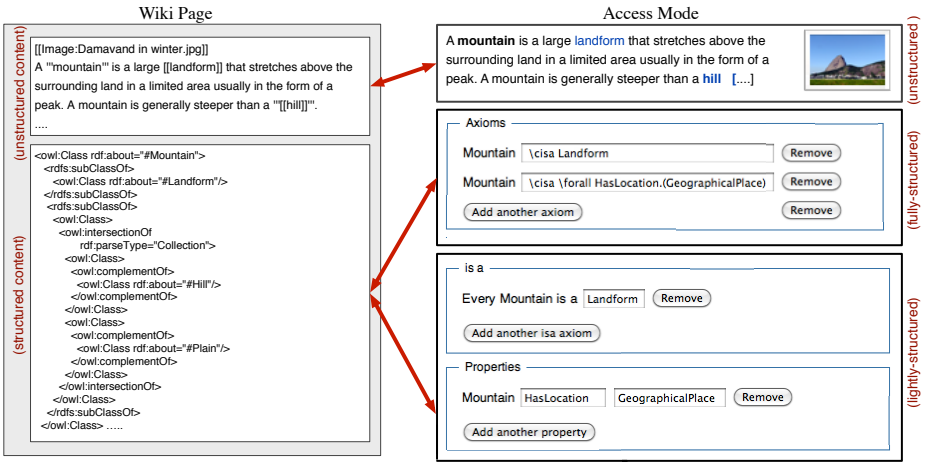

Figure 4: The MoKi page for the OWL class Mountain.

the inverses are retrieved for the relations stored in memory as an effort to reduce computational time. This process of traversing upwards, matching relations, and storing them continues until the top is reached. Thus, it automatically reduces the possible set of part-whole relations for the selected domain ontology classes to only the (onto-)logically viable ones, without the modeller having to conduct manual cross-checking between each object property, its domain and range, and the position of the classes in the hierarchy. In the final step (step 4), the memory storage is checked to see if there are relations stored there; if so, all found relations are returned to the user, if not, then the user will be notified that no suitable relations were found for the selected (DOLCE categories aligned with the) part and whole. The following example illustrates the algorithm.

Example 5. *Let us continue with our example and assume that the experts of the tour operating agency want to relate* CarnivalPartyGoer *to* Carnival. *The following happens, given our current scenario and part-whole relations:*

1. *The modeller chooses* CarnivalPartyGoer *to play the role of part and* Carnival *to play the role of the whole.*
2. *OntoPartS-2 gets the parent class of* CarnivalPartyGoer*, and its parent class, and so on, until it reaches* SocialObject*, which is the DOLCE category it was aligned with (recall Example 4).*
3. *It checks whether* SocialObject *is declared as a domain of an object property. It is a domain of* member-of.
4. *It iteratively moves up further in the hierarchy, and finds as parent class* Endurant*, which is the domain of* participates-in*, and* s-part-of.
5. *A similar process occurs with* Carnival. *Moving up automatically in the class hierarchy, the first DOLCE class it encounters is* Accomplishment *(recall Example 4).*
6. *It checks whether* Accomplishment *is a range for* member-of*,* participates-in *and* s-part-of. *It is not.*
7. *It continues upward in the DOLCE hierarchy and checks whether that class is a range for* member-of*,* participates-in*, and* s-part-of. *This holds for* Perdurant*, which is a range for* participates-in.
8. *The top-class is* Particular*, which does not match either candidate relation.*
9. *Finally, it returns the only viable relation:* participates-in.

This can then be saved into the ontology; that is, the axiom CarnivalPartyGoer $\sqsubseteq \exists$participates-in.Carnival *can be added. In case the modeller is also interested in the inverses, OntoPartS-2 will get the inverse relation for the matched object property* participates-in *instead of computing it from scratch.*

The only other tool that provides guidance in selecting part-whole relations is OntoPartS [8]. OntoPartS-2's improvements over OntoPartS are manifold. It uses the OWLized version of DOLCE and a reasoner to traverse the hierarchy upwards to narrow down the possible set of part-whole relations, compared to having a hard-coded DOLCE taxonomy and hand-crafted rules. It incorporates the D3 decision diagram for selecting DOLCE categories instead of providing examples when hovering over the DOLCE category when a category had to be selected. In addition, one can save the alignment of the domain class to the foundational ontology or one did not need to select it in the first place, whereas in OntoPartS, this process has to be repeated each time regardless whether the domain ontology was already aligned or not. Finally, OntoPartS-2 offers users the freedom to select the 'direction' of the chosen part-whole relation, opting for an axiom from the part to the whole or its inverse or both, versus the necessary conceptual leap of an `ObjectInverseOf` axiom. The first improvement makes the process more reliable and maintainable, the second resolves a noted bottleneck of OntoPartS [8], the third one reuses already provided knowledge in the process, saving the modeller any repetition, and the fourth one meets a cognitive and OWL 2 feature issue regarding the explicit inverses.

