# Dimensions Affecting Representation Styles in Ontologies

Pablo Rubén Fillottrani[1,2] and C. Maria Keet[3]

[1] Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina, prf@cs.uns.edu.ar
[2] Comisión de Investigaciones Científicas, Provincia de Buenos Aires, Argentina
[3] Department of Computer Science, University of Cape Town, South Africa
mkeet@cs.uct.ac.za

**Abstract.** There are different ways to formalise roughly the same knowledge, which negatively affects ontology reuse and alignment and other tasks such as formalising competency questions automatically. We aim to shed light on, and make more precise, the intuitive notion of such 'representation styles' through characterising their inherent features and the dimensions by which a style may differ. This has led to a total of 28 different traits that are partitioned over 10 dimensions. The operationalisability was assessed through an evaluation of 30 ontologies on those dimensions and applicable values. It showed that it is feasible to use the dimensions and values and resulting in three easily recognisable types of ontologies. Most ontologies had clearly one or the other trait, whereas some were inherently mixed due to inclusion of different and conflicting design decisions.

## 1 Introduction

Ontology developers and logic-savvy domain experts are familiar with the question "how to formalise it?", which has been asked for many years [2], observing the problems of alternative ways of representing the same piece of information or knowledge. For instance, whether to represent the entity Marriage as a class or as relationship, say, married to and whether to represent Spouse as a subclass of Person or as a role that a person plays. Some ontologies may exhibit more than one alternative modelling options, which may result in conflicting decisions in the modelling stage, such as with the Organisation ontology [18] that is being standardised and updated with the W3C, which has both Membership as a class and memberOf as an object property. When the same modelling decision is chosen throughout an ontology, including, e.g., process-as-class, this gives rise to the notion of a 'representation style'.

Besides the philosophical questions whether one way of representing a piece of knowledge is truly better than another, having such alternative ways of representing it in an ontology poses a range of engineering issues. A notion of an "ontology building style" was already observed with several bio-ontologies [20], as was style due to representation language [21], identification of "regularities"

in ontology metrics [12] and "emerging design patterns" based on axiom patterns in ontologies [10], and alternate "ontology patterns" that roughly fit a 'foundational' and 'applied' style [4], but what the 'styles' are precisely, is not clear. On top of its lack of clarity, representation incompatibilities between ontologies hamper easy reuse; e.g., when the two ontologies have represented the domain at different levels of detail or other choices, such as class vs object property like aforementioned Marriage vs married to [4]. Knowing the style upfront could be a parameter for decision as to which ontology to reuse. Even before reuse, the issue can present itself, however. Wiśniewski et al. [23] manually converted competency question into SPARQL-OWL queries, observing that one competency question (CQ) pattern (a sentence without the ontology's vocabulary) can map onto multiple SPARQL-OWL signatures, i.e., the same information request can be realised in different ways. Knowing the style would thus be helpful for automating converting CQs into SPARQL-OWL queries as well as into axioms [8, 3]. This remains a laborious manual process for as long as there is no clear insight into which ways this can be done systematically.

In this paper, we aim to shed light on the fuzzy notion of 'representation style' by aiming to characterise its inherent features and the dimensions by which a style may differ. Some values of dimensions combine well together, some do not. This is subsequently evaluated with 30 ontologies to determine its potential for operationalisation so that one could classify an ontology to be of one or another style or a mix thereof and whether perhaps a dimension was missed. The first round of testing revealed the need for, especially, more precise descriptions to evaluate an ontology on that dimension, which have been added. The second round of evaluation showed high inter-annotator agreement and revealed preliminary trends among the test ontologies, especially in the values between the theory-oriented ontologies versus applied ontologies. Knowing an ontology's representation style or choosing one for an ontology that is yet to be developed is expected to streamline and support further automation of the use and evaluation of CQs, ontology alignment, and the option to automatically change one's style, and improve the quality of one's own ontology though adhering to the same modelling decision.

The remainder of the paper is structured as follows. Section 2 presents the main theoretical contribution of the attempt to clarify what a representation style is (and is not) and elucidates its dimensions. Section 3 presents the evaluation of classifying a set of different types of ontologies. We discuss in Section 4 and conclude in Section 5.

## 2 Representation styles

We first aim to characterise the fuzzy notion of 'representation style', why some artefacts that sound relevant are not styles, and discuss related work. Section 2.2 lists and justifies the dimensions for determining a style.

