The Journal of Systems and Software 83 (2010) 803–814

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Measuring design complexity of semantic web ontologies

Hongyu Zhang [a,*], Yuan-Fang Li [b], Hee Beng Kuan Tan [c]

[a] School of Software, Tsinghua University, Beijing 100084, China
[b] School of ITEE, The University of Queensland, Brisbane, Australia
[c] School of EEE, Nanyang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 24 January 2009
Received in revised form 19 November 2009
Accepted 19 November 2009
Available online 1 December 2009

Index Terms:
Design complexity
Ontology
OWL
Ontology metrics
Software metrics

ABSTRACT

Ontology languages such as OWL are being widely used as the Semantic Web movement gains momentum. With the proliferation of the Semantic Web, more and more large-scale ontologies are being developed in real-world applications to represent and integrate knowledge and data. There is an increasing need for measuring the complexity of these ontologies in order for people to better understand, maintain, reuse and integrate them. In this paper, inspired by the concept of software metrics, we propose a suite of ontology metrics, at both the ontology-level and class-level, to measure the design complexity of ontologies. The proposed metrics are analytically evaluated against Weyuker's criteria. We have also performed empirical analysis on public domain ontologies to show the characteristics and usefulness of the metrics. We point out possible applications of the proposed metrics to ontology quality control. We believe that the proposed metric suite is useful for managing ontology development projects.

© 2009 Elsevier Inc. All rights reserved.

1. Introduction

The Semantic Web (Berners-Lee et al., 2001) is an envisioned extension of the current World Wide Web in which data is given well defined meaning so that software agents can autonomously process the data. It is also widely believed that Semantic Web ontologies provide a solution to the knowledge management and integration challenges (Searls, 2005; Auer et al., 2008; Smith et al., 2007; Ruttenberg et al., 2009). Ontology languages such as RDF Schema (Brickley and Guha, 2004) and OWL (Horrocks et al., 2003) provide essential vocabularies to describe domain knowledge, the underlying common model for data aggregation and integration.

A great deal of efforts are being invested in applying Semantic Web ontologies to create mutually agreeable and consistent vocabularies to describe domain knowledge from disparate sources. For example, the NCI Thesaurus Ontology[1] developed and actively curated by the National Cancer Institute is such an OWL ontology. It defines 60,000+ named classes, a roughly equal number of anonymous classes and 100,000+ connections (properties) from and to these classes. This ontology covers information about nearly 10,000 cancers and 8000 therapies. More recently, as the result of the Linked Data project[2], a large number of inter-connected RDF datasets, such as DBPedia[3], DBLP[4], FOAF[5], US census data, etc., are being generated and integrated. With more information being converted to RDF/OWL and integrated, we believe that properly designed OWL ontologies is essential to the effective management, reuse and integration of these data.

As ontologies grow in size and number, it is important to be able to measure their complexity quantitatively. It is well known that "You cannot control what you cannot measure" (DeMarco, 1986). Quantitative measurement of complexity can help ontology developers and maintainers better understand the current status of the ontology, therefore allowing them to better evaluate its design and control its development process. Research on human cognition shows that humans have limited capabilities in information processing (e.g., Miller, 1956; Simon, 1974). Experiences from the software engineering field also suggest that there are correlations between software complexity and quality (such as reusability and maintainability) (Li and Cheung, 1987; Wilde et al., 1993; Koru and Tian, 2003; Zhang et al., 2007). We believe such correlation exists between ontology complexity and quality too – in general, more complex ontologies tend to be more difficult for a human to comprehend, therefore more difficult to be maintained and reused.

In software engineering domain, the term software complexity is often defined as "the difficulty of performing tasks such as coding, debugging, testing and modifying the software" (Zuse, 1991).

* Corresponding author. Tel.: +86 10 62773275.
E-mail addresses: hongyu@tsinghua.edu.cn (H. Zhang), liyf@itee.uq.edu.au (Y.-F. Li), ibktan@ntu.edu.sg (H.B.K. Tan).

