Quality of Content Ontology Design Patterns

Karl Hammar

Jönköping University, Linköping University

1 Introduction

The idea of using Ontology Design Patterns, particularly Content Ontology Design Patterns (hereafter CODPs) to support Ontology Engineering has gained traction within the academic community, as evidenced by the Workshop on Ontology Patterns series of workshops held in conjunction with the International Semantic Web Conference, and the recent formation of the Association for Ontology Design & Patterns (ODPA). Adopting CODPs and the associated eXtreme Design [24] (hereafter XD) Ontology Engineering method allow developers to reuse known good solutions to modelling problems, and thus hopefully enables these developers to work in a more efficient manner.

However, as the author has previously shown [14], the published work on Ontology Design Patterns is lacking in some aspects. While many patterns have been presented, not enough work has been done on evaluating the structure and design of those patterns. Consequently, little is known about what qualities or properties of patterns are beneficial in ontology engineering tasks, and inversely, what properties are not helpful or are possibly even harmful in such tasks.

This chapter summarises the author's recent work on remedying the aforementioned lack of knowledge, by developing a CODP Quality Model. This work is important for three main reasons:

Firstly, if Semantic Web technologies are to reach broader adoption outside of academia, the academic community needs to present a tenable value proposition – we need to offer mature methods, patterns, and tooling that show industry clear benefits over alternative approaches. Given the relative newness of Semantic Web technology and the complexity of the standards and tool stack, both CODPs and CODP tooling need to be of sufficient quality that they can be used easily by non-expert ontologists, i.e., not only academic prototypes. We need to understand and improve their quality.

Secondly, there are several notions with regard to CODP structure and quality that are assumed, but not evaluated. For instance, it is generally

agreed that CODPs should be small – but is this always true? And if so, how small is small? Or it is assumed that the shared conceptualisation brought about by the many CODPs that reuse ideas from the DOLCE ontology provides a degree of interoperability which is beneficial – but does this not constrain the use of those CODPs? Providing a framework for how to speak of and measure quality allows the CODP community to test these assumed hypotheses and possibly break preconceived notions.

Thirdly, as illustrated quite clearly by the example given in the previous paragraph, there are several cases where CODP qualities clash with one another. Another common case of such clashes are between reasoning performance efficiency on the one hand and full logic expressiveness on the other. This is of course not news to Semantic Web researchers, but it is important that we understand these clashes and the tradeoffs they give rise to, when pitching CODPs as a solution to industry or to researchers from other fields.

The chapter is organised as follows: Section 2 summarises existing work on quality evaluation for ontologies, models, and software, upon which the latter sections extend. Section 3 introduces and describes the CODP Quality Model, consisting of a set of quality characteristics and quality indicators relevant to evaluate CODPs by. Section 4 suggests areas for further work in CODP quality research, exemplifies some tensions between qualities and discusses the resulting tradeoffs to be made when designing and using CODPs, and recommends improvements to CODP tools based on the work presented here, before Section 5 closes the chapter.

2 Related Work

The following section presents existing approaches to quality evaluation which have been reused in the construction of the CODP Quality Model. Section 2.1 discusses evaluation and quality of related non-ontological resources (information systems, ER models, and OOP patterns), while Section 2.2 discusses related work on ontology evaluation.

2.1 Quality in Related Fields

2.1.1 Information System Quality

Ontologies are almost always used as components within an information system. As such, research on software artefact quality is useful in understanding the demands on an ontology from an information system perspective. This field has seen considerable work going back to the 1970s. In particular, the ISO standard 25010 [16] introduces a *Product Quality Model* which covers many aspects of quality that are transferable to a CODP context.

Table 1: ISO 25010 Product Quality Model (adapted from [16])

Qual. characteristic	Sub-Characteristic	Qual. characteristic	Sub-Characteristic
Func. suitability		Reliability	
	Functional completeness		Maturity
	Functional correctness		Availability
	Functional appropriateness		Fault tolerance
Perf. efficiency			Recoverability
	Time behaviour	Security	
	Resource utilisation		Confidentiality
	Capacity		Integrity
Compatibility			Non-repudiation
	Co-existence		Accountability
	Interoperability		Authenticity
Usability		Maintainability	
	Appropriateness recognisability		Modularity
	Learnability		Reusability
	Operability		Analysability
	User error protection		Modifiability
	User interface aesthetics		Testability
	Accessibility		
Portability			
	Adaptability		
	Installability		
	Replaceability		

The ISO 25010 Product Quality Model is expressed according to a framework that defines certain basic concepts. Per this framework Quality Characteristics, Quality Properties, and Quality Measures are disjoint but related concepts. Quality Characteristics are general and not directly measurable attributes of an artefact, such as *Usability* or *Reliability* (Quality Characteristics may also have more specific Sub-Characteristics). Quality Properties are measurable attributes of an artefact that contribute to some Quality Characteristics, and Quality Measures are the results of performing measurement of such properties by some method.

The Product Quality Model [16] is displayed in Table 1. It defines a set of eight quality characteristics, each composed of two to six sub-characteristics. These quality characteristics have definitions written from a system or product perspective. For instance, the quality characteristic *availability* is defined as *degree to which a system, product or component is operational and accessible when required for use.* The quality characteristics have been adapted for an ontology and CODP context and are used as the basis for the set of quality characteristics included in the CODP Quality Model (presented in Section 3.1).

While ISO 25010 also states the existence of measurable quality properties contributing to these quality characteristics, no instances of such quality properties are introduced or formalised.

2.1.2 ER Model Quality

In [6] Genero et al. study a set of metrics believed to affect the learnability and modifiability of ER models. Their study participants were given a set of ER models that differed in the metrics being studied, and were tasked with first filling out a questionnaire to evaluate their understanding of the ER model, and secondly, to modify the ER model in accordance with a newly introduced set of additional requirements. In analysis, the time taken to respond to the questionnaire and to perform the model changes was studied to ascertain the relative understandability and modifiability of the ER models that exhibited different values for the metrics of study. Genero et al. find a significant correlation between a higher number of relations and attributes within the model and an increased understandability and modifiability time, but no significant correlation between the number of entities as a whole and understandability.

