Identifying ontology design styles with metrics

Eleni Mikroyannidi, Robert Stevens and Alan Rector

The University of Manchester
Oxford Road, Manchester, M13 9PL
{mikroyannidi,stevens,rector@cs.manchester.ac.uk}

Abstract. This paper presents a metrics framework for identifying high-level ontology composition styles as an abstraction layer for ontology comprehension. The metrics are implemented on top of the OWL API metrics, which have been extended to capture various aspects of an ontology. The metrics are applied to an ontology repository, and using clustering methods underlying design styles have been revealed. We seek to give a high-level view of how an ontology has been "put together" through the identification of authoring style using a range of ontology metrics. This 'ontological style' can be considered as a metamodel, giving an intuition of what lies "under the hood" of an ontology. This can be useful in the broader area of ontology comprehension. The results highlighted five design styles covered by the repository describing simple, as well as rich and complex ontologies.

1 Introduction

We present the use of data-mining across a broad range of ontology metrics to describe the *ontological design style* of how a particular ontology has been authored in terms of its size, shape, number and disposition of axioms. Our aim is to use this guide to ontological style as part of a framework to facilitate *ontology comprehension*. OWL ontologies are often considered to be complex entities that require an excessive amount of time and effort to be used in conjunction with other ontologies or to be integrated into an application [16]. One important task in authoring, using and re-using an ontology is its comprehension [6,12]; that is, understanding the ontology at a range of levels.

The process of understanding ontologies can be compared to the process of understanding code, its structure, organisation etc in software engineering. The OWL language can be compared with programming languages, as it represents the ontology as a series of statements or axioms. To comprehend an ontology, a reader has to take these statements and synthesise an understanding of how the ontology has been 'put together' and how it works; without such a comprehension, extension and maintenance are difficult. On a second level, the user has to identify those parts of the ontology that have been modelled with the same combination or pattern of constructors and entities, together with their interconnection with the other parts of the ontology. This already difficult task is made harder still as ontologies that are modelled with logic axioms can be

difficult to understand [13]. In addition, ontologies that make use of advanced characteristics of OWL can lead to inferences that add complexity in their structure. As a consequence, a user can be faced with a logical artefact that is difficult to comprehend at many levels: The style of the modelling; the size and shape of the whole ontology; the patterns of axioms used; the entailments afforded by the ontology; and so on.

Metamodels can be helpful towards the comprehension of an ontology, as they provide abstraction layers that hide the complexity of the ontology [5]. One such high-level abstraction is the *ontological design style* employed in a given ontology. By ontological design style we mean, very informally, how an ontology's axioms have been "put together". More precisely, ontology style is the summary of the main characteristics of composition and structure of an ontology. The design style can be assessed by inspecting the ontology,; however, metrics are helpful for giving an overview of an ontology's design style. The workflow for the generation of the ontology design styles is presented in Figure 1. In this paper, we use a range of metrics measuring various aspects of an ontology, such as the size, the complexity of the structures in the ontology, the axiom types etc. to provide a series of dimensions that characterise the design style. These dimensions constitute a rating scheme for describing the design style from a high level of abstraction. We then use clustering methods to group ontologies in a repository according to these various dimensions. Intuitively, we expect to divide the repository into groups such as small and complex, large and simple etc. Indeed, the results showed five main design styles covered by the repository, describing ontologies, which for example are small but rich and complex; ontologies in which most information is expressed in the taxonomy; ontologies in which information is expressed with rich object property hierarchies and class expressions; etc. These proposed design styles should help to reveal the main characteristics of the ontologies.

We believe that the detection and description of design styles can be helpful for comprehending an ontology, as it can give an intuition of its composition and an initial guide on how to inspect the ontology. For example, if there is a complex object property hierarchy and if all the entailments refer to property assertions, the user can start by inspecting these elements in the ontology. In general, these design styles could be used as the initial information that guides the choice of the comprehension tactic for an ontology.

2 Related work

Existing work on metrics is related to the assessment of the ontology quality in the scope of ontology integration and ontology modularisation [11]. Recent work shows that metrics are used more for ontology analysis than for ontology comprehension by ontology authors or other users.