[1] http://www.cancer.gov/cancertopics/terminologyresources.
[2] http://linkeddata.org.
[3] http://dbpedia.org/.
[4] http://www.informatik.uni-trier.de/ley/db/.
[5] http://www.foaf-project.org.

0164-1212/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2009.11.735

Software metrics (Fenton and Pfleeger, 1998) are designed to quantify software products and processes. In the same spirit, we define ontology design complexity as the difficulty of performing tasks such as developing, reusing and modifying the ontology. This paper addresses the increasing needs for measuring the complexity of ontology designs by utilizing the concepts of software metrics.

We consider ontology complexity as a profile multidimensional construct (Law et al., 1998), which is formed as various combinations of dimensional characteristics and cannot be measured directly using a single metric. Therefore, we propose a suite of metrics (at both the ontology-level and class-level) to measure different aspects of the design complexity of ontologies. Together, these metrics help us gain a more complete understanding of ontology complexity.

Weyuker's criteria (Weyuker, 1988) is a set of properties for evaluating software metrics. We analyze the applicability of Weyuker's criteria in the context of ontology and analytically evaluate our proposed metrics against them.

We have also collected real-world ontologies from public domains to show the characteristics of the proposed metrics and to evaluate the usefulness of the metrics. By doing so, we seek to demonstrate the level of rigor required in the development of useful ontology metrics. An automated tool based on the Protégé-OWL API[6] has been developed to facilitate metric computation. We also point out how the proposed metrics can be applied to ontology quality control. Our proposed metrics are theoretically and empirically sound, are capable of revealing the internal structure of ontologies, and are useful for ontology engineering practices.

The rest of the paper is organized as follows: in Section 2 we introduce the background on complexity measures and related work. Section 3 introduces the problem of evaluating ontology complexity and formally defines the graphic-centric representation of OWL ontologies, for the discussion of complexity metrics. In Section 4, we describe our proposed metric suite. Sections 5 and 6 give analytical evaluation and empirical evaluation of the metrics, respectively. Section 7 discusses how the proposed metrics can be applied to ontology development practices. Finally, in Section 8 we conclude the paper and suggest future work directions.

2. Background and related work

Complexity has been a subject of considerable research. In cognitive psychology, a convenient metaphor treats human cognition as a computer-like information processor (Lindsay and Norman, 1977). Both of them involve similar concepts such as input/output, memory, processing power, and critical resources.[7] Like an information processor, it is believed that humans' problem solving and other complex cognitive processes have limited capabilities, which restrict the understanding and development of complex structures. For example, the seminal works on 7 ± 2 limits (Miller, 1956) and the size of a memory chunk (Simon, 1974) reveal that a human can only cope with limited information at a time via short-term memory, independent of information content. It is also discovered that the difficulty of a task can be measured by the number of cognitive resources required to perform the task (Sheridan, 1980).

In software engineering domain, researchers and engineers attempt to quantitatively understand the complexity of the software undertaken and to find the relationships between the complexity and the difficulty of development/maintenance task. Many software complexity metrics have been proposed over the years. Examples include cyclomatic complexity (McCabe, 1976), coupling metrics (Fenton and Melton, 1990) and the CK object-oriented design metrics (Chidamber and Kemerer, 1994). Many researchers have shown that complexity measures can be early indicators of software quality. For example, empirical evidence supporting the role of object-oriented metrics, especially the CK metrics, in determining software defects was provided in (Basili et al., 1996; Subramanyam and Krishnan, 2003).