Moody and Shanks [22] discuss the issue of redundancy and simplicity, arguing that simpler models have shown to be more flexible, easier to implement, and easier to understand. They suggest that if the size of a model is calculated from the number of entities and relationships in the model, the simplest solution is the one that minimises this size. However, it is important to note that the model must still be suitable for its intended purpose. This mirrors the perspectives brought forward by Lindland et al. in [18]: a high-quality model is constrained by both completeness and relevance criteria, such that it should be large enough to fulfil its purpose, but no larger than that.

In the CODP Quality Model the work presented by Moody and Shanks, and Lindland et al., are captured in quality indicators measuring CODP size and CODP minimalism (in Section 3.2 listed as *MI10* and *MI20*), contributing to the learnability and operability of CODPs. The work presented by Genero et al. (both their findings and method) are also reused – the findings in the form of indicators measuring class to property ratios (*MI07*) contributing to learnability and operability, and the method in the form of in-use indicators of usability and modifiability (*UI01* and *UI02*).

2.1.3 Pattern Quality

Ontology Design Patterns are a sort of design patterns, that is, packaged solutions to commonly occurring problems. In this, they share purpose with software design patterns, the most common of which are design patterns for object oriented programming (OOP). Prechelt et al. [23] report on two experiments in which the characteristics of OOP design pattern code implementations (in particular, the number of pattern comment lines, PCL, associated with each such implementation) affect the maintainability of software products in which the patterns are used. They find that the more well

documented and explicit that design pattern usage in software code is, i.e., the higher the PCL value, the faster and better (in terms of rarity of errors made) maintenance tasks are performed over that software code. This finding is mirrored in the CODP Quality Model via an indicator measuring the annotation coverage of the CODP (*MI01*).

2.2 Ontology Quality

One perspective on CODPs is to consider them as being simple, small, and reusable ontologies. The following section introduces work on ontology evaluation frameworks, methods, and indicators, that have been studied during and in several cases influenced development of the CODP quality model. In addition to the ontology evaluation work introduced below, the interested reader is referred to [7], in which Goméz-Peréz et al. summarise and discuss several types of common taxonomical errors and anti-patterns in ontologies.

2.2.1 O^2 and oQual

A thorough study on the evaluation of of ontologies is performed by Gangemi et al. in [3] and [4]. Their approach is based on two perspectives of ontologies, formalised into two meta-ontologies for understanding, classifying, and selecting ontologies, *O^2* and *oQual*.

The *O^2* meta-ontology views ontologies as semiotic objects, i.e., information objects with intended conceptualisations to be used in particular communication settings. O^2 holds concepts such as *Rational agent*, *Conceptualisation*, and *Graph* – representing the settings in which ontologies are used, the users of said ontologies, their intended meaning of the ontologies, the actual implementation (i.e., graph models), etc. This model also holds the concept *QOOD*, or *Quality Oriented Ontology Description*, which represents roles and tasks associated with processes and elements of the ontology – in essence, a type of requirements specification for part of, or a whole, ontology engineering project.

Based on the concepts in O^2, three dimensions of quality or evaluation are presented and discussed – *structure*, *functionality*, and *usability*. The first kind of measures treat the ontology as a directed graph (which is consistent with the RDF data model) and concern the structures present in this a graph. This includes measures like subsumption hierarchy depth, breadth, fan-out, cycle ratios, density, etc. Several of these measures have been reused as indicators in the CODP Quality Model, including the notions of axiom to class ratio, class disjointness ratio, class to property ratio, and tangledness (in Section 3.2 listed as indicators *MI05*, *MI06*, *MI07*, and *MI23*). The second kind of measures concern the intended functionality of the ontology, and include such examples as precision, coverage, and accuracy, which are all measured against some set of requirements over the

ontology. The third kind of measures concern the communication aspects of the ontology, i.e., how it is documented, annotated, and understood by users. This includes measures related to recognition, efficiency, and interfacing. While the structural measures included in [4] are presented using formal definitions (in many cases even mathematic formulas) the functionality measures are less formally defined (giving some specific indicators, but mostly general method suggestions), and usability measures are even less defined (given almost entirely as examples or areas for future development).

The *oQual* formal model for ontology validation (and its associated meta-ontology) provides a bridge connecting the ontology engineering situation and context (modelled according to O^2) with the aforementioned measures, enabling validation that a certain developed ontology is sufficient and appropriate for the use for which it was developed. oQual includes concepts like value spaces and parameters over ontology elements, ordering functions for selecting parameters from different QOODs to prioritise, trade-offs that may need to be made, etc.

2.2.2 ONTOMETRIC

ONTOMETRIC, introduced in [21], is an approach for formalising ontology suitability for different tasks, heavily influenced by the Analytic Hierarchy Process (AHP) [25], an established method for aiding decision-making when dealing with multi-criteria problems. Per this method, a multi-criterion problem is broken down into the different criteria that need to be met. These criteria are sorted using a comparison matrix, such that the "competing" criteria on each level of the decision hierarchy are easily and intuitively compared pairwise, and the resulting prioritisation used to calculate a weighting of the total set of criteria with respect to the problem. When selecting between available alternatives, the weighting can be used to calculate a total suitability score for each option (i.e., the highest score would be associated with the most suitable ontology for reuse).

In order to support this method, ONTOMETRIC provides a set of criteria for ontology suitability that can be used to populate an AHP decision hierarchy. These general criteria of ontology suitability are divided up into five different dimensions, representing different aspects of an ontology suitability problem: content, language, methodology, tool, and costs. A drawback of using ONTOMETRIC criteria for quality assurance purposes is that the method and model does not in itself suggest which criteria are useful in which situations - it merely provides them as examples of things that can be measured and prioritised by the user. Added to that, the criteria are rather technical and would likely be inaccessible to the non-expert user. However, for the intended usage, ontology selection guidance for ontology engineers), particularly in the early phases of an ontology engineering project, ONTOMETRIC has great potential.