### 2.1 Core features of a representation style

The first main question one should ask is: *what is a representation style?* We identified the following set of characteristics features toward a specification:

- It is a way of representing some piece of knowledge or information in a particular way;
- It may be formalised differently in different logics, and differently even in the same logic, depending on the constructors at one's disposal, which indicates it is a sort of 'conceptual thing' where there is informally meaningful sameness, but possibly no axiomatic equivalence;
- It is independent of the subject domain;
- A style is a possible/common/well-known way of representing some piece of knowledge. This does not mean that that way of representing it is an unqualified 'good' way of doing things, and some style may be bad for some purpose;
- They are knowledge engineering/representation decisions (not ontological) a modeller takes about the conceptual thing.

These aspects considered, we arrive at the following short-hand description.

**Definition 1 (Representation style, first version).** *A* representation style *is a way of representing a particular piece of conceptualisation or (understanding of) reality, for which there are alternative ways to represent it formally that are generally considered to be meaningfully equivalent (though not necessarily logically).*

The list above and the shorthand definition are still very generic, in that it would include artefacts of which one can state they are not representation styles. To clarify this further, let us look at related works that allude to styles and similar notions, but are excluded from 'style' for various reasons, i.e.: what is *not* a style, but sounds closely related nonetheless?

- A content Ontology Design Pattern (ODP) [5]. At best, a content ODP is an *instantiation of* a style, adhering to the principles of a particular style. An ODP, if not formalised by specification, may be formalised in different ways. This may make it look like a style, but the content ODP is for that ontology with that subject domain specifically. What might be the case is that when one analyses multiple informal content ODPs, an (instantiation of) a style might emerge.
- Anti-patterns (e.g., [19]). They are at least sub-optimal, if not outright bad, modelling decisions, which is largely due to inexperience of the modellers. Conversely, general guidelines are then also not styles by definition.
- One or more related axioms. Such a set, or mini-module or micro-theory, may respect or not violate a representation style. This thus also means one will not be able find a representation style bottom-up through mining a large or small set of ontologies, alike the pattern/style/regularities finding in [10, 13, 12]. This also suggests that recurring collections of axioms/mini-modules

(say, 'axiom signature') do not make a representation style, because it can be formalised in different ways. A simple example of different formalisations would be using inverse readings or not (e.g., has part and part of vs. has part and Inv(has part)).
– Workarounds for language limitations, because by its definition a workaround is not a style. A well-known workaround is an approximation of reification due to the lack of $n$-aries in OWL [16].

The notion of "ontology building style" featured in [20] when they compared several bio-ontologies, which included the following parameters: application scenario, modularisation, domain/task/generic, instances or not, level of detail, and representation language. An application scenario of itself is not a style, but one may *choose a style* so that it fits with one or more desired application scenarios. Modularisation or not and instances or not certainly would indicate a representation style: the former on how one structures the knowledge and the latter on whether the artefact is intended as a database or knowledge base or as an ontology in the 'purist' sense (TBox only or ABox only). A particular representation language does not constitute a style or a dimension thereof, however, but may *enforce a style*. For instance, some designers may have decided that one is allowed to use only a few relations (say, BT, NT, RT, USE/UF in thesauri), which is then hard-coded in the representation language (e.g., SKOS [14]). Then SKOS is not the style, but the decision of few relations is a (component or value of a) style. The same decision was taken for, e.g., OBO, which percolates through to the OWL-ised OBO ontologies [7]. Finally, the 'level of detail', or granularity in representation, is a design decision to take, but possible values for such dimension are difficult to set and evaluate consistently.

The same group also proposed automatically computed "ontology design styles" [13], using 20 OWL metrics applied to the Tones repository and clustered them into five groups. Numbers of elements do no say anything about what has been represented, however; e.g., a 'two classes, two object properties, and two class expressions' does not reveal whether the modelling choice has been made to represent processes as classes or as object properties. As noted above, and also in Mikroyannidi et al.'s follow-up works on this approach [12]: a particular axiom pattern or "regularity" may only exemplify a particular realisation of a style.

Wisniewski et al. use the term "modelling style" [23] to refer to what is dubbed here 'representation style', and is thus not to be confused with a different use of "modelling style" as in, e.g., [17], that refers to the process of creating the model, like in which possible sequences of steps the information or knowledge is added to the ontology with as aim to discover editing styles [22].

Two threads reappear in these related topics, which are 1) the alternate possible formalisations of some piece of domain knowledge so that axioms or a logic alone cannot be a style but at best exhibit one, and now also 2) excluding 'negatives' such as common mistakes and workarounds, being declarations in the ontology that no one would want as a conscious choice from among known alternatives and in the ideal case. Definition 1 can thus be refined into:

**Definition 2 (Representation style, refined version).** *A* representation style *is set of features used for representing a particular piece of conceptualisation or (understanding of) reality, for which there may be different (meaningfully equivalent, but not necessarily logically equivalent) ways to represent it in a logic that supports the style. The representation style is a justification-based positive design decision in at least one scenario.*

### 2.2 Dimensions for determining a style

The dimensions are separated into two main categories or 'levels': one is predominantly theoretical and concerns the type of artefact one aims to create irrespective of the practical considerations, whereas the other set of dimensions amount to engineering decisions.