OntoQA [15] is a metrics framework for assessing the quality in terms of reusing an ontology. Tartir *et al.* [15] define metrics for the schema inheritance tree and its instances to evaluate the success of it in terms of modelling a real-

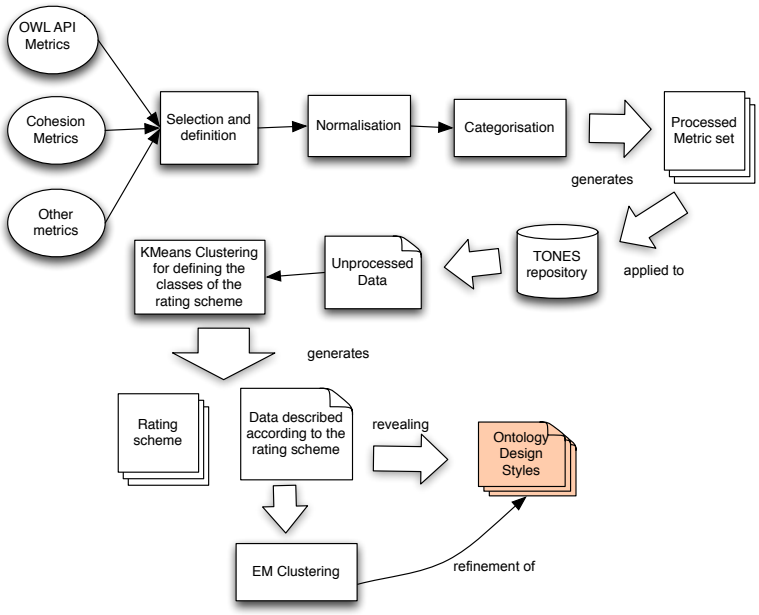


Fig. 1. Workflow for generating the ontology design styles

world domain. Although the described metrics are indications of the schema and the population of the ontology, they do not explicitly refer to the asserted or inferred class graph. both the asserted and inferred structure of an ontology need to be understood by an ontology's users. In addition, the schema metrics focus only on the class hierarchy. Object and data property hierarchy structures can also exist in the ontology and should also be considered, as they can cause additional implications upon use of an automated reasoner.

Yao *et al.* [18] describe cohesion metrics to measure the modular relatedness of OWL ontologies. The cohesion refers to the degree to which the elements in a module belong together. Strong cohesion is recognised as a desirable property of object-oriented classes, as it measures separation of responsibilities, independence of components and control of complexity. The authors focus only on the class hierarchy of the ontology, without explicitly defining the constraints that should be considered for the computation of the metrics. Alternative representations can derive from the same subsumptions that could alter the measures. Such an example is shown in Figure 2. The subsumption axioms $\mathsf{D} \sqsubseteq \mathsf{C}$, $\mathsf{C} \sqsubseteq \mathsf{B}$, $\mathsf{B} \sqsubseteq \mathsf{A}$ can be represented differently depending on the consideration of direct and indirect inheritance. Therefore, the consideration of additional parameters are required to ensure the stability of the metrics [17].

In another direction is the use of statistical analysis to assess the design of an ontology from a high level. Ontology editors such as Protégé-4 and Swoop

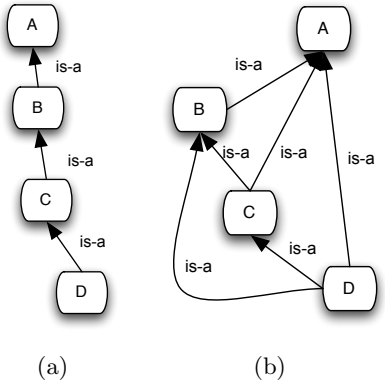


Fig. 2. Different representation of the hierarchy derived from the same subsumption axioms.

provide simple metrics for an ontology. Protégé-4 has an introductory view pane that presents metrics that can give an indication of the size and some of the characteristics of the ontology, while Swoop provides some more details on the taxonomy and property hierarchy of the loaded ontology. Even though there are helpful tools that provide general statistics for ontologies, there is a lack of systematic and comprehensive metrics for characterising ontologies in terms of design. These could be further used as a metamodel that presents the main characteristics of the ontology. Work close to metrics and metamodels is presented in [5,4]. However, the structural metrics defined by the authors have the same instability in terms of their computation as the aforementioned frameworks.

Efforts for describing ontologies and their complexity using metrics are reported in [9,10]. In [9] the authors use the OWL API metrics and additional ones on the entailments of the ontologies to describe a biomedical repository. The work presented here is also based on the OWL API metrics. However, we use additional machine learning techniques to pinpoint underlying design styles of an ontology; it is this ability to give a high-level 'overview' of what kind of ontology style has been used in an ontology that is the contribution of this paper.