5. IMPLEMENTING FORZA

Here we describe the current implementation of the FORZA method. First, we briefly describe the implementation of D3 and OntoPartS-2 as stand-alone tools, and then how we have integrated them in the collaborative ontology authoring tool MoKi [1].

5.1 D3 and OntoPartS-2

The D3 system is designed as an interactive decision tree, implemented with HTML, CSS and JavaScript. As this system is realised as a client-side application, it will allow its integration with any web based application. The set of questions for guiding the users are stored in a XML file. This approach provides the flexibility of updating or modifying the set of questions independently of the front-end application.

The algorithm summarised in Figure 2 has been implemented by using Java at the back-end and the OWL API (v3.3) to interact with the ontologies. The back-end application contains the main logic for the tool and all the required libraries. It has been aggregated in a jar file stand-alone application that can be called by any external interface. Currently, OntoPartS-2 is customised to the FORZA scenario[4] but it is easily customisable for all the combinations illustrated in Figure 1.

5.2 Integrating FORZA in MoKi

MoKi [1] is a collaborative MediaWiki-based [11] tool where heterogeneous teams of knowledge engineers and domain experts, with different knowledge engineering skills, can actively collaborate to the modelling of ontological knowledge. MoKi is grounded on three main pillars, which we briefly illustrate with the help of Figure 4:

1. Similarly to most state of the art wiki-based tools, MoKi associates a wiki page to each each basic entity of the ontology (i.e., classes, object and datatype properties, and individuals);
2. Each wiki page describes the entity it is associated with by means of both unstructured (e.g., free text, images) and structured (i.e., OWL axioms) content.
3. A multi-mode access is provided to support both domain experts and knowledge engineers in the authoring process. In particular, MoKi provides two modes to access the structured formal content: a *fully-structured* access mode, where knowledge engineers can edit the ontology content by means of OWL axioms, and a *lightly-structured access mode*, where domain experts, can edit the ontology content by means of a simplified, form-based, view on the (same) structured formal description.

As argued extensively in [1], one of the advantages of MoKi, and of MediaWiki-based tools in general, is the ability of easily customising their form-based user interface to support the involvement of domain experts. This has made MoKi, and in particular the *lightly-structured access mode* form-based interface for domain experts, the natural candidate for the integration of FORZA in an ontology authoring tool.

In a nutshell, the main idea of the integration of FORZA in MoKi is to load both the domain and foundational ontologies in MoKi, and to extend the form-based *lightly-structured access mode* interface to accommodate for the user interface of the Q/A system D3 and for the suggestions of OntoPartS-2. Now we will illustrate the tool by means of a possible walk-through,Êwhich is described in Example 6 and Figures 5 and 7.

[4]Jar files are available at: `sourceforge.net/projects/cikmontology/files/CIKM2013.zip/download`

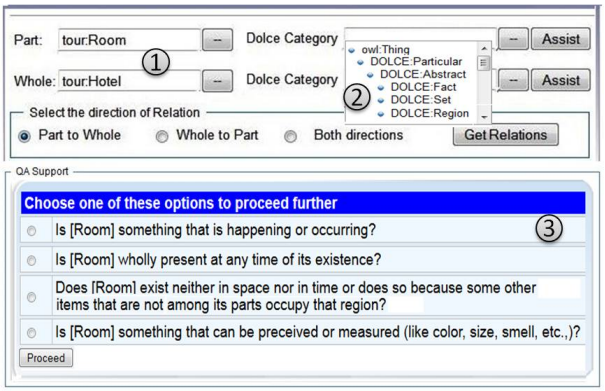

Figure 5: Linking with DOLCE in MoKi.