1. Level 1: how 'ontological' the ontology is or should be.
   (a) *Degree of adherence to ontological principles in representing the knowledge.* The two extremes on the scale are the "foundational ontology way", i.e., predominantly theory-focussed, and the "applied way" where the ontology is, arguably, a logic-based conceptual data model. The style differences fall in place because of this decision; e.g., using qualities and qualia to represent attributions, such as a class Colour, versus data properties, such as hasColour:`String`. The values can be set as follows:
   `Theoretical`: predominantly or entirely with ontological principles, such as qualities, reification of processes, inherence of roles, no data properties.
   `Applied`: predominantly or entirely with decisions for applications, such as attributes/data properties and processes as relations.
   `Mixed`: the ontology contains both such decisions.
   (b) *Granularity of relations.* The two extremes on the scale are parsimony and abundance. With the former, one chooses to limit oneself to a few core relations, such as parthood, participation, causality, and membership; with the latter, one declares relations for every subtle distinction, such as a structural parthood relation between physical objects as subtype of the generic part-of and, e.g., a celebrates relation that refines a participates in. The values can be set as follows:
   `Parsimony`: when there are no refinements of the basic relations.
   `Abundance`: when there are refinements on the basic relations or when there are domain-specific relations, or both.
   (c) *Classes or instances.* From an ontological viewpoint, an ontology is about classes or about instances, but would not combine the two into what is historically called a knowledge base. The values can be set as follows:
   `Class-level only`: the ontology does not contain individuals (i.e., for an OWL file, the ABox is empty), or only very few such that it is intended as an illustrative example or attempted workaround.
   `KB-like`: the ontology has many individuals in the 'ABox' in addition to the class-level knowledge ('TBox').

2. Level 2: engineering, or consequences of praxis
   (a) Tooling/user interface effects:
      i. *Hierarchy, flat, or meshed* structure of the knowledge represented; e.g., a specific TBox tab with the hierarchy or a hierarchical lay-out graphically (OBO-Edit, Protégé) versus other graphical interfaces that are inspired by, say, UML or ER notation (Icom). The values can be set as follows:
         `Hierarchy, bare`: there is a hierarchy with one or more branches of substantial ($\geq 4$) depth[4], but barely or no class expressions have been declared on the classes.
         `Hierarchy, mesh`: there is a hierarchy with one or more branches of substantial ($\geq 4$) depth, and many class expressions have been declared on the classes, likely to classes residing in other branches of the hierarchy.
         `Flat`: there is no substantial hierarchy (typically a depth of $\leq 3$), but still properties are declared among the classes.
      ii. *Modular vs monolithic*; some tools have module management incorporated in the tool (e.g., Icom, Ontohub), facilitating dividing the subject domain into modules, versus not having that option by default (e.g., Protégé, Moki). The values can be set as follows:
         `Monolithic`: there is one file, with no imports or mergers.
         `Modular, external`: at least one ontology is imported or merged such that the import has maintained its IRI; hence, it is associated with the process of ontology reuse.
         `Modular, internal`: at least one ontology is imported, such that it is associated with the process of decomposition of a domain.
      iii. *General Concept Inclusions (GCIs) vs only named entities* on the left-hand side of the inclusion; e.g., Protégé's interface for declaring GCIs is hidden and text-based only, therewith discouraging its use. The values can be set as follows:
         `Explicitly declared GCIs`: they have been declared by the modeller, such as $\mathsf{Property} \sqcap \exists \mathsf{propertyOf}.\mathsf{Presential} \sqsubseteq \mathsf{Presential}$ in gfo-basic.
         `Hidden GCIs`: they have not been declared by the modeller explicitly, but they are there indirectly through other axioms: there is a pair $A \equiv C$ and $A \sqsubseteq D$ and $C$ and $D$ are complex class expressions[5].
         `No GCIs`: they have not been declared explicitly or implicitly.
      iv. *A separate RBox manager* interface that facilitates reuse of relations (notably: Protégé), compared to adding new ones each time one uses essentially the same relation (most conceptual modelling tools and ontology editors inspired by them). The values can be set as follows:

[4] A cut-off at 3 layers is subjective, based mainly on the experience that a 2-3 layer mini-hierarchy is easy to declare, whereas 4 or more requires some thought to put it right, and conceptual models typically do not have more than 2-3 layers, if there is a hierarchy at all.

[5] `http://protegeproject.github.io/protege/views/ontology-metrics/`

`Relation reuse`: object properties declared in the RBox typically appear in more than one class expression (beyond domain and range declarations).