3 Ontology metrics framework

The main question for analysing ontologies with metrics for 'design style' is which metrics are appropriate to characterise an ontology and give an indication of the design style and complexity of the ontology? This cannot be answered easily, as there can be multiple metrics that can describe a particular aspect of an ontology, such as the taxonomy, and all of them might be equally informative as to design style.

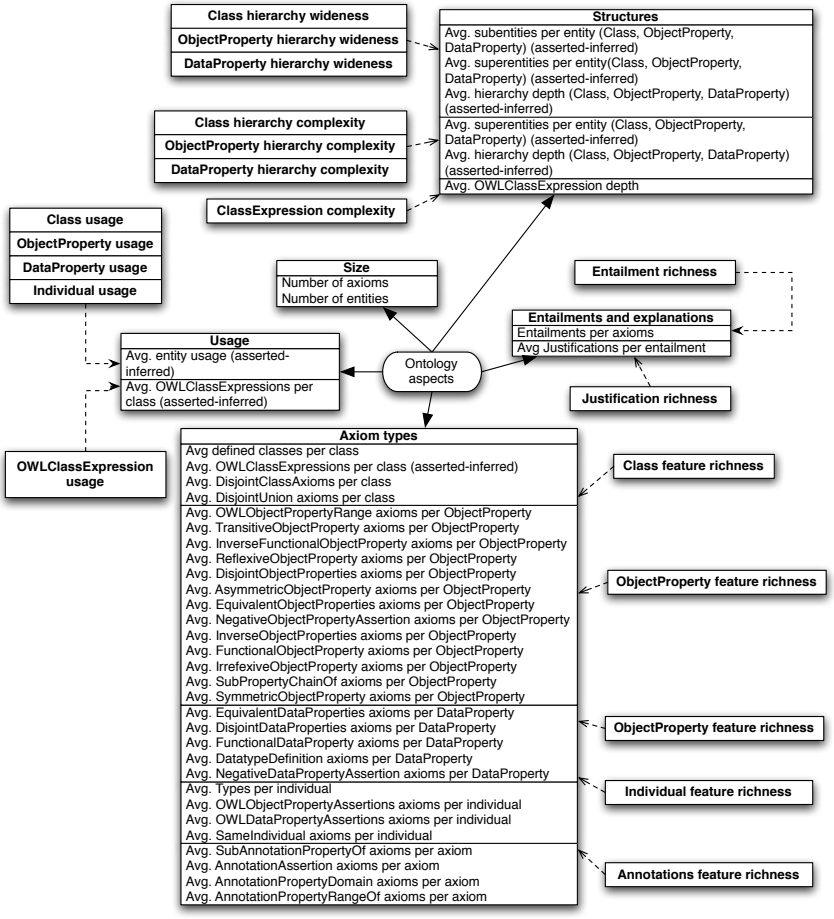


Fig. 3. The metrics framework: Ontology aspects, metrics and dimensions.

The full list of metrics and their descriptions covered by this framework can be found online[1]. In Figure 3, the core of the metrics framework is presented. The figure shows five main aspects of an ontology that are considered by this framework. These are:

1. Size
2. Axiom types
3. Structures
4. Entity usage
5. Entailments and explanations.

[1] `http://www2.cs.man.ac.uk/~mikroyae/2011/metrics/`

As can be seen in Figure 3, these aspects are further analysed into 20 dimensions and each one is represented by a set of metrics (the dimensions are mapped with dashed arrows to the corresponding set of metrics). The size aspect is described by two metrics for characterising the ontology in terms of size. The Axiom type aspect is represented by five dimensions and sets of metrics, which try to assess the variety and type of OWL constructs and features that are used to develop the ontology. The structures we consider in this framework are the class, object and data property hierarchies and the nesting of class expressions. A corresponding set of metrics is used to assess the complexity of these structures. The entity usage metrics give a higher granularity of information related to the axiom type metrics, as they show how many entities of the ontology are used over and above their declaration. Finally, a set of metrics is related to the implications of the ontology. Theoretically, the inferences in an ontology are infinitely many. However, from a practical point of view, an ontology editor shows a finite number of entailments. In the proposed metric framework, we extract a finite subset of and provide metrics for capturing the underlined semantics.

3.1 OWL API metrics

This work uses as a basis the metrics that have been implemented in the OWL API. All the additional metrics have been developed using the OWL API and have been implemented on top of the existing metrics package, by using and extending the same interfaces. The OWL API has already a set of 46 implemented metrics. These include all the axiom types in the ontology, the expressivity and import closure. We selected a subset of 28 metrics, and some of these were modified and computed for the inferred ontology, in order to take into account the implications of the ontology. These mainly refer to axiom types and are shown in Figure 3.