2.2.3 OntoClean

One of the most well-known methodologies for evaluating the conceptual consistency of ontologies is OntoClean by Guarino and Welty [26, 8, 9]. OntoClean uses a logical framework including very general ground notions and definitions from philosophy, believed to hold in any reasonable representation of the world, including an ontology model. The framework includes the notions of *rigidity*, *identity*, and *unity* as characteristics applicable to classes in an ontology, and a set of constraints on which taxonomic relations that may exist between classes that exhibit these different characteristics. The listed characteristics are in OntoClean parlance denoted *metaproperties*.

By annotating the classes in an ontology using the OntoClean metaproperties and checking whether the associated constraints hold, a developer can test whether their ontology is conceptually and philosophically consistent with regards to the notions of rigidity, identity and unity. It should be noted that this is not a guarantee that the ontology is sound with respect to real world phenomena or requirements. The methodology has been applied beneficially in several projects [5, 2].

3 The CODP Quality Model

Based on the related work presented in Section 2 and extending on the work detailed in [12], the author has developed a CODP Quality Model, consisting of three main components:

1. A CODP Quality Metamodel.
2. A set of CODP quality characteristics encapsulating different aspects of quality as relevant to CODP usage.
3. A set of CODP quality indicators that measure to what degree a given CODP expresses the aforementioned quality characteristics.

The quality characteristics and quality indicators are discussed in Sections 3.1 and 3.2 respectively. The notions of quality characteristic and quality indicator are defined in the CODP Quality Metamodel, which provides a conceptual understanding of how to discuss quality as applied to CODPs. The CODP Metamodel is illustrated in Figure 1. The key concepts it encompasses are:

- **Usage Context** – Represents the context in which a CODP is used to construct an ontology. Component concepts of this context are the skills of the ontology engineer(s), the use to which the resulting ontology is intended to be put, and the social or business environment in which the development and subsequent use of the ontology will take

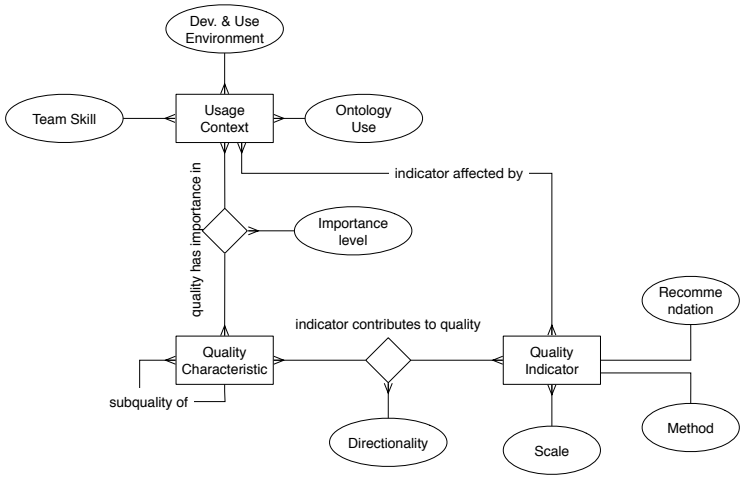

Figure 1: CODP Quality Metamodel

place (the latter including aspects such as development method, team distribution, customer relationship, etc).

- **Quality characteristic** – Denotes a particular aspect of CODP quality that may be of greater or lesser importance in a particular usage context. Quality characteristics can be decomposed into subcharacteristics. Quality characteristics represent concerns or perspectives on quality on an abstract level. They are not themselves directly measurable using some metric or method.

- **Indicator** – Individually measurable properties of an CODP that contribute to increasing or decreasing some quality characteristic(s). Can be accompanied by scales of measure, measurement methods, and recommended values. Can be affected by usage context.

It should be noted that the CODP Quality Model is partially (e.g., [10, 11, 12, 13]) but not exhaustively evaluated. The author invites the Semantic Web research community to further test the model's indicators and their predicted effects.

3.1 Quality Characteristics

The CODP Quality Model quality characteristics (developed from the ISO 25010 Product Quality Model discussed in Section 2.1.1) are presented in Table 2, and discussed in detail below. Table 2 also includes references to the quality indicators that contribute to each quality characteristic.

Table 2: CODP Quality Characteristics. For indicator names, see Table 3

Quality characteristic	**Sub-Characteristic**	**Indicators**
Functional Suitability		
	Functional Completeness	
	Functional Appropriateness	
	Consistency	
	Accuracy	MI12
Usability		
	Appropriateness recognisability	DI01, DI03, DI06, DI07
	Learnability	DI03-DI07, MI01, MI02, MI07, MI10, MI17-MI23, UI01, UI03
	Operability	MI02, MI07, MI10, MI19, MI21-MI23, MI25
	User error protection	DI02
	User interface aesthetics	DI04
	Accessibility	DI01, DI06
Maintainability		
	Modularity	MI17, MI18
	Analysability	MI01, MI05, MI20
	Modifiability	UI02
	Testability	
	Stability	
Compatibility		
	Reusability	MI17, MI18, MI25
	Co-existence	MI10
	Interoperability	MI25
Resulting performance efficiency		MI02-MI04, MI06, MI08, MI09, MI11, MI13-MI19, MI22-MI25

While most of the developed quality characteristics are associated with one or more quality indicators, several are not. This does not imply that those characteristics are unimportant, but rather that more work is required in order to find suitable indicators to measure them.

3.1.1 Functional Suitability

Degree to which a CODP meets stated or implied needs.

- *Functional completeness* – Degree to which the CODP meets expressed knowledge modelling requirements (i.e., competency questions and other design requirements).
- *Functional appropriateness* – Degree to which the CODP facilitates simple storage and retrieval of knowledge formalised according to its definitions (e.g., does the CODP require simple or complex SPARQL queries to retrieve knowledge).
- *Consistency* – Degree to which the CODP is internally logically consistent.
- *Accuracy* – Degree to which the CODP accurately represents the real

world domain being modelled (e.g., whether it adheres to established industry standards and protocols, or legislation).