`No reuse`: object properties declared in the RBox typically do not appear in a class expression (other than, possibly, domain and range axioms).

(b) Language feature effects

i. *Hierarchy, flat, or meshed* structure of the knowledge represented; the language has more or less features to add relations, resulting in largely a 'bare' hierarchy where a class has no or very few properties vs few hierarchies but many interconnected elements; e.g., compare the restricted set of relations in OBO and SKOS vs OWL where one can freely and easily specify new relations. The values can be set as described above in Item 2(a)i.

ii. *Modular vs monolithic*; whether the language supports axioms for handling modules, such as OWL's `import` statement, and how the modules relate: if not or with difficulty, then the ontology necessarily will be monolithic. The values can be set as described in Item 2(a)ii.

iii. *The commitment built into the language for 'standard view' or 'positionalist'* and, within the former, whether the language has the inverse feature; e.g., whether to represent "eating" (or any other relation) with 1) a 'forward' and 'backward' reading direction as two relations, eats and eatenBy, and declare them inverses (eats $\equiv$ eatenBy$^-$), as 2) a relation with one reading direction only (eats) and the other implicitly without a vocabulary element (Inv(eats) meaning 'eaten by'), or 3) no reading direction (eating) and two roles the objects play in that relation (say, predator and prey).[6] The values can be set as follows:

`Standard view, with inverses`: as described in option 1, with forward and backward readings.

`Standard view, with Inv()`: as described in option 2, in one direction only.

`Standard view, both`: if both option 1 and option 2 are used in the ontology.

`Positionalist`: as described in option 3.

iv. *Values/instances/classes interplay.* Representation of certain entities that may be deemed different kind of elements, depending on one's modelling viewpoint, practicalities, and which constructors are available in the language by means of 1) nominals, 2) enumerated datatypes, or 3) class and instances; e.g., the days of the week[7]. The values can be set as follows:

[6] There are arguments from Ontology for option 1 and 2 vs option 3, but the intended choice here the corresponding commitment built into the language.

[7] the intended choice here is also the commitment built into the language, not theory (option 3) versus applied (options 1 and 2).

`Nominals`: as described in option 1; i.e., Week ≡ {Sunday, ...., Saturday} where Sunday etc. are individuals.

`Enumerated`: as described in option 2; i.e., the values of a data property onWeekday can be one of the values `Sunday, ...., Saturday`.

`Class-instance`: as described in option 3; i.e., they are all classes appropriately related, and a 'Sunday 6 January 2019' is an instance of Sunday etc.

`Mixed`: any two or three appear in the ontology.

`-`: not applicable.

v. *Attributes and data types* as core language feature. The inclusion of attributes (OWL data properties) and, with it, datatypes, has certain consequences from the viewpoint of ontology—theory vs applied (Item 1a) and could be considered to be subsumed by that item, as would other theory/applied decisions be. It has been added separately, as it is of a different calibre than whether, say, qualified number restrictions are supported, because it represents a certain kind of element, at the same level as class and object property. The values can be set as follows:

`Attributes`: the language has that feature and it is being used.

`No attributes`: the language may or may not have that feature, but it is not used anyway/not present.

Note that if the representation language permits more than one option, an ontology may exhibit instances of more than one modelling style, i.e., contain conflicting design decisions, and therefore the 'both' and 'mixed' have been added to the options. For instance, using both nominals (instances as classes) and class&instances for the days of the week, or both the class Colour and a data property hasColour. Further, note that representation language is a dimension, but not a particular language, such as, e.g., the OWL 2 RL profile, because the latter incorporates selected traits (values) of the dimension rather than determining traits.

For the 'level 1' dimensions and values, it is possible to construct a simple matrix of options with the four possible combinations: Theoretical & Parsimony, Theoretical & Abundance, Applied & Parsimony, and Applied & Abundance. Adding the 'level 2' dimensions with their possible values would result in a combinatorial explosion of theoretically possible styles. This would be exacerbated if it were to be for all the specific tools and languages. On purely theoretical grounds, one should be able to exclude certain combinations upfront already. For instance, Some consequences that should follow from the theory are:

a. the ontologically abhorrent nominals and enumerated data types (Item 2(b)iv) should not appear in a foundational ontology or theory-focussed ontology;
b. there will not be GCIs in applied ontologies/conceptual data models, because such models never had either explicitly declared or implicit GCIs, except for simple domain and range declarations.
c. If no tool is used in ontology development, then no option in Item 2a can be a cause.