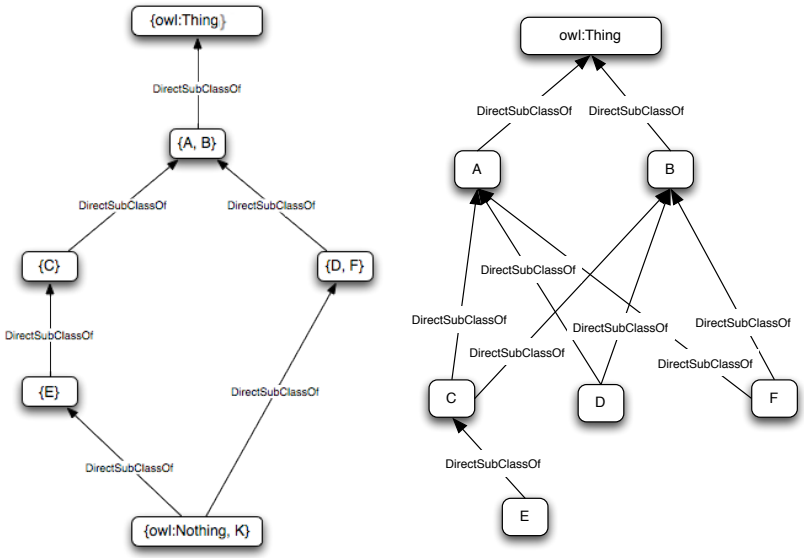
3.2 Structural metrics

The purpose of the metrics related with the hierarchies (class, object property, data property) is to give an intuition of their "shape', meaning its wideness, nesting and complexity. A nested class hierarchy might introduce a complexity in the ontology, but it also represents the level of organisations of the entities. This set of metrics is computed for both the asserted and inferred hierarchies. Changes between the asserted and inferred values reflect the changes in the hierarchy before and after the classification of the ontology. The type of entailments that are taken into account are direct atomic subsumptions.

The interface of the reasoner in the OWL API handles hierarchies similarly to the model presented in [1]. We compute all the structural metrics using the reasoner. For the asserted ontology the OWL API provides the `StructuralReasoner`, which works with the 'told' information only. The reasoner interface of the OWL API handles hierarchies as directed acyclic graphs (DAG), containing nodes connected via edges. Each node in the hierarchy represents a set of entities that are equivalent to each other. The hierarchy has the top node $\top$ (in OWL is owl:Thing,

owl:TopObjectProperty, owl:TopDataProperty) and the bottom node $\perp$ (in OWL is owl:Nothing, owl:BottomObjectProperty, owl:BottomDataProperty). Figure 4(a) shows an example of a class hierarchy; each box in the hierarchy represents a Node. Class K is unsatisfiable, so it is equivalent to owl:Nothing, and therefore appears in the bottom node. In addition, with this approach any cycles are excluded by the reasoner, as this would lead to an infinite number of inheritance levels in the ontology. For the computation of the structural metrics, a number of adjustments have been made to the representation of Figure 4(b). These are:

- Each node in the hierarchy is a single entity. Therefore, equivalent entities are represented as sibling nodes. Such a representation is also adopted by ontology editors. We adopt this kind of representation for having a standard way for computing metrics such as the wideness of the hierarchy.
- A Leaf node is considered as a node without any direct sub nodes, minus the bottom node. Therefore, for the computation of the metrics, the bottom node is excluded as it contains all the unsatisfiable classes, which are captured by a separate metric.



(a) Example class hierarchy as it is handled by the reasoner in the OWL API.

(b) Example class hierarchy as it is represented by the structural metrics.

Fig. 4. Representation of the hierarchy by the reasoner and the structural metrics.

Figure 4(b) shows how the hierarchy of Figure 4(a) is formulated according to the previous adjustments. It should be also mentioned that for the computation

of the structural metrics referring to superclasses and subclasses, only direct superclasses and subclasses are taken into account.

Finally, for assessing the complexity of OWLClassExpressions, we use the average OWLClassExpression nesting.

3.3 Entity usage metrics

This set of metrics gives an insight into the interconnectivity between entities in the ontology. For the computation of these metrics, we exclude usage that refers to declaration and subsumption axioms. The subsumption axioms are excluded as they are assessed with the structural metrics. We consider and measure the usage for the classes, object properties, data properties, individuals and class expressions in the ontology. The OWLClassExpressions usage metric tries to assess the number of universal and existential restrictions that enrich the description of classes. The inferred metric is computed by extracting for each class structural information from the class hierarchy. In the OWL API, the reasoner returns entailments referring to atomic subsumptions. Based on this information and on the inheritance that derives from the class hierarchy, we compute the usage of OWLClassExpressions.