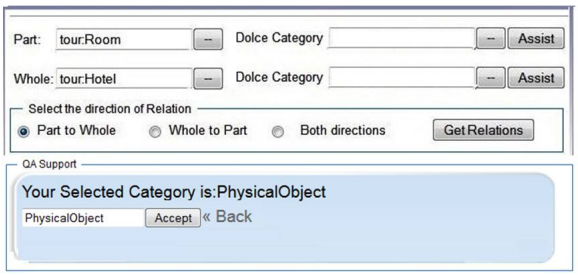

Figure 6: Aligning Room from the domain ontology to the DOLCE category PhysicalObject with D3.

Example 6. *Suppose that after selecting the relation between* Carnival *and* CarnivalPartyGoer *of Example 5, the domain expert has to figure out what relation exist between the classes* Room *and* Hotel *of her domain ontology. She loads the initial taxonomy into MoKi and selects the* Room *and* Hotel *classes that are to be related by means of a part-whole relation in a dedicated interface (see (1) in Figure 5). If these entities are already aligned with DOLCE, then MoKi will automatically show them in the "Dolce Category" text box, otherwise the user will have to select them either manually using a drop down menu (see (2) in Figure 5) or by clicking the "Assist" button next to the "Dolce Category" text box. This will make the Q/A support appear (see (3) in Figure 5). Once the user has submitted her initial choice, the system proceeds with the other questions as follows (we omit the MoKi screenshots for lack of space)*

D3: *Is* Room *a collection of things, regardless of their spatial or other differences?*

User: *No*

D3: *Does* Room *not take up space or does Room need some other object to exist in, or is Room a role played by some object? (e.g., a student)*

User: *No*

D3: *Does* Room *exist insofar its host exist (like holes, bumps, boundaries, or spots of color)?*

User: *No*

D3: *Is* Room *something that cannot be counted, or only in specific quantities?*

User: *No*

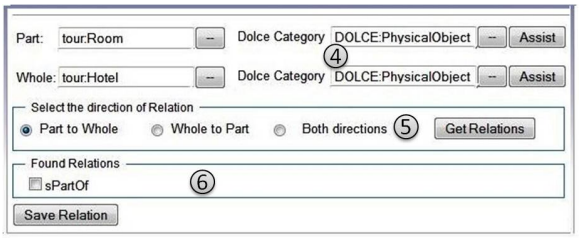

Figure 7: Selecting relations in MoKi.

D3 will select the DOLCE category PhysicalObject *for* Room*, and will present the result to the user for confirmation, as shown in Figure 6. A "Back" button is present at any time of the interaction to go back to the previous decision point. A similar interaction is also performed for the* Hotel *class, which is also linked to the DOLCE category* PhysicalObject*. At this point* PhysicalObject *is used to fill in the "Dolce Category" textbox in the MoKi interface (see (4) in Figure 7), and the axioms* Room $\sqsubseteq$ PhysicalObject *and* Hotel $\sqsubseteq$ PhysicalObject *are saved in the domain ontology along with the imported DOLCE. Once aligned, the desired concepts with DOLCE, the Q/A support closes and the domain expert can use the interface to specify the direction of the relation that OntoPartS-2 should retrieve by selecting one of the radio buttons in Figure 7 (see (5)). Then she will press the "Get Relations" button to retrieve the set of possible relations, which will be shown in the Found Relation section of the interface (see (6) in Figure 7). In the background, OntoPartS-2 checks whether* PhysicalObject *is declared as a domain of an object property and it finds that it is defined as domain for* member-of *and* constituted-of*. Now, it iteratively moves up further in the hierarchy until the top class is reached, and finds as parent class* Endurant*, which is the domain of* participates-in *and* s-part-of*. After matching the domain and range of these retrieved properties (*member-of*,* constituted-of*,* participates-in*, and* s-part-of*) with the parent classes of the* Room *and the* Hotel*, it returns the only viable relation:* s-part-of*. Now the user can select the desired relation from the list by clicking the check box next to it and add this relation to the ontology by pressing the "Save Relation" button. This will add the axiom* Room $\sqsubseteq \exists$sPartOf.Hotel *to the Tourism ontology along with importing the part-whole relations taxonomy.*

6. DISCUSSION

In Sections 3–5 we presented a rather elaborate method to support the linking and reuse of existing knowledge in the process of ontology authoring, and a practical implementation. Here we discuss first how the method addresses the issues emphasised in Section 1, and subsequently considerations about the design and realisation of the method in practical implementations.