3.1.2 Usability

Degree to which a CODP can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction.

- *Appropriateness recognisability* – Degree to which users can recognise whether a CODP is appropriate for their needs.
- *Learnability* – Degree to which a CODP's structure, and intended usage can be learned by users new to it, such that they can thereafter apply the CODP successfully and efficiently.
- *Operability* – Degree to which a CODP has attributes that make it easy to apply and use.
- *User error protection* – Degree to which a CODP prevents users from making modelling errors.
- *User interface aesthetics* – Degree to which the CODP's documentation (text, graphics, etc.) is pleasing for the user.
- *Accessibility* – Degree to which the CODP's documentation can be used by people with the widest range of characteristics and capabilities.

3.1.3 Maintainability

Degree of effectiveness and efficiency with which a CODP (and consequently, ontologies built using that CODP) can be adapted and modified by maintainers after deployment in some usage scenario.

- *Modularity* – Degree to which the CODP is composed of discrete components such that a change to one component has minimal impact on other components.
- *Analysability* – Degree of effectiveness and efficiency with which it is possible to assess the impact on a CODP-based ontology module (or the CODP itself) of intended change to one or more of its parts, or to diagnose a CODP for deficiencies or causes of failures, or to identify parts to be modified.
- *Modifiability* – Degree to which a CODP-based ontology module (or the CODP itself) can be effectively and efficiently modified without introducing defects or degrading existing CODP quality.

- *Testability* – Degree of effectiveness and efficiency with which test criteria can be established for a CODP and tests can be performed to determine whether those criteria have been met.
- *Stability* – Perceived change expectation on the CODP - high stability denotes a low degree of change is expected, and vice versa.

3.1.4 Compatibility

Degree to which a CODP can successfully be reused and integrated with other CODPs or IT artefacts in the construction of ontologies or systems.

- *Reusability* – Degree to which a CODP can be used in more than one system, or in building other assets.
- *Co-existence* – Degree to which a CODP can coexist with other CODPs as modules in an ontology, i.e., without detrimental impact on other CODP modules.
- *Interoperability* – Degree to which two or more CODPs share representations of co-occurring concepts.

3.1.5 Resulting performance efficiency

Reasoner or system performance efficiency over ontologies created using the CODP. Can be discussed in terms of the average time taken, or of the system resources required, to materialise inferences of ontologies using this CODP.

3.2 Quality Indicators

Table 3 lists a set of 36 quality indicators that measure different aspects of CODP quality. The indicators are in the table and in the subsequent discussion grouped by the artefact or action that they most directly measure - the CODP documentation, the CODP model, or the CODP when used by human ontology engineers:

- The *documentation quality indicators* all concern the textual and graphical description of a CODP, typically provided in the form of an accompanying web page or other document. Such indicators contribute mostly to usability-related CODP qualities, e.g., appropriateness recognisability, learnability, operability, etc.
- *Model indicators* measure different aspects of the reusable OWL building block that makes up the core of a CODP, or its underlying RDF/RDFS model. These indicators contribute to qualities that are important in terms of how a CODP or an CODP-based ontology is used together with other CODPs, ontologies, or in information systems, including maintainability, compatibility, and reasoning performance.

- *In-use indicators* measure properties of CODPs that cannot be captured using quantitative methods over the CODP itself, but rather requires the use or evaluation of that CODP by a human ontology engineer. These indicators contribute primarily to those qualities that are contextually bounded, e.g., maintainability, usability, and functional suitability.

Table 3: CODP Quality Indicators

Category	**Nr**	**Indicator**
Documentation Indicators		
	DI01	Accompanying Text Description
	DI02	Common Pitfalls Description
	DI03	Competency Question Count
	DI04	Documentation Minimalism
	DI05	Example Illustration Count
	DI06	Example Text Count
	DI07	Structure Illustration
Model Indicators		
	MI01	Annotation Ratio
	MI02	Anonymous Class Count
	MI03	Average Class In-Degree
	MI04	Average Class Out-Degree
	MI05	Axiom/Class Ratio
	MI06	Class Disjointness Ratio
	MI07	Class/Property Ratio
	MI08	Cyclomatic Complexity
	MI09	Existential Quantification Count
	MI10	Minimalism
	MI11	Nary Relation Count
	MI12	OntoClean Adherence
	MI13	OWL Horst Adherence
	MI14	OWL 2 EL Adherence
	MI15	OWL 2 QL Adherence
	MI16	OWL 2 RL Adherence
	MI17	Property Domain Restrictions
	MI18	Property Range Restrictions
	MI19	Redundant Axiom Count
	MI20	Size
	MI21	Subsumption Hierarchy Breadth
	MI22	Subsumption Hierarchy Depth
	MI23	Tangledness
	MI24	Terminological Cycle Count
	MI25	Transitive Import Count
Usage Indicators		
	UI01	Functionality Questionnaire Time
	UI02	Modification Task Time
	UI03	User-Reported Abstraction Level Rating

The indicators, their provenance, and effects, are discussed in the following sections. For reasons of brevity, not all of the indicators in Table 3 are covered. The interested reader can find more details, including suggested measurement methods and scales for several indicators, in [12].

3.3 Documentation Indicators

The first documentation indicator, *DI01 – Accompanying Text Description*, simply measures whether there actually exists a dedicated documentation page or other equivalent documentation for a given CODP module. It may be surprising to the reader that CODPs exist that are not at all documented. This can occur when patterns are extracted from existing ontologies or other data structures in an exploratory manner, as opposed to designed in a structured manner, particularly in project contexts where the resulting CODPs are intended to be used directly and not necessarily published for a wider audience. Examples of such undocumented CODPs[1] include Nary Classification[2] and Time Indexed Membership[3].