3.4 Entailment metrics

There are different metrics that are used to assess the inferences in the ontology. A set of metrics computes the entailments referring to relations between atoms. Authors in [2] describe parameters that need to be considered when extracting a finite set of entailments from an ontology. Although reasoners share the same interface in the OWL API, they can handle entailments differently and return subsets or supersets of the desired ones. Therefore, we need to set all the required parameters in order to extract and describe them precisely. For the computation of entailment metrics, we consider entailments that cover relations between atoms of the following types:

- Equivalent classes and object properties.
- Non-trivial direct subsumptions between entities (classes and object properties).
- Class and object property assertions .

Based on the definition of finite sets of entailments, described in [2], for the computation of direct subsumptions between classes, we select the inferred transitive closure, with the exclusion of asserted of entailments. The main normalised metric that is used to assess entailments is the ratio of entailments to the total number of axioms.

It should be noted that there are different metrics that also consider the entailments as they are computed for both the asserted and inferred ontology. However, they are used to highlight different aspects of the ontology. For example, for the computation of the entailment metrics, the asserted information of

the ontology is excluded. That is because these metrics are used to measure only the size and type of entailments in the ontology. On the other hand, the structural metrics are computed for both the asserted and inferred class hierarchy. These metrics take into account also the asserted information in the ontology and aims to show how the entailments affect the shape of the hierarchy (nesting, complexity and wideness).

In order to assess the complexity of the entailments, we define the *Justification richness*, which is the average number of justifications per entailment. Justifications and their usage are described in [7,8].

3.5 Design style dimensions

We define 20 design style dimensions, which organise the metrics into groups for assessing the aspects of an ontology. These are presented in Figure 3. They comprise the rating scheme that is used to give the metamodel describing the design style of an ontology. Apart from the metrics that are used to assess the size of the ontology, the rest of them are normalised. We deliberately use normalised metrics as these are more convenient for the comparison of two ontologies, which might vary in size, but are similar in terms of richness and complexity.

We selected these dimensions to pinpoint the main aspects of an ontology that normally a user will deal with on the initial inspection; These can be the size, the shape of the hierarchies, the entailments of the ontology etc. On the selected metrics for representing each dimension we applied feature selection algorithms in order to exclude from the set redundant metrics.

For the final induction of a rating scheme for describing the design style of an ontology from a high level, we further normalise the metrics to have a single normalised metric representing each dimension. For example the *class feature richness* metric is defined as $class\ feature\ richness = \frac{\sum CC_i}{CC_n}$, where CC_i represents the normalised metrics of the class characteristics and in this case are the *Avg. defined classes per class*, *Avg. OWLClassExpressions per class*, *Avg. DisjointClass axioms per class*, *Avg. DisjointUnion axioms per class* and CC_n is the total number of characteristics, which in this case is equal to 4. Details about the process that is followed to conclude the rating scheme are presented in the next section.

4 Results

The TONES repository was selected for testing the metrics as it appears to include a variety of different ontologies with different design styles. The metric framework was applied to 112 ontologies from TONES[2] repository, which were classified in less than 50 minutes. For the classification of the ontologies we use the Pellet[3] reasoner. The results are further processed for the induction of the

[2] `http://owl.cs.manchester.ac.uk/repository/`
[3] `http://clarkparsia.com/pellet/`

ontology design styles that are covered in the repository. For this process we use clustering methods to define a rating scheme that divides the set of ontologies into different groups. All the results can be found in detail online[4]. In this section we focus on the process that was followed, and present the main outcomes.

After the application of the metrics in the repository, the next step is to separate the ontologies into groups according to their similar description. The results of the repository are expressed with normalised metrics. For the conclusion of the rating scheme we define three nominal values (classes) to describe each dimension in the rating scheme. The set of classes is $S = \{\mathsf{Small}, \mathsf{Medium}, \mathsf{High}\}$ characterising the ontology in terms of richness and complexity for every dimension. The reason for defining these classes is to make more practical use of metrics and how to interpret a value of a metric. We set the range of these classes into the corresponding values by applying the KMeans [14] clustering algorithm on the data and setting the parameter of clusters equal to 3. Based on the output from clustering, the normalised value of each dimension is mapped to the corresponding class.