6.1 Ontology-driven ontology authoring

GENERATOR itself amounts to *ontology-driven ontology authoring* in two senses: 1) by using higher-level, relatively well-researched notions from Ontology with foundational ontologies or reuse of agreed-upon domain ontology knowledge to add entities and axioms to an ontology, and 2) by exploiting the logic-based representation and therewith automated reasoning during the stage of adding knowledge to the ontology. The first component provides a top-down approach for answering the "Where to start? Where to add classes and how to relate them? What can be reused?", which is demonstrated concretely with FORZA and its D3 and OntoPartS-2. By availing of the foundational ontology, one increases the ontology's quality, prospect for reuse, and interoperability with other DOLCE-aligned ontologies. Moreover, D3 saves the time of delving into the documentation for the typical unambiguous cases, and its implementation as D3 can easily be reused in other ontology development efforts. The reasoner-enhanced part-whole selection with OntoPartS-2 saves the user from selecting the category of the class that plays the part and that plays the whole, as was the case for OntoPartS, and have this sorted out either by D3 or directly through traversing the hierarchy upward to find all candidate properties. Thus, we are now one step 'up' from the post-hoc trial-and-error usage of automated reasoners. To the best of our knowledge, it is the first time a reasoner is used in ontology authoring in this way. The time to retrieve relations in the tourism and DMOP[5] ontologies is 406 ms and 1408 ms, respectively, and the whole operation including retrieving the DOLCE category for already aligned classes is 1197 and 4267 ms, respectively, where in both cases the OWLReasoner of OWLAPI was used. The difference in time observed for these ontologies is proportional to the depth of the branch and the number of properties defined in the ontology.

6.2 Design considerations

The material necessary to instantiate GENERATOR for a specific method, like FORZA, have to be set up only once and thereafter they can be reused for the development of any domain ontology. Setting up such an instantiation, however, is not trivial and it presupposes that at least some domain and range axioms have been declared. The latter is certainly the case for foundational ontologies, but some might find it a prohibitive requirement for the single-ontology scenario, despite that it makes for a more accurate and precise ontology with respect what it is supposed to represent. We stress, however, that the purpose here is axiomatisation, not preventing novices from encountering unexpected deductions. Concerning FORZA, we chose to add explicitly the inverses, instead of using OWL's `ObjectInverseOf`, which is motivated by perceived user preference (a full investigation on this issue is planned for future work). FORZA with OntoPartS-2 focuses on adding class expressions with part-whole relations from a separate relation ontology, but one equally well can extend OntoPartS-2 for any DOLCE object property (there is a considerable overlap of properties in DOLCE and mereoTopoD already anyway). Other possible extensions entail broadening the scenarios; e.g., guidance on adding new properties, new OntoPartS-2 functionality for ABox assertions, or adding an optional foundational ontology selection step to GENERATOR.

Another consideration refers to the ability to produce an adequate decision diagrams (Q/A system) for the foundational ontology at hand. While the one presented in the FORZA instantiation is general enough and reusable in several cases, it does not aim at providing a base for a universal Q/A system. From the experience of building D3, we can

[5] `http://www.dmo-foundry.org/download-DMOP`

describe several considerations that should be taken into account in oder to successfully design a new decision diagram:

- Consider the description of the entities provided in the ontology and its documentation and, if required, modify it to make it easily understandable for the users by using 'layperson terms';
- Consult a lexical resource for ideas for informal definitions of the entities if they are not already defined or described in the ontology;
- Exploit disjointness axioms among classes in ontology: they provide clear distinctions between entities that facilitate designing closed questions;
- If more than one criterion applies, then combine them together or select the best one that applies to that category;
- Avoid using the term presented in one branch for defining question for a category in another branch;
- When preferring yes/no answers over radio buttons, then when there are more than 2 subclasses to choose from, introduce a temporary node in the decision diagram.

7. CONCLUSIONS

A novel general method for the reuse of ontological knowledge for ontology authoring, GENERATOR, was introduced, and we have provided a concrete instantiation of it for the reuse of knowledge from the DOLCE foundational ontology and a general taxonomy of part-whole relations. This instantiation, called FORZA, can be used to build ontologies in any specific domain. FORZA includes novel, automated support for (i) the linking of a domain ontology with DOLCE (the D3 tool), and (ii) an automated reasoner-based computation of part-whole relation(s) (the OntoPartS-2 tool). The independently usable D3 and OntoPartS-2 have been integrated in the MoKi ontology authoring tool, therewith demonstrating a working proof-of-concept of FORZA.