Given that CODP documentation exists, a variety of usability aspects stem from the size and complexity of this documentation. It is important to provide the reader with all the information they need to ascertain the utility and intended use of an CODP, while at the same time not overwhelming them with too much or unnecessary information. Lodhi and Ahmed [19] find that several of the documentation fields used in the OntologyDesignPatterns.org portal are by users and maintainers consistently ranked more important to understanding the CODP than others. The quality indicator *DI04 – Documentation Minimalism* captures the need to display all of this core documentation, and no more. Per this indicator, the documentation fields shown should primarily include graphical representation, examples, pattern intent, OWL building block link, OWL example file, competency questions, common pitfalls, and consequences of use – other documentation fields should be hidden unless the user specifically asks for them. Initial evaluation indicates that documentation minimalism indeed correlates with faster understanding of the CODP by non-expert users [12].

Of the above listed description component fields, the author's forthcoming work indicates that the most important ones affecting appropriateness recognisability are, in order, examples of use (measurable by *DI05 – Example Illustration Count* and *DI06 – Example Text Count*), description of intent, competency questions (measurable by *DI03 – Competency Question Count*), and graphical representation (measurable by *DI07 – Structure Illustration*). The component field "description of intent" is not associated with an indicator of its own due to the difficulty in quantifying such an indicator, and due to the overlap with the aforementioned indicator *DI04* which anyway requires it.

In addition to the count of illustrations of structure (*DI07*) and example uses (*DI05*), it is reasonable to assume that the format and design of those illustrations also have usability-related effects. Some results of the

[1] At the time of writing, 2016-03-29

[2] `http://www.ontologydesignpatterns.org/cp/owl/naryclassification.owl`

[3] `http://ontologydesignpatterns.org/cp/owl/timeindexedmembership.owl`

authors ongoing work, include that users report preferring graph-oriented representations over tree-oriented representations or Venn diagrams. When presented with several common graphical OWL representations, the VOWL format[4] [20] was consistently the most preferred.

3.4 Model Indicators

CODPs are intended to be reusable, from which follows that they should be able to represent or solve the modelling issue for which they are intended to be used, but make few (ideally no) ontological commitments not pertaining to this issue – i.e., they should contain no extraneous axioms not required to fulfil their design requirements. The measure of such CODP minimalism is captured in the indicator *MI10 – Minimalism*, which contributes to increased learnability and operability of a CODP (this effect is also apparent in other models than CODPs, see the discussion in Section 2.1). Minimalism also contributes to the co-existence quality characteristic, i.e., the ability of the CODP to co-exist with other CODPs in an ontology without detrimental effects.

Since the OWL imports feature is transitive and complete, and since CODPs often build upon and specialise one another, relatively few CODPs are actually minimal. It is generally the case that several foundational concepts and properties, that are not strictly necessary to the modelling scenario at hand, are included in an CODP through imports. The number of ontologies or CODPs in the transitive import closure of an CODP is measured via the indicator *MI25 – Transitive Import Count*, which contributes negatively to operability – since more work steps are needed to reconcile these foundational and many times not strictly needed concepts with a model under development. The same indicator also affects interoperability, increasing it for those pairs of CODPs that share foundational concepts, but decreasing it for those that do not.

A CODP that expresses the minimalism property may still be difficult to understand and operate due to its size (i.e., even though the CODP does not exceed its design requirements, those design requirements are so large as to be unintuitive to fulfil in one single CODP). CODP size can be measured in a variety of ways, by number of axioms, number of classes, number of RDF triples, number of bytes in some given serialisation format, etc. The indicator *MI20 – Size* does not specify which measure must be used, but recommends that OWL-level constructs be used as its basis, since that is the level that ontology development tooling tends to operate on, and which users see. The author's prior work [10] indicates that an appropriate size for users to understand fully within a minute or two of study is three to four classes, and the properties and restrictions associated with them. If CODPs

[4] `http://vowl.visualdataweb.org/`

are significantly larger than this they could be considered candidates for splitting apart into sub-patterns, particularly if intended to be used by non-expert ontology engineers.

The structure of a CODP's class subsumption hierarchy can have significant effect on the subsumption hierarchy of an ontology constructed using that CODP. Indicators that measure these structures include *MI21 – Subsumption Hierarchy Breadth*, *MI22 – Subsumption Hierarchy Depth*, and *MI23 – Tangledness*. The former two measure the average number of siblings on a given subsumption level of the CODP, and the average depth in number of levels of subsumption respectively. The latter measures the number of multi-parent classes. Higher values for either of these indicators affect learnability and operability of an CODP negatively [3]. The latter two have also been shown to contribute negatively to reasoning performance [17]. It should be noted that while the mentioned effects on usability-related qualities are derived from the asserted subsumption hierarchy only, the reasoning effects are of course also derived from the inferred subsumption hierarchy, including anonymous classes.

There are several quality indicators that measure properties of the underlying RDF representation upon which an OWL CODP is built. These indicators include *MI03 – Average Class In-Degree* and *MI04 – Average Class Out-Degree*, which measure the average number of in-going and out-going edges from the set of nodes in the RDF graph that are OWL classes (or in other words, the average number of RDF triples involving OWL classes where those OWL classes are in the subject or object positions). They also include *MI08 – Cyclomatic Complexity*, which measures the number of linearly independent paths through in RDF representation of the CODP. All three of these indicators contribute negatively to reasoning performance efficiency [17].

As shown in [13], object properties in CODPs can be specialised by two different methods, the choice of which affects both the reasoning characteristics and the usability of the resulting ontology or CODP. Either new sub-properties are constructed, with defined domains and ranges corresponding to some specialised classes, or existing properties of the CODP are reused and locally constrained by way of property restrictions on those specialised classes. The former method is preferable in terms of reasoning performance of the resulting ontologies/CODPs, whereas the latter method is preferable for data integration purposes. The indicators *MI17 – Property Domain Restrictions*, *MI18 – Property Range Restrictions*, and *MI09 – Existential Quantification Count* all measure how property usage is constrained in the CODP. Constraining property use in the resulting model, whether by use of domain and range restrictions or anonymous class restrictions (measurable by indicator *MI02 – Anonymous Class Count*), has been shown to improve learnability of the resulting CODP, by providing users with clarification on the intended use of the properties in question [12]. However, the flip side of

this is that such constraints limit the reusability of the properties in question, and also increase interdependency between the properties and class definitions that they are constrained by, contributing negatively to CODP modularity.