The results appear to show an intuitive description of the ontologies. An example is shown in Figure 5. The cell ontology[5] [3] is a medium size ontology, with many annotations, but few entailments, although with many justifications for those entailments. There is much information that is expressed in the class hierarchy, which is wide and complex. There is a medium usage of the class expressions and object properties, which do not have many characteristics. Such a description summarises the main characteristics of a single ontology. However, we do not yet know which design styles are covered by the repository and which ontologies have similar descriptions.

High level description of the cell ontology:

Size :	Medium
Class hierarchy wideness :	High
Class hierarchy complexity :	Medium
ObjectProperty hierarchy wideness :	Small
ObjectProperty hierarchy compexity :	Medium
DataProperty hierarchy wideness :	Small
DataProperty hierarchy complexity :	Small
Class usage :	Small
Class expression usage :	Medium
Class expression complexity :	Small
ObjectProperty usage :	Medium
DataProperty usage :	Small
Individual usage :	Small
Class feature richness :	Small
ObjectProperty feature richness :	Small
DataProperty feature richness :	Small
Individual feature richness :	Small
Annotation feature richness :	High
Entailment richness :	Small
Justification richness :	Medium

Fig. 5. Description of the Cell ontology based on the rating scheme

[4] http://www2.cs.man.ac.uk/~mikroyae/2011/metrics/

[5] http://owl.cs.manchester.ac.uk/repository/download?ontology=http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/anatomy/cell_type/cell.obo

For further analysis of the repository, we performed additional clustering on the results generated from the TONES repository. The dimensions of the rating scheme give a high level metamodel of the ontology. The additional clustering is done using only the 20 dimensions of the rating scheme, which now are not separated into groups but are tested all together. The reason is to highlight the most important design styles that exist in the repository. These clusters use the combination of the metrics and can reveal possible similarities in design style between the ontologies. In this case we do not know how many design styles exist in the repository, thus we choose the EM [14] clustering algorithm, because it can also predict the number of clusters. For the generation of these clusters, we represent the set of classes ($S = \{\mathsf{Small}, \mathsf{Medium}, \mathsf{High}\}$) in the dimensions with the values $S = \{1, 2, 3\}$. We make this transformation because the EM algorithm does not work with nominals. In addition, we do not use the initial normalised values for this clustering as it would be more difficult to cluster values that are relatively close to each other. We perform clustering with the absolute values because they are better separated. The clustering algorithm separates all the data into 5 clusters, the description for which is given in Table 1. In this way, the repository is analysed into five dominant design styles.

Cluster 0 is of small to medium sized ontologies with most of the information expressed in the class hierarchy, which is complex and wide. There are not many class expressions, complex property hierarchies, data properties or individuals. The inferential richness is small to medium while the justification richness is medium to high. In addition, this cluster includes the ontologies with many annotations and annotation features. This cluster includes many bio-ontologies in OBO, such as the fly-taxonomy[6] and the environment[7] ontology.

Cluster 1 is of medium sized ontologies, with a simple class hierarchy and an object property hierarchy of small to medium complexity. There is a small usage of class expressions and a medium object property usage. Also, this cluster seems to include ontologies with data properties as well as individuals with many features and the highest usage of all of the clusters. Finally, it has a small inferential and justification richness. Some ontologies in this cluster are the time[8], units[9] and substance[10].

Cluster 2 includes small ontologies, but with a medium to high inferential and justification richness. Even though its ontologies have a simple class hier-

[6] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/taxonomy/fly_taxonomy.obo`

[7] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://obo.cvs.sourceforge.net/*checkout*/obo/obo/ontology/phenotype/environment/environment_ontology.obo`

[8] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://sweet.jpl.nasa.gov/1.1/time.owl`

[9] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://sweet.jpl.nasa.gov/1.1/units.owl`

[10] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://sweet.jpl.nasa.gov/1.1/substance.owl`

Table 1. Clusters of ontologies revealing design styles in the TONES Repository.