To the best of our knowledge, there are no works in the literature that explicitly aim at defining comprehensive methodological and tool-based support for ontology developers in deciding how (where) to reuse any of the knowledge already present in (foundational) ontologies (the few works with the same aim [5, 12, 8] have been described). By providing such methodological and tool-based support, this paper provides a first significant contribution towards effectively supporting ontology authoring driven by ontology reuse.

8. REFERENCES

[1] C. Di Francescomarino, C. Ghidini, and M. Rospocher. Evaluating wiki-enhanced ontology authoring. In *Proc. of EKAW'12*, volume 7603 of *LNAI*, pages 292–301. Springer, 2012. Oct 8-12, Galway, Ireland.

[2] V. Dimitrova, R. Denaux, G. Hart, C. Dolbear, I. Holt, and A. G. Cohn. Involving domain experts in authoring OWL ontologies. In *Proc. of ISWC'08*, volume 5318 of *LNCS*, pages 1–16. Berlin, 2008.

[3] M. Fernandez, A. Gomez-Perez, A. Pazos, and J. Pazos. Building a chemical ontology using METHONTOLOGY and the ontology design environment. *IEEE Expert: Special Issue on Uses of Ontologies*, January/February:37–46, 1999.

[4] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, R. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In *Proc. of EKAW'02*, pages 166–181. Springer, 2002.

[5] G. Kassel. Integration of the DOLCE top-level ontology into the OntoSpec methodology. Technical report, LaRIA, October 2005.

[6] C. M. Keet. The use of foundational ontologies in ontology development: an empirical assessment. In G. Antoniou et al., editors, *Proc. of ESWC'11*, volume 6643 of *LNCS*, pages 321–335. Springer, 2011.

[7] C. M. Keet and A. Artale. Representing and reasoning over a taxonomy of part-whole relations. *Applied Ontology*, 3(1-2):91–110, 2008.

[8] C. M. Keet, F. C. Fernández-Reyes, and A. Morales-González. Representing mereotopological relations in OWL ontologies with ONTOPARTS. In E. Simperl et al., editors, *Proc. of ESWC'12*, volume 7295 of *LNCS*, pages 240–254. Springer, 2012.

[9] Z. Khan and C. M. Keet. ONSET: Automated foundational ontology selection and explanation. In *EKAW*, volume 7603 of *LNAI*, pages 237–251. Springer, 2012.

[10] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. WonderWeb deliverable D18 ontology library (final). Technical report, LOA-CNR, 2003.

[11] Wikimedia Foundation. Mediawiki. http://www.mediawiki.org, Accessed on 6 Nov 2011.

[12] A. Oltramari. A tutoring methodology for the enrichment of ontologies. In *Cahiers de Lexicologie, Classique Garnier*, pages 221–229, 2011.

[13] I. Opalicki and S. Lovrencic. How well are domain and upper ontologies connected? In *CECIIS*, pages 17–22, 2012. September 19-21, 2012, Varazdin, Croatia.

[14] V. Presutti, A. Gangemi, S. David, G. A. de Cea, M. C. Surez-Figueroa, E. Montiel-Ponsoda, and M. Poveda. A library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. NeOn deliverable D2.5.1, NeOn Project, Institute of Cognitive Sciences and Technologies (CNR), 2008.

[15] S. K. Semy, M. K. Pulvermacher, L. J. Obrst, and M. K. Pulvermacher. Toward the use of an upper ontology for u.s. government and u.s. military domains: An evaluation. Technical report, IIWeb-04, in conjunction with VLDB-2004, 2004.

[16] E. Simperl, M. Mochol, and T. Bürger. Achieving maturity: the state of practice in ontology engineering in 2009. *International Journal of Computer Science and Applications*, 7(1):45–65, 2010.

[17] M. C. Suarez-Figueroa, G. A. de Cea, C. Buil, K. Dellschaft, M. Fernandez-Lopez, A. Garcia, A. Gomez-Perez, G. Herrero, E. Montiel-Ponsoda, M. Sabou, B. Villazon-Terrazas, and Z. Yufei. NeOn methodology for building contextualized ontology networks. NeOn Deliverable D5.4.1, NeOn Project, 2008.

[18] T. Tudorache, others Üstün, M.-A. D. Storey, and M. A. Musen. Ontology development for the masses: Creating ICD-11 in WebProtégé. In *Proc. of EKAW'10*, volume 6317 of *LNCS*, pages 74–89. Springer, 2010. Lisbon, Portugal.