The indicators *MI05 – Axiom/Class Ratio* and *MI07 – Class/Property Ratio* reflect different aspects of how classes are expressed in the CODP. The former indicator captures a perspective on the logic complexity of an CODP, i.e., whether it is a simple schema with few details described per class, or whether concepts are formalised in a logically more expressive or richer manner. A higher value is preferable in terms of analysability of classes [3]. The latter indicator measures the ratio of classes to properties, i.e., possible interconnections between class members. As discussed in Section 2.1, ER models displaying a higher number of relations and attributes per entity are more difficult to understand. The same is true for CODPs; a higher number of properties per class contributes to decreased learnability and operability while the opposite (i.e., a higher value for *MI07*) increases these qualities [3, 12].

Finally, *MI12 – OntoClean Adherence* measures to which degree the CODP is consistent with a the OntoClean [9] taxonomic constraints, i.e., whether it is logically sound and consistent vis-a-vis the real world. This indicator contributes to the accuracy quality characteristic. As OntoClean can be time consuming to apply over a larger model, the indicator can be expressed either in terms of complete adherence (is the entire model and all taxonomic relations in it fully consistent according to OntoClean), or in terms of the ratio of inconsistent to consistent taxonomic relations. Most CODPs are small enough that the former measure is feasible to use in terms of size - however, CODPs are often also relatively abstract in nature, which can complicate the tagging of their concepts with OntoClean metaproperties, which is made simpler if those concepts are more tangible.

3.5 In-Use Indicators

Indicators *UI01 – Functionality Questionnaire Time* and *UI02 – Modification Task Time* measure the time needed for representative users to perform tasks that require correctly understanding and using a CODP (inspired by the work of Genero et al. [6] discussed in Section 2.1).

In the former case, users respond to a survey on the functionality and intended use of the CODP – for instance, the users might be asked which out of a number of listed competency questions the CODP would be capable of answering, or which class in the CODP that corresponds to a certain term or concept in a textual description. In the case that the same users take multiple surveys for different CODPs, it is important to ensure sufficient randomisation in survey ordering to avoid learning effects affecting the results.

In the latter case, users are tasked with performing a set of modification tasks for a CODP, that require not only understanding the intended CODP usage and structure, but also putting that understanding into practice. Again, in the case that the same users are used to evaluate multiple CODPs, some randomisation of order would be required.

In either case, only the time taken by users who provide correct responses and modification results should be taken into account. The greater the time required, the lower the learnability or operability of the CODP in question. It is however important to note that these indicators are very much dependent on the ontology engineering expertise, or lack thereof, of the participating users – so while they can be useful in evaluating candidate CODPs for a particular project (where the participating rating users are representative of project members), they should not be used to compare CODPs in general.

UI03 – User-Reported Abstraction Level Rating attempts to capture the notion of how abstract or concrete a CODP is. A more abstract CODP (as reported by users) has been shown to be more difficult to understand [12]. Unlike the previous two indicators, abstraction level is difficult to measure in a quantitative manner – we need to rely on user ratings. A survey format using a five-point Likert scale ranging from "Very concrete" to "Very abstract" has worked well for the author in the past. The users doing the rating need to have had sufficient time to study and apply several CODPs in test scenarios before rating them. Providing the users with multiple patterns allows them to see a variation of patterns that makes it easier for them to answer confidently (especially if they lack prior experience of CODPs). As a benchmark of rating reliability, we recommend including some CODPs that have previously been rated a sufficiently large number of times that skew to either end of the spectrum can be identified and adjusted.

4 Discussion and Open Questions

In developing the CODP Quality Model the author has made several observations that warrant further discussion. This section provides input to such discussion by suggesting areas for further work in CODP quality, exemplifying tensions and tradeoffs for developers to keep in mind, and recommending CODP tooling improvements taking the presented qualities and indicators into account.

4.1 Where are the Gaps?

The CODP Quality Model contains several quality indicators contributing to *Usability* and *Resulting Performance Efficiency*, but fewer contributing to *Functional Suitability*, *Maintainability*, and *Compatibility*. This lack is

likely a consequence of the difficulty in studying the latter quality characteristics. Understanding *Maintainability* requires studying a real ontology engineering project as it progresses beyond the initial development phases and into deployment and maintenance phases. By contrast, most research projects are formulated to focus primarily on the development phase, and consequently we do not have enough studies on deployment and maintenance yet. *Functional Suitability* is difficult to study rigorously as it is a very context-bounded quality characteristic, making it quite difficult to generalise any findings. Studying CODP *Compatibility* requires that a sufficient number of high quality CODPs from different sources or designed in different styles already exist, that compatibility can be gauged against. This has until recently not been the case.

The lack of indicators for the above mentioned quality characteristics is particularly unfortunate as these characteristics have several sub-qualities that are important in enabling CODP adoption outside of academia. *Maintainability* is perhaps the most obvious candidate for improvement – without understanding how CODP and XD use in ontology engineering affects the need for maintenance and support of developed ontologies, we will not see industry deploying these methods to any greater extent. In particular CODP *Testability* and *Analysability* are critical qualities to understand when developing CODP-based ontologies that will need to be maintained (possibly by another team than the initial developers) in the future. *Testability* is also highly relevant in supporting the XD method, which depends on tests to ensure that component modules of a ontology project do not break during integration and refactoring [24]. In addition to understanding these quality characteristics, tooling to aid in constructing and executing tests, or in analysing CODP-based ontologies, will also need be developed.