	Cluster				
Attribute	**Cluster 0**	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**
Ontologies	37 (33%)	15 (13%)	24 (21%)	17 (15%)	19 (17%)
Size					
mean	1.129	1.1909	1	1.4122	1
std. dev.	0.4049	0.5	0.4214	0.6001	0.4214
Class Hierarchy Wideness					
mean	3	1.1426	1.0908	2.5892	2.7709
std. dev.	0.9542	0.3498	0.4163	0.7716	0.4203
Class Hierarchy Complexity					
mean	2.0516	1.2858	1.0454	1.9411	2
std. dev.	0.317	0.4518	0.2082	0.2354	0.5204
objectProperty Hierarchy Wideness					
mean	1.1032	1.0477	2.0437	1.2333	1.4553
std. dev.	0.3042	0.2132	0.6389	0.4229	0.498
objectProp Hierarchy Compexity					
mean	1.5099	1.6242	2.7209	2.1758	2.377
std. dev.	0.5491	0.4843	0.4598	0.3829	0.4846
Class Usage					
mean	1	1.2387	1	1.1731	1.5323
std. dev.	0.3675	0.5265	0.3675	0.3784	0.499
Class Expression Usage					
mean	1.0721	1.1432	1.2723	2.4679	1.8502
std. dev.	0.2587	0.4671	0.5375	0.6974	0.8581
Class Expression Complexity					
mean	1	1	1.3177	1.4711	1.3769
std. dev.	0.488	0.488	0.4656	0.7762	0.7324
ObjectProperty Usage					
mean	1	1.0955	1	1.5889	1
std. dev.	0.388	0.4264	0.388	0.6911	0.388
DataPropery Usage					
mean	1	1.0955	1	1	1.0754
std. dev.	0.2105	0.4264	0.2105	0.2105	0.264
Individual Usage					
mean	1.0721	1.4774	1.0454	1.1744	1.3954
std. dev.	0.2586	0.7321	0.2081	0.3794	0.4889
Class Feature Richness					
mean	1.3035	1	1.2723	2.1134	2.2308
std. dev.	0.4598	0.6145	0.4451	0.4691	0.5728
Object Property Feature Richness					
mean	1	1.2	1.3998	1.0589	1.0754
std. dev.	0.4349	0.5586	0.6771	0.2354	0.264
DataProperty Feature Richness					
mean	1	1.5251	1.0454	1.0589	1
std. dev.	0.397	0.732	0.2081	0.2354	0.0043
Individual Feature Richness					
mean	1	1.1909	1	1	1
std. dev.	0.2297	0.5	0.2297	0.2297	0.2297
Annotations Feature Richness					
mean	2.0218	1.1936	1.5874	1.9988	1.409
std. dev.	0.891	0.5887	0.5775	0.9086	0.6478
Entailment Richness					
mean	1.1237	1.4735	2.5466	1	2.8246
std. dev.	0.3294	0.7294	0.7813	0.8772	0.3921
Justification Richness					
mean	2.0516	1.8285	2.4807	1.9965	2.3815
std. dev.	0.317	0.5257	0.6567	0.6838	0.4857

archy, there is a rich object property hierarchy having many object property characteristics.

Cluster 3 includes the big ontologies in the repository, with a wide class hierarchy of medium to high complexity and a quite complex object property hierarchy. Its ontologies also have a rich description of the classes, with complex class expressions and high class usage. However, the inferential richness is small and the justification richness is small to medium.

Cluster 4 has the richest ontologies in terms of inferential and justification richness. However, its ontologies are of small size. The class hierarchies are wide, with a high complexity. There appears also to be rich object property hierarchies, but the properties do not have many property characteristics. This cluster also has the highest class expression usage. This cluster seems to have many 'toy' ontologies, which have been used for demonstrating OWL usage, such as the family-tree[11], generations[12] and university[13] ontology.

Some of these clusters share characteristics, and thus appear closer in design style. However, there is at least one dimension in the rating scheme that splits them into different groups. It should be noted though, that this clustering does not produce fully accurate results, as there is expected to be some incorrect classifications (e.g. an ontology might be very close to two clusters). However, the purpose of this clustering is to give a general view and analysis of the repository, highlighting the most common design styles of its ontologies.

In order to test how well the data were clustered, we use classifications algorithms with 10 fold cross validation to classify the ontologies to the corresponding 5 design styles, which were identified by the clustering. The accuracy of the classification algorithms (SMO, BayesNet, RandomForest) [14] were between 80.4% to 83.9% with an F-Measure range between 0.75 and 0.90.