An ontology is a specification of a conceptualization (Gruber, 1993), which can capture reusable knowledge in a domain. In software engineering area, object-oriented design also involves the conceptualization of domain knowledge, producing deliverables such as class diagrams. It has been shown that object-oriented modeling languages can be grounded on ontological theory (Opdahl and Henderson-Sellers, 2001). There is much similarity between object-oriented design and ontology development, suggesting that we may borrow the principles and methods from software metrics research to design ontology metrics. However, we cannot apply the metrics originally designed for software complexity to ontology without adaptation. Many software complexity metrics are based on program control flow or the number of methods. For example, three of the six CK metrics involves information about methods, which are not applicable to ontology. Therefore it is necessary to design a new suite of metrics for measuring complexity of ontologies.

In recent years, various metrics for measuring ontologies were proposed. For example, Yao et al. suggested three metrics (Yao et al., 2005) (namely the number of root classes, the number of leaf class, and the average depth of inheritance tree) to measure the cohesiveness of an ontology. Kang et al. (2004) proposed an entropy-based metric for measuring the structural complexity of an ontology represented as UML diagram. These efforts only focus on one or two aspects of structural complexity and lack sound theoretical or empirical validations.

Some researchers also proposed integrated frameworks for ontology measurement. For example, Gangemi et al. (2006) proposed a meta-ontology O^2 that characterizes ontologies as semiotic objects. Based on this ontology they identified three types of measures for ontology evaluation: structural measures, functional measures and usability-profiling measures. A large number of potential metrics were proposed as well. Some of these metrics cannot be automatically calculated, limiting their utility. It also did not provide an empirical analysis for the metrics.

In Wang et al. (2006), a large number ($\sim$1300) of OWL ontologies were collected and statistically analyzed. The main focus of that work is ontology expressivity (e.g., to which OWL species – Lite, DL or Full – an ontology belongs) and consistency characteristics. Besides expressivity, an analysis on the shape of ontology class hierarchy (a graph of subsumptions) was also presented. The authors compared the morphological changes between the classified and inferred (as in OWL reasoning) versions of a class hierarchy and suggested that it may be useful to determine which classes are over- or under-modeled. Their work on graph morphology of class hierarchies is similar to the intention of our tree impurity *TIP* metric that will be presented in Section 4.

Vrandečić and Sure (2007) proposed guidelines for creating ontology metrics based on the notions of "normalization". Their work laid out a set of principles for designing stable, semantic-aware metrics. They proposed five normalization steps, namely: (i) name anonymous classes, (ii) name anonymous individuals, (iii) materialize the subsumption hierarchy and unify names, (iv) propagate instances to deepest possible class or property within the hierarchy, and (v) normalize properties. The normalization process attempts to transform the ontology into a semantically-equivalent form to facilitate the creation of "semantic-aware" metrics. As we stated previously, the objective of our research is to measure

[6] http://protege.stanford.edu/overview/protege-owl.html.

[7] We should note that although we use this metaphor, we are not saying that human's brain functions like a Von Neumann computer.

the design complexity of ontologies and to evaluate the usefulness of metrics in ontology quality control. Aggressive normalization may drastically change the "shape" of an ontology. As a result, metrics based on the normalized ontology may not faithfully represent the complexity of the original one. In fact, in Vrandečić and Sure (2007), the authors also state that "normalization is not an applicable solution for every metric", or wrong results could be returned. Therefore, in this work, we choose to apply minimal normalization to preserve the original form of ontologies as much as possible. We only consider the normalization of anonymous classes for some of the metrics.

Burton-Jones et al. (2005) proposed a metrics suite based on the semiotic frameworks and demonstrated how the metrics can be used to assess the usefulness of DAML ontologies for their semantic retrieval system. Their metrics suite includes measures for syntactic quality, semantic quality, pragmatic quality, and social quality. This suite, including metrics such as lawfulness, richness, relevance and history, are used to assess the "quality" of DAML ontologies. It covers both the intrinsic properties such as syntactic correctness of ontological terms and the relationship the ontology being audited has with its context, e.g., task domain and other ontologies.