4.2 Tradeoffs to be Made

As the attentive reader is sure to have noticed, there are indicators that contribute negatively to some quality characteristics while contributing positively to others. The presence of such tensions imply that ontology engineers developing or using CODPs may need to make tradeoffs between which qualities they consider most important in their project. Organised by quality characteristic, the tradeoffs found so far (others may surface as the above mentioned gaps are filled) include *Resulting Performance Efficiency* versus *Functional Suitability*, *Resulting Performance Efficiency* versus *Learnability*, *Interoperability* versus *Operability*, and *Learnability* versus *Reusability*.

The tradeoff between *Resulting Performance Efficiency* and *Functional Suitability* stems from the fact that employing the full logic expressivity of the OWL language is computationally expensive, hence the development of reasoning-friendly OWL subsets and the OWL 2 profiles (captured in indicators MI13-MI16). The most reasoning-efficient OWL model imaginable

would be a null model that makes no assertions whatsoever. As a model comes closer and closer to representing some real world domain (i.e., the model's functional completeness increases), the number of axioms (and most likely also the variance of OWL constructs used) in it also increases, which in turn decreases reasoning performance over the model. One concrete example of this is the use of existential quantification axioms (indicator MI09) which may well be required to accurately model the domain, but which are associated with poor reasoning performance [17]. Another example is the use of property domain or range restriction axioms (MI17 and MI18), which on RDF-level increase the number of ingoing edges to each class definition, which has also been shown to be associated with decreased reasoning performance [17].

When applying CODPs in scenarios where the resulting ontology will be used for reasoning*Reasoning* and where the time or resource consumption characteristics of that reasoning matters, developers are recommended to consider carefully whether the CODPs they use are compliant with their reasoning requirements. The simplest way of going about this is to ensure that the CODPs used comply to a suitable OWL 2 profile. However, it is important to also ensure that in the process of specialising and applying those CODPs, no OWL constructs not supported by the chosen profile are introduced. Thankfully the two common ontology development environments Protégé and WebProtégé both include features to easily check the profile adherence of an ontology under development, simplifying this work. If a task-suitable CODP exists but does not adhere to a required reasoning profile, the author encourages the developer who notices this to reimplement and release that CODP as a new profile-compliant version.

The tradeoff between *Reasoning Performance Efficiency* and *Learnability* is caused by the same underlying mechanism – several OWL constructs that increase learnability of the model are associated with poor reasoning characteristics. This includes the aforementioned use of domain and range restrictions, which typically help users understand the intended use of CODP properties. It also includes the use of class restriction axioms that can similarly help illustrate how classes and properties in a CODP are to be reused, but which in several cases decrease reasoning performance [17]. When constructing the CODPs it may be beneficial to consider increasing the scope of the CODP metadata or documentation to better describe the CODP's features and their intended use, rather than relying on the above discussed performance-affecting constructs for this purpose (unless required to realise some actual functionality of the CODP).

The tradeoff between *Interoperability* versus *Operability* relates to the transitive import closure indicator (*MI25*). For a CODP to be as interoperable as possible, it should be aligned to and share foundational concepts with as many well-established CODPs as possible – which would be indicated by MI25 having a high value. However, in order for a CODP to be

as easy to use as possible, it should not require the developer to have to understand a large number of phenomena that are outside of scope of the problem the CODP intends to solve, i.e., the MI25 value should be low.

Finally, the tradeoff between *Learnability* versus *Reusability* again concerns indicators MI17 (property domain restrictions) and MI18 (property range restrictions). As discussed above, the use of property and range restrictions improve learnability, so a CODP that has higher values for these indicators would be easier to understand and learn than one that has lower ones. However, in order for a CODP to be as reusable as possible it should make as small an ontological commitment as possible while still fulfilling its design goals. Consequently for reusability MI17 and MI18 should measure as low as possible, as the properties defined in the CODP should not make assumptions about what classes they are used together with, but rather be reusable outside of the exact CODP scope if need be. This is particularly important as the semantics of OWL property domain and range definitions imply that if a property has multiple asserted ranges or domains, the resulting inferred domain or range of that property is the intersection of the set of domain or range class expressions. In other words, it is not possible for a child CODP to extend a property domain or range from an imported parent CODP, only to constrain it further.

4.3 Improving ODP Tooling

As mentioned in Section 1, in order to increase CODP adoption, the research community needs to improve the quality of both CODPs themselves and CODP support tooling and methods. The presented quality indicators provide guidance on what types of features such improved CODP support tooling (including both client-side XD support tooling and online CODP repositories) should include.

The largest and most well-known CODP repository on the web is that hosted by OntologyDesignPatterns.org. The presentation in this repository would benefit from reducing the number of extraneous documentation fields displayed by default for each listed CODP, in order to comply better with indicator *DI04 – Documentation Minimalism* (see Section 3.2 for details). In addition, example uses of each CODP should be described, ideally both in text format and with accompanying graphical illustrations (as per indicators *DI05 – Example Illustration Count* and *DI06 – Example Text Count*). While the portal has support for describing usage scenarios, rather few CODPs do so at the time of writing.

There is also room for improvement with regard to more easily finding CODPs compliant with project requirements (particularly in light of the above discussed tradeoffs). Whether users browse the CODP repository manually or use a search engine, they would benefit from being able to constrain or filter the resulting CODP list using the presented indicators (par-

ticularly those indicators contributing to degraded reasoning performance, decreased interoperability, or decreased reusability). Related to this, a useful addition to the repository would be if it were to support the notion of different versions or flavours of CODPs, i.e., different solutions or reusable modules to the same problem that vary across these indicators. For instance, there might be a full OWL 2 CODP representing event participation, and a reduced OWL 2 EL version of that same CODP, sacrificing some logic expressiveness for increased reasoning performance.

In addition to the above improvements, the author's ongoing work in several projects indicates that repository usability would be improved by increases in the documentation coverage, by a clearer overview of CODP interdependence, and by harmonisation of the graphical CODP representations used. Thankfully the Association for Ontology Design & Patterns (ODPA), stewards of the portal in question, have recently begun work aiming to improve the quality of the repository, both in terms of features and in terms of content.