d. If one uses a representation language that permits only few relations (e.g., `.obo` ontologies, SKOS), then the relation reuse (Item 2(a)iv) is built into it.
e. If one uses a representation language that permits only few relations, then the granularity of the relations is that of parsimony (or vice versa in direction of causality: one chooses parsimony, and enforces it in the language).
f. The theory option (Item 1a) ought to go with option 3 in Item 2(b)iv and, likewise, the applied option (Item 1a) ought to go with option 2 in Item 2(b)iv, which is the typical approach in conceptual data models.
g. If the language cannot handle modules well (Item 2(b)ii), then so will the corresponding tool not have good module management and result in monolithic ontologies (Item 2(a)ii), and, one would expect, vice versa.
h. Theory-oriented ontologies (Item 1a) do not use data properties/attributes as a recurring feature other than, perhaps, a `hasValue xml:anyType` (Item 2(b)v).
i. Given that hierarchy/mesh/flat and monolithic/modular are listed twice (as Item 2(a)i & Item 2(a)i and Item 2(a)ii & Item 2(b)ii), one will not be able to say with certainty which one is the core cause, if any, when trying to assess an ontology on its style.

It remains to be seen which of these expected consequences hold up in practice.

Finally, it is possible to devise multiple reasons to choose one particular style over another, which are, effectively *imposed* engineering decisions, rather than engineering decisions based on theoretical foundations (as listed under 'level 2'). These may include, among others:

- Language and tool support; e.g., at the time of writing, OWL and Protégé are the most popular among the options, therefore, the ontology is necessarily `standard view`.
- Other ontologies one has to link to/import adhere to that particular style.
- Need to follow corporate or consortium styles.
- Efficiency in the implementation (prospected use of the ontology-driven information system); e.g., one cannot link an ontology in the expressive OWL 2 DL language to a database for Ontology-Based Data Access and expect the same level of performance as for relational databases, simply due to inherent computational complexity limitations (and with implementation, one veers toward at least `applied`, and possibly also `attributes`, `enumerated`, and `no GCIs`).
- Parsimony in the representation, reducing cognitive load by humans who author and read the ontology or conceptual data model, lowering the barrier of uptake of the artefact, which thus may save money.

With respect to representation styles, we will not consider them, because they all already have a style implied, i.e., they do not determine, but impose already determined styles, and we are interested in what determines a style.

## 3 Applying Style Dimensions to Ontologies

The aim of this evaluation is to determine whether the dimensions and traits identified in the previous section can be applied to existing ontologies, whether

they suffice or another one or more have to be added, and to see whether some combinations of dimensions already emerge from the data. The materials and methods are described first, and subsequently the results and discussion thereof.

### 3.1 Materials and Methods

The following process was followed:

1. Select different types of ontologies: foundational, core, domain, and tutorial.
2. Classify them manually, using the dimensions and their appropriate values. This is carried out by two people (the authors) independently.
3. Check for inter-annotator agreement and whether the identified dimensions suffice; if there is disagreement, either:
   (a) Harmonise and move to Step 4, below;
   (b) Resolve conflict in classification, refine either the affected dimension's value or description thereof, and return to Step 2.
4. Analyse the data on expected consequences (see previous section; e.g., foundational ontologies without nominals and data properties) and on whether any recurring combinations of dimensions emerged.

The ontologies were handled with Protégé, WebVOWL, and the OWL Classifier tools whenever the OWL code was available. The only exceptions were SNOMED CT (BioPortal browser) and DOLCE on paper.

It is factored in that two rounds of annotations may be needed: while the list of dimensions is deemed comprehensive at the start of the evaluation, the actual inspection of the ontologies may result in emerging refinements thereof. This is facilitated by the qualitative approach taken (cf. computed metrics).

The top and tutorial ontologies were collected based on cognisance by at least one of the authors; please refer to the supplementary material online for their references[8]. The relatively well-known selected top-level ontologies are BFO, DOLCE (on paper and in OWL), GFO, SUMO, and YAMATO, and the tutorial ontologies are Pizza, Wine, Family, Books, and AWO. Six core ontologies were selected based on cognisance (stuff, biotop, time), considering general topics (e.g., events, units) and checking whether there is an ontology for it, and a simple online search for "core ontologies" (services). The selection criteria for the domain ontologies were: some of our own as that would make analysis easier (Adolena, DMOP), the most accessed ontologies from BioPortal[9], based on their respective names that sounded like a real ontology (FMA, DIOD, EDAM, SNOMED CT), and an arbitrary selection of W3C ontologies that appeared on the main/news page scrolling down for recent activity on it (SSN, Profiles, Organization and related UNDO).