Based on the above analysis, we believe that there is still a lack of systematic method for measuring the design complexity of ontologies. In the following sections, we introduce a suite of structural metrics and evaluate the proposed metrics against established criteria for validity. We also collect empirical data from real-world, public ontologies to show the characteristics and usefulness of the proposed metrics in ontology development. The proposed metrics can be integrated into a more comprehensive metrics suite (such as the one proposed in Burton-Jones et al., 2005) to measure the overall quality of an ontology.

3. The Graph-centric representation of ontologies

One may perceive that the larger the file size and the more the number of classes and properties, the more complex an ontology is. However, we argue that it is very difficult to measure ontology complexity with a single metric.

For example, ontology size alone is not a sufficient complexity measure. Take the NCI Thesaurus ontology[8] as an example. It is one of the largest ontologies available (81.5 MB). Another ontology, the gene ontology[9], of less than half of its size (39.2 MB), has more than twice the number of nodes than the NCI Thesaurus ontology has. Apart from physical size, very often neither can we judge the complexity of ontology design solely by counting the number of classes and properties. Instead, a set of metrics shall be used to measure different aspects of the complexity in order to achieve a more complete understanding. In this research, we have derived a set of metrics based on the graph-centric representation of ontologies.

For the discussion of ontology complexity metrics, we define a graph-centric view for OWL ontologies. Specifically, an ontology can be viewed as a directed graph $G = \langle N, P, E \rangle$, where N is a set of nodes representing classes and individuals; P is a set of nodes representing properties; and E is a set of edges representing property instances and other relationships between nodes in the graph G. $E \subseteq N \times P \times N$. N includes both N_n (named classed and individuals) and N_a (anonymous classes and individuals). P includes both P_n (user-defined properties) and P_a (OWL/RDFS properties such as `rdfs:subClassOf` and `owl:equivalentClass`).

[8] http://www.nci.nih.gov/cancerinfo/terminologyresources.
[9] http://www.geneontology.org/.

$$
\begin{aligned}
\tau(A) &= A, \text{where } A \in N && (1)\\
\tau(a) &= a, \text{where } a \in N && (2)\\
\tau(Q) &= Q, \text{where } Q \in P && (3)\\
\tau(\neg\, C) &= \_\texttt{:0}, \text{where } \_\texttt{:0} \in N_a \text{ and} && (4)\\
&\quad (\_\texttt{:0}, \texttt{rdfs:subClassOf}, \texttt{owl:Thing}) \in E \text{ and}\\
&\quad (\_\texttt{:0}, \texttt{owl:complementOf}, \tau(C)) \in E\\
\tau(C \sqcap D) &= \_\texttt{:1}, \text{where } \_\texttt{:1} \in N_a \text{and} && (5)\\
&\quad (\_\texttt{:1}, \texttt{rdfs:subClassOf}, \texttt{owl:Thing}) \in E \text{ and}\\
&\quad (\_\texttt{:1}, \texttt{rdfs:subClassOf}, \tau(C)) \in E \text{ and}\\
&\quad (\_\texttt{:1}, \texttt{rdfs:subClassOf}, \tau(D)) \in E\\
\tau(\{a, b, \ldots\}) &= \_\texttt{:2}, \text{where } \_\texttt{:2} \in N_a \text{and} && (6)\\
&\quad (\_\texttt{:2}, \texttt{rdfs:subClassOf}, \texttt{owl:Thing}) \in E \text{ and}\\
&\quad (\tau(a), \texttt{rdf:type}, \_\texttt{:2}) \in E \text{ and}\\
&\quad (\tau(b), \texttt{rdf:type}, \_\texttt{:2}) \in E \text{ and} \ldots\\
\tau(\exists\, Q.C) &= \_\texttt{:3}, \text{where } \_\texttt{:3} \in N_a \text{ and} && (7)\\
&\quad (\_\texttt{:3}, \texttt{rdfs:subClassOf}, \texttt{owl:Thing}) \in E \text{ and}\\
&\quad (\_\texttt{:3}, \tau(Q), \tau(C)) \in E\\
\tau(\forall\, Q.C) &= \_\texttt{:4}, \text{where } \_\texttt{:4} \in N_a \text{ and} && (8)\\
&\quad (\_\texttt{:4}, \texttt{rdfs:subClassOf}, \texttt{owl:Thing}) \in E \text{ and}\\
&\quad (\_\texttt{:4}, \tau(Q), \tau(C)) \in E\\
\tau(Q : a) &= \_\texttt{:5}, \text{where } \_\texttt{:5} \in N_a \text{ and} && (9)\\
&\quad (\_\texttt{:5}, \texttt{rdfs:subClassOf}, \texttt{owl:Thing}) \in E \text{ and}\\
&\quad (\_\texttt{:5}, \tau(Q), \tau(a)) \in E
\end{aligned}
$$