Client-side XD support tools and ontology IDE:s would, as touched upon in Section 4.1, benefit from better support for constructing and evaluating tests over CODP-based ontologies. For full XD support such test tooling would need be able to evaluate whether Competency Questions[5] and Reasoning Requirements of an ontology are fulfilled (for instance by evaluating SPARQL query result over asserted and inferred data respectively), and whether Contextual Statements still hold (possibly using a rules engine, as testing based on open world reasoning would not trigger an error if data expected to be in the model was lacking).

Additionally, such tools would benefit from functions that simplify dealing with CODPs that have large import closures (*MI25 – Transitive Import Count*). For instance, a tool might provide the user with a guided interface that partially clones a reused CODP, keeping only those parts that are required in order to successfully model the solution to the problem at hand, while removing extraneous higher-level classes and properties. Promising work in this direction, the MIREOT guidelines and tooling, has been carried out by Courtot et al.[1] and Hanna et al. [15]. Integrating their work in a CODP and XD workflow has the potential to bring great benefit to CODP users and developers.

5 Summary

In this chapter we have presented a CODP Quality Model consisting of a metamodel providing a conceptual understanding and language by which CODP quality can be discussed. We have also presented a set of quality

[5]For an introduction to CODP requirements, including the types mentioned here, see [24]

characteristics to consider when using CODPs to develop ontologies, and an initial set of quality indicators by which these quality characteristics can be measured. Some of the tradeoffs that need to be made when prioritising among the presented quality characteristics have been introduced and discussed, and directions for further development (of research and tooling) proposed.

References

[1] M. Courtot, F. Gibson, A. L. Lister, J. Malone, D. Schober, R. R. Brinkman, and A. Ruttenberg. Mireot: The minimum information to reference an external ontology term. *Applied Ontology*, 6(1):23–33, 2011.

[2] M. Doerr, J. Hunter, and C. Lagoze. Towards a Core Ontology for Information Integration. *Journal of Digital Information*, 4(1), 2006.

[3] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling Ontology Evaluation and Validation. In *The Semantic Web: Research and Applications*, pages 140–154. Springer, 2006.

[4] A. Gangemi, C. Catenacci, M. Ciaramita, J. Lehmann, R. Gil, F. Bolici, and O. Strignano. Ontology evaluation and validation. Technical report, Laboratory for Applied Ontology, ISTC-CNR, 2005.

[5] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening WordNet with DOLCE. *AI magazine*, 24(3):13, 2003.

[6] M. Genero, G. Poels, and M. Piattini. Defining and Validating Measures for Conceptual Data Model Quality. In *Advanced Information Systems Engineering*, pages 724–727. Springer, 2006.

[7] A. Gómez-Pérez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Springer-Verlag, London, Berlin, 2004.

[8] N. Guarino and C. Welty. Evaluating Ontological Decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2002.

[9] N. Guarino and C. A. Welty. An Overview of OntoClean. In *Handbook on Ontologies*, pages 201–220. Springer, 2009.

[10] K. Hammar. Ontology design patterns in use: lessons learnt from an ontology engineering case. In *Workshop on Ontology Patterns in conjunction with the 11th International Semantic Web Conference 2012 (ISWC 2012)*, 2012.

[11] K. Hammar. Reasoning performance indicators for ontology design patterns. In *Proceedings of the 4th International Conference on Ontology and Semantic Web Patterns-Volume 1188*, pages 27–38. CEUR-WS. org, 2013.

[12] K. Hammar. *Towards an Ontology Design Pattern Quality Model*. Linköping University Electronic Press, 2013.

[13] K. Hammar. Ontology design pattern property specialisation strategies. In *Knowledge Engineering and Knowledge Management*, pages 165–180. Springer, 2014.

[14] K. Hammar and K. Sandkuhl. The State of Ontology Pattern Research: A Systematic Review of ISWC, ESWC and ASWC 2005–2009. In *Workshop on Ontology Patterns: Papers and Patterns from the ISWC workshop*, pages 5–17, 2010.

[15] J. Hanna, C. Chen, W. A. Crow, R. Hall, J. Liu, T. Pendurthi, T. Schmidt, S. F. Jennings, M. Brochhausen, and W. Hogan. Simplifying mireot: a mireot protégé plugin. In *11th International Semantic Web Conference ISWC 2012*, page 25. Citeseer, 2012.

[16] ISO. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Technical report, International Organization for Standardization, 2011.

[17] Y.-B. Kang, Y.-F. Li, and S. Krishnaswamy. Predicting Reasoning Performance Using Ontology Metrics. In *The Semantic Web – ISWC 2012*, pages 198–214, 2012.

[18] O. I. Lindland, G. Sindre, and A. Solvberg. Understanding Quality in Conceptual Modeling. *Software, IEEE*, 11(2):42–49, 1994.

[19] S. Lodhi and Z. Ahmed. Content Ontology Design Pattern Presentation. Master's thesis, Jönköping University, 2011.

[20] S. Lohmann, S. Negru, F. Haag, and T. Ertl. VOWL 2: User-oriented visualization of ontologies. In *Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW '14)*, volume 8876 of *LNAI*, pages 266–281. Springer, 2014.

[21] A. Lozano-Tello and A. Gómez-Pérez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 2(15):1–18, 2004.

[22] D. L. Moody and G. G. Shanks. What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models. In

Entity-Relationship Approach – ER '94 Business Modelling and Re-Engineering, pages 94–111. Springer, 1994.

[23] L. Prechelt, B. Unger-Lamprecht, M. Philippsen, and W. F. Tichy. Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance. *Software Engineering, IEEE Transactions on*, 28(6):595–606, 2002.

[24] V. Presutti, E. Blomqvist, E. Daga, and A. Gangemi. Pattern-based ontology design. In *Ontology Engineering in a Networked World*, pages 35–64. Springer, 2012.

[25] T. L. Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15(3):234–281, 1977.

[26] C. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering*, 39(1):51–74, 2001.