[8] `http://www.meteck.org/files/StyleDimensionsData.xlsx`
[9] `https://bioportal.bioontology.org/visits`

### 3.2 Results and Discussion

The first round of annotation resulted in a low inter-annotator agreement, mainly because the scope of dimensions and several values turned out to have been interpreted differently and found in need of more precision. For instance, i), instead of making only a distinction between yes/no GCIs, it is useful to know whether a 'yes' was deliberately added explicitly by the modeller or not; ii) instead of 'hierarchy vs flat/mesh', where the need for three values emerged to capture the variations better (see above); iii) instead of monolithic vs modular, an ontology may be modular through either reusing one or more ontologies and dividing up one's intended subject domain to make it more manageable. This resulted in the specification of possible values with a description of each, in order to better annotate each ontology, which have been included in the previous section. Also, when examining the ontologies, it was deemed useful to record whether it uses data properties, which also has been added (Item 2(b)v in the previous section), and whether a substantial number of individuals have been declared (Item 1c).

The same selected ontologies were then categorised anew and the data compared among the annotators, resulting consensus on the traits and interpretations. Some styles were found to be non applicable to all ontologies, and the lack of a formal version of SNOMED CT (only available through browsing a web page) made some categories difficult or impossible to evaluate. The complete results are available at the URL noted in fn. 8 and a visualisation is included in Fig. 1, where the 10 clearly `Theoretical` and `Applied` ontologies were selected and the annotations coded to generate the graph (e.g., `No attributes` as 1.0 and `Attributes` as 2.0, `Class-level only` as 0, an occasional class as 1.0, and (full-fledged) `KB-like` as 2.0).

In general, some expected outcomes were confirmed. Applied ontologies exhibit more usage of nominals, instances, and data properties. Notable exceptions are DOLCE and DMOP with a few data properties and some class instances. Since DMOP links to DOLCE and was redesigned to facilitate alignment to DOLCE [9], this is an 'inherited' style. Other expected results are the facts that either explicit or hidden GCI are mostly absent from applied ontologies (only MEO has a large number of hidden GCIs); that Books can be seen more as a conceptual data model than as an ontology; and that almost all flat ontologies have no relation reuse apart from domain/range declarations.

There were also some outcomes that we did not expect. The number of theoretical ontologies with none or only hidden GCI is higher than expected, suggesting a low level of usage for advanced ontology expressions. Relatively poor support of GCIs in popular tools such a Protégé may be a cause for this. Also, this justifies the independence of the language and tooling dimensions and distinction explicit/implicit for GCIs. The categorisation of SUMO as an applied ontology was also unexpected, because it is intended as as an "upper" ontology by its developers [15]. Several ontologies declare domain and range axioms for object properties, but these declarations are mostly not reflected in other class expressions. This was especially different between theoretical and applied

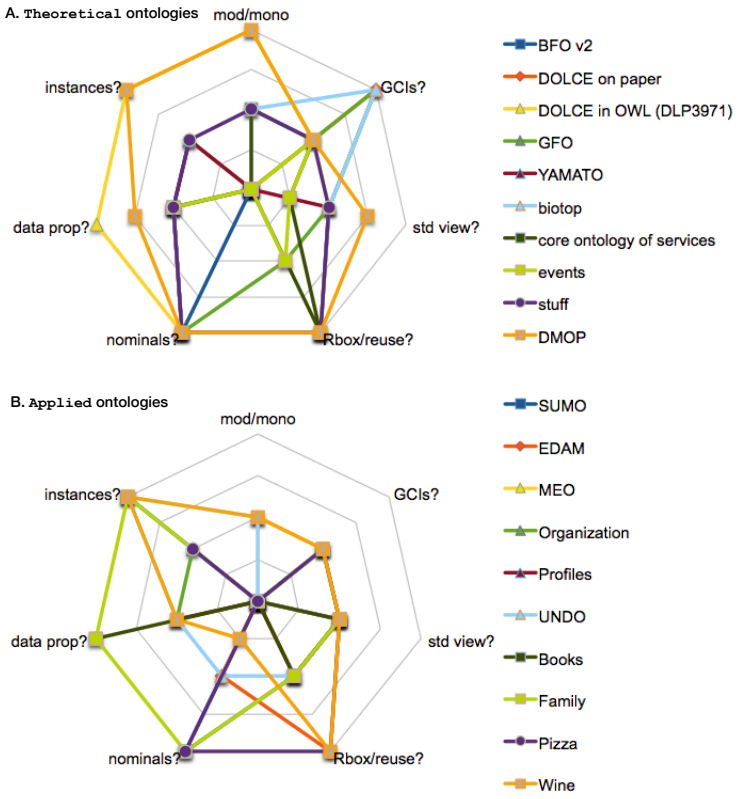


**Fig. 1.** Visualisation of coded annotations of ontologies that were clearly `Theoretical` and clearly `Applied` ontologies. The values of the dimensions for the evaluated ontologies have been scaled to a scale of 0-2 to produce the graph; e.g., no GCIs = 0, hidden GCIs = 1, and explicit GCIs = 2.

ontologies, as also shown in Fig. 1: 7 out of 10 ontologies do reuse object properties, whereas that was the case for only 3 out of the 10 applied ontologies. Yet, parsimony of relations was rarely found in any ontology (only in BFO, DOLCE on paper and MEO). As a future work, a further refinement of this dimension might be possible.