Fig. 1. Translation rules from OWL descriptions to the graph-centric representation.

$$
\begin{aligned}
\tau(A \sqsubseteq D) &= (\tau(A), \texttt{rdfs:subClassOf}, \tau(D)) \in E && (10)\\
\tau(A \equiv D) &= (\tau(A), \texttt{owl:equivalentClass}, \tau(D)) \in E && (11)\\
\tau(C \sqcap D = \bot) &= (\tau(C), \texttt{owl:disjointWith}, \tau(D)) && (12)\\
\tau(S \sqsubseteq Q) &= (\tau(S), \texttt{rdfs:subPropertyOf}, \tau(Q)) \in E && (13)\\
\tau(\geq 1 Q \sqsubseteq C) &= (\tau(Q), \texttt{rdfs:domain}, \tau(C)) && (14)\\
\tau(\top \sqsubseteq \forall\, Q.C) &= (\tau(Q), \texttt{rdfs:range}, \tau(C)) && (15)\\
\tau(a \in C) &= (\tau(a), \texttt{rdf:type}, \tau(C)) \in E && (16)\\
\tau(\langle a, b\rangle \in Q) &= (\tau(a), \tau(Q), \tau(b)) \in E && (17)\\
\tau(a = b) &= (\tau(a), \texttt{owl:sameAs}, \tau(b)) \in E && (18)\\
\tau(a \neq b) &= (\tau(a), \texttt{owl:differentFrom}, \tau(b)) \in E && (19)
\end{aligned}
$$

Fig. 2. Translation rules from OWL axioms to the graph-centric representation.

Formally, we define the translation function τ from OWL constructs to G in Figs. 1 and 2. *A and B* represent named classes; *C and D* represent potentially complex (OWL class descriptions and restrictions) classes; *a and b* represent individuals; *Q and S* represent properties; and _:0, _:1, etc. represent anonymous classes in N_a. For brevity reasons, OWL abstract syntax (Horrocks et al., 2003) is used.

Specifically, rules 1 and 2 state that named classes and individuals are translated into nodes in N of G. Rule 3 state that named properties are translated into P of G. Rules 4–9 specify how anonymous class descriptions and restrictions are translated: as nodes such as _:n in N_a of G, with the translation function τ recursively applied to the inner language constructs. Rules 10–19 specify how OWL axioms and assertions are translated, in a similar fashion. Translation rules for other OWL descriptions and axioms can be similarly defined and are omitted here.

The inheritance hierarchy of an ontology can be described as $G' = \langle N', P', E' \rangle$, where N' is the set of nodes representing classes, P' is the RDF property `rdfs:subClassOf`, and E' is the set of edges

```
LivingThing ⊑ ⊤                    Plant ⊑ LivingThing
Animal ⊑ LivingThing               Grass ⊑ Plant
Animal ⊑∀ eats.LivingThing         Tree ⊑ Plant
Person ⊑ Animal                    Cow ⊑ Animal
Carnivore ⊑ Animal                 Herbivore ⊑ Animal
Carnivore ⊑∀ eats.Animal           Herbivore ⊑∀ eats.Plant
Animal ⊓ Plant = ⊥
```

Fig. 3. The OWL axioms in the "Animal" ontology.