5 Conclusions

The contribution of this paper is the definition and categorisation of metrics for the description of underlying design styles in ontologies. This high level abstract description of an ontology can be considered as a metamodel, giving an intuition of the main characteristics of the ontology as a collection of axioms and entailments. In particular, we defined a rating scheme consisting of 20 dimensions, covering five different aspects of an ontology. Each dimension is assessed using mainly normalised metrics. The metric framework presented here was built on top of the existing metric package that is implemented in the OWL API. The results of the application of the metric framework in 112 ontologies from the TONES repository revealed five important design styles in the repository. It should be noted that these design styles depend on the content of the repository, and these results do not exclude the existence of other design styles. However,

[11] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://www.co-ode.org/roberts/family-tree.owl`

[12] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://www.owl-ontologies.com/generations.owl`

[13] `http://owl.cs.manchester.ac.uk/repository/download?ontology=http://www.mindswap.org/ontologies/debugging/university.owl`

the identification of potential design styles should be helpful for the analysis of a repository, as it can show from a high level the type of ontologies that reside in the repository. It also reveals similarities between ontologies that are included in the same cluster of design style; information that should be useful for the integration and reuse of ontologies.

The final five clusters that were detected using clustering algorithms in the repository, show five different design styles of ontologies. Their inspection showed that there were small but complex ontologies as well as relatively big and simple ontologies. In addition, ontologies of similar apparent design style were found in the same cluster (e.g. they were toy ontologies, or were using the same OWL features). We believe the approach taken here for providing a metamodel based on metrics, has a potential as a starting point when comprehending an ontology. Future work remains to demonstrate its utility for users of ontologies. This will involve the evaluation and refinement of the rating scheme for categorising and assessing ontologies in terms of their design. In addition, future work will also involve the testing of the metric framework with multiple repositories in order to highlight the most common design styles of their ontologies.

Metrics are abstract descriptions that can give an intuition of the overall design style of the ontology. Based on this initial information the user has an indication of what to expect to find in the ontology. This should be helpful for the decision on what comprehension tactic to follow. This framework of metrics, together with the use of straight-forward data-mining techniques, thus offers the possibility of providing ontologists with a high-level abstraction of an ontology's design style as an aid to ontology comprehension.

References

1. F. Baader, B. Hollunder, B. Nebel, H. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence*, 4(2):109–132, 1994.
2. S. Bail, B. Parsia, and U. Sattler. Extracting finite sets of entailments from owl ontologies. International Workshop on Description Logics (DL), 2011.
3. J. Bard, S. Rhee, and M. Ashburner. An ontology for cell types. *Genome Biology*, 6(2):R21, 2005.
4. A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling ontology evaluation and validation. *The Semantic Web: Research and Applications*, pages 140–154, 2006.
5. A. Gangemi and P. Mika. Understanding the semantic web through descriptions and situations. In *CoopIS/DOA/ODBASE'03*, pages 689–706, 2003.
6. A. Gibson, K. Wolstencroft, and R. Stevens. Promotion of ontological comprehension: Exposing terms and metadata with web 2.0. In *Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge at 16th International World Wide Web Conference (WWW2007)*. Citeseer, 2007.
7. M. Horridge, J. Bauer, B. Parsia, and U. Sattler. Understanding entailments in OWL. In *Proc. of OWLED*. Citeseer, 2008.
8. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in owl. *The Semantic Web-ISWC 2008*, pages 323–338, 2010.

9. M. Horridge, B. Parsia, and U. Sattler. The state of bio-medical ontologies. ISMB conference, Bio-Ontologies SIG workshop, 2011.
10. D. Kang, B. Xu, J. Lu, and W. Chu. A complexity measure for ontology based on uml. In *Distributed Computing Systems, 2004. FTDCS 2004. Proceedings. 10th IEEE International Workshop on Future Trends of*, pages 222 – 228, May 2004.
11. C. Keet. Enhancing comprehension of ontologies and conceptual models through abstractions. *AI* IA 2007: Artificial Intelligence and Human-Oriented Computing*, pages 813–821, 2007.
12. N. Noy, A. Chugh, and H. Alani. The CKC challenge: Exploring tools for collaborative knowledge construction. *Intelligent Systems, IEEE*, 23(1):64–68, 2008.
13. A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. *Engineering Knowledge in the Age of the SemanticWeb*, pages 63–81, 2004.
14. P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.
15. S. Tartir, I. Arpinar, M. Moore, A. Sheth, and B. Aleman-Meza. OntoQA: Metric-based ontology quality analysis. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, volume 9. Citeseer, 2005.
16. M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods. Ontology reuse and application. In *Formal Ontology in Information Systems*, volume 179, page 192. Citeseer, 1998.
17. D. Vrandecic and Y. Sure. How to design better ontology metrics. In *The Semantic Web: research and applications: 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007: proceedings*, page 311. Springer-Verlag New York Inc, 2007.
18. H. Yao, A. Orme, and L. Etzkorn. Cohesion metrics for ontology design and application. *Journal of Computer Science*, 1(1):107–113, 2005.