## 4 Discussion

The formulation of the dimensions and values that influence a representation style, while originating from theory, did show it is operationalisable. To arrive at that stage, it was crucial to have had the evaluation with independently made annotations that the annotators discussed for each disagreement. This mainly served to elucidate implicit assumptions, clarify them, and formulate them more precisely, where possible.

Reconsidering the expectations based on theory (Section 2.2) as compared to the results of the evaluation, most of those that could be assessed indeed did hold (refer to online supplementary material for details). Exceptions are that no ontology used enumerated data types, even though it is allowed in OWL except for OWL-Lite[10], and while there are no explicit GCIs in applied ontologies, most of them do have hidden GCIs. Also, comparing Fig 1-A with B does show a different trend among the theoretical and applied ontologies, but it must be noted that each set has only 10 ontologies, which is a small set to draw conclusions from about emerging patterns. The positive trend toward difference merits further investigation to determine whether it holds for more ontologies and also why. For instance, one might expect `Modular, external` for applied ontologies that would be importing other ontologies and save themselves design time, but this did not occur and our data does not reveal why.

It is clearly faster to run metrics over an OWL file to find regularities, like in [13, 10], compared to the manual and more qualitative approach to annotating ontologies that was taken here. The regularities do not reveal modelling decisions, however, such as granularity in relations or whether attributions are represented in a class hierarchy, and it excludes all non-OWL ontologies, such as DOLCE on paper [11] in full first-order logic, the BFO Core[11], and the COLORE repository ontologies [6] that are represented in Common Logic [1]. The list presented in Section 2.2 also can be applied to ontologies not formalised in OWL, for it is independent of the representation language, and, in fact, explicitly caters for it with the 'language feature effects' dimensions and values (Item 2b). It therewith also would assist in solving language issues as described by (but not systematically solved in) Uschold et al. [21], in addition to contributing to solving modelling decisions. Further, we have identified more dimensions than in [20] and also specified criteria when which value applies, which is thanks in a considerable part to the uptake of ontologies, ontology engineering research, and tooling support over the past 20 years. Finally, the ontology patterns in [4] are all examples of representing some piece of knowledge in the `Applied` way vs `Theoretical` way, hence, constitute a subset of the dimensions we propose in this paper.

Whilst trying to optimise the task of annotating the ontologies, the `No reuse` of relations typically went together with `Hierarchy, bare` or `flat`, and there were (more) data properties in `Applied` ontologies compared to `Theoretical`, which can be expected. It may be of interest to carry out the experiment with more ontologies so as to discover patterns in the values, as well as examine past ontology alignment and integration efforts on whether their ease/difficulties match with the dimensions proposed in this paper.

---

[10] `https://www.w3.org/TR/owl-ref/#EnumeratedDatatype` as `owl:OneOf` and in OWL 2 as `DataOneOf` (`https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Enumeration_of_Literals`)

[11] `http://www.acsu.buffalo.edu/~bittner3/Theories/BFO/`

## 5 Conclusions

The paper presented a comparatively comprehensive set of dimensions that contribute to determining representation styles in ontologies, making their intuitive notions more precise and providing a conceptual grounding step towards their formal characterisations. Each dimension is presented with a given set of possible values and their sentinel pointers to identify them in an ontology, and what other dimensions are related. We have evaluated this theory on applicability with 30 well-known published ontologies, classifying and comparing approaches and analysing the interactions and relationships between the different dimensions, laying the foundations for future research that can establish better modelling support.

Good engineering practice to develop ontologies consists of not only applying scientific techniques but also by taking modelling design choices. Such choices are taken early in the process and can have far reaching consequences. Therefore, identifying and characterising these styles—by availing of the here presented dimensions—constitute a step toward more comprehensive best practices for ontology engineering tasks such as alignment, integration, evolution, and data access. Furthermore, since style reveals basic characteristics of an ontology, it can promote the development of automated processes in tools and methods. Future work includes, therefore, among others, testing the dimensions in additional experiments both regarding more ontologies and with more ontology developers, examining the framework as a whole, analysing the interplay between styles, patterns, and quality attributes, as well as developing ontology engineering methodologies involving dimensions and styles, and ultimately studying the influence of representation style on the resulting ontology.

## References


1. Common Logic (CL): a framework for a family of logic-based languages (2007), `https://www.iso.org/standard/39175.html`
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logics Handbook – Theory and Applications. Cambridge University Press, 2 edn. (2008)
3. Fernandez-Izquierdo, A.: Ontology testing based on requirements formalization in collaborative development environments. In: Aroyo, L., Gandon, F. (eds.) Doctoral Consortium at ISWC (ISWC-DC'17). CEUR-WS, vol. 1962 (2017), vienna, Austria, October 22nd, 2017
4. Fillottrani, P.R., Keet, C.M.: Patterns for heterogeneous tbox mappings to bridge different modelling decisions. In: Blomqvist, E., et al. (eds.) Proceeding of the 14th Extended Semantic Web Conference (ESWC'17). LNCS, vol. 10249, pp. 371–386. Springer (2017), 30 May - 1 June 2017, Portoroz, Slovenia
5. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 221–243. Springer Verlag (2009)
6. Grüninger, M., Hahmann, T., Hashemi, A., Ong, D., Ozgovde, A.: Modular first-order ontologies via repositories. Applied Ontology 7(2), 169–209 (2012)

7. Hoehndorf, R., Oellrich, A., Dumontier, M., Kelso, J., Rebholz-Schuhmann, D., Herre, H.: Relations as patterns: bridging the gap between OBO and OWL. BMC Bioinformatics 11(1), 441 (2010)
8. Keet, C.M., Lawrynowicz, A.: Test-driven development of ontologies. In: Sack, H., et al. (eds.) Proceedings of the 13th Extended Semantic Web Conference (ESWC'16). LNCS, vol. 9678, pp. 642–657. Springer, Berlin (2016), 29 May - 2 June, 2016, Crete, Greece
9. Keet, C.M., Lawrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., Hilario, M.: The data mining optimization ontology. Web Semantics: Science, Services and Agents on the World Wide Web 32, 43–53 (2015)
10. Lawrynowicz, A., Potoniec, J., Robaczyk, M., Tudorache, T.: Discovery of emerging design patterns in ontologies using tree mining. Sem Web J. p. in press (2018)
11. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: Ontology library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003). (2003), http://wonderweb.semanticweb.org
12. Mikroyannidi, E., Iannone, L., Stevens, R., Rector, A.: Inspecting regularities in ontology design using clustering. In: Proceedings of the International Semantic Web Conference (ISWC'11). LNCS, vol. 7031, pp. 438–453. Springer (2011)
13. Mikroyannidi, E., Stevens, R., Rector, A.: Identifying ontology design styles with metrics. In: 7th International Workshop on Semantic Web Enabled Software Engineering (SWESE) (2011)
14. Miles, A., Bechhofer, S.: SKOS Simple Knowledge Organization System Reference. W3c recommendation, World Wide Web Consortium (W3C) (18 August 2009), http://www.w3.org/TR/skos-reference/
15. Niles, I., Pease, A.: Towards a standard upper ontology. In: Welty, C., Smith, B. (eds.) Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001) (2001), ogunquit, Maine, October 17-19, 2001
16. Noy, N., Rector, A.: Defining nary relations on the semantic web (April 2006), https://www.w3.org/TR/swbp-n-aryRelations/., w3C Working Group Note, 12 April 2006
17. Pinggera, J., Soffer, P., Fahland, D., Weidlich, M., Zugal, S., Weber, B., Reijers, H.A., Mendling, J.: Styles in business process modeling: an exploration and a model. Software & Systems Modeling 14(3), 1055–1080 (2015)
18. Reynolds, D.: The organization ontology (January 2014), https://www.w3.org/TR/vocab-org/
19. Roussey, C., Corcho, O., Vilches-Blázquez, L.: A catalogue of OWL ontology antipatterns. In: Proc. of K-CAP'09. pp. 205–206 (2009)
20. Stevens, R., Goble, C.A., Bechhofer, S.: Ontology-based knowledge representation for bioinformatics. Briefings in Bioinformatics 1(4), 398–414 (2000)
21. Uschold, M., Healy, M., Williamson, K., Clark, P., Woods, S.: Ontology reuse and application. In: Guarino, N. (ed.) Proc Int Conf on Formal Ontology and Information Systems (FOIS'98). FAIA, IOS Press (1998)
22. Walk, S., Singer, P., Strohmaier, M., Tudorache, T., Musen, M.A., Noy, N.F.: Discovering beaten paths in collaborative ontology-engineering projects using markov chains. Journal of Biomedical Informatics 51, 254–271 (2014)
23. Wisniewski, D., Potoniec, J., Lawrynowicz, A., Keet, C.M.: Competency questions and SPARQL-OWL queries dataset and analysis. Technical Report 1811.09529 (November 2018), https://arxiv.org/abs/1811.09529