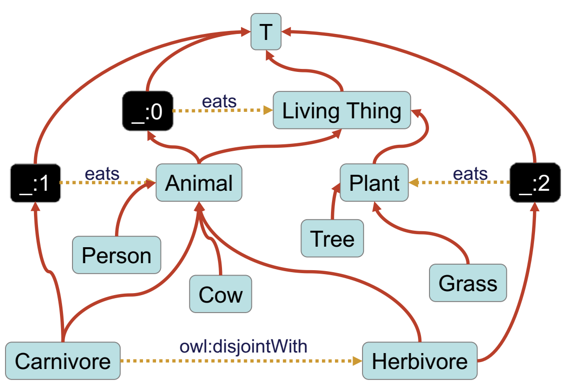

Fig. 4. The Animals ontology represented as a graph.

representing the inheritance relationship (`rdfs:subClassOf`) among classes.[10]

A number of points are worth discussion.

- The graph-centric representation is not a lossless translation; nor does it preserve the OWL semantics. It is meant to represent the ontology structure with minimal "normalization" so as to represent the structure of the original ontology as faithfully as possible.
- N includes a special class `owl:Thing`, which is added for uniformity in the calculation of inheritance-related metrics. More discussions will be presented when we discuss class-level metrics.
- Every top-level class node $C \in N$ generates an additional edge: $(C, \mathtt{rdfs:subClassOf}, \mathtt{owl:Thing}) \in E$, if it is not already present.
- Anonymous OWL classes (e.g., class descriptions such as `owl:unionOf`, and `owl:intersectionOf`; and class restrictions such as `owl:someValuesFrom` and `owl:allValuesFrom`) add to the expressivity of ontologies. Hence, anonymous classes are represented in N_a of G and are included in the calculation of relevant metrics.
- Only OWL individuals used in the definition of other classes (e.g., through the use of `owl:oneOf` enumeration construct) are included in N, and hence the calculation of complexity metrics.
- Annotation-related entities are not considered in G and the metric suite.

As an example, Fig. 3 shows part of the OWL axioms from the "Animals" ontology that is adapted from (Rector et al., 2005). Fig. 4 shows the translation of the axioms into the graph representation. Unlabeled edges represent `rdfs:subClassOf` relationships among classes. The three nodes `_:0`, `_:1` and `_:2` are created, representing anonymous classes used as value restrictions in the definition of other classes. Note that all top-level classes have `owl:Thing` (⊤ in Fig. 4 at the top) as their super class, as stated above.

In this paper, we use the graphical representations of ontologies (such as the one shown in Fig. 4) to help define some complexity measures intuitively. We classify these metrics into two sets: one for measuring the overall design complexity of an ontology (ontology-level metrics), and the other for measuring the complexity of internal structure (class-level metrics). The larger the metric value, the more the cognitive resources are required to understand and maintain the ontology, therefore the greater the complexity is. We will formally present the proposed suite of complexity metrics in the next section.

4. The proposed metrics for measuring complexity of ontology

4.1. Ontology-level metrics

We propose four ontology-level metrics (namely SOV, ENR, TIP, and EOG) to measure complexity of an ontology design:

4.1.1. Size of vocabulary (SOV)

Definition. SOV measures the amount of vocabulary defined in an ontology. Given a graph representation $G = \langle N, P, E \rangle$ of an ontology, *SOV* is defined as the cardinality of the name entities N_n and P_n in G:

$$SOV = |N_n| + |P_n|$$

where N_n representing named classes and individuals, and P_n representing user-defined properties.

Rationale. An ontology contains structured vocabulary, including named classes (representing a collection of individuals), property (representing a collection of pairs of individuals/literals), and individuals (instances of classes). *SOV* measures the complexity of an ontology by counting the total number of named entities. In an ontology graph G, *SOV* is the total number of N_n and P_n defined by the ontology. The greater the *SOV*, the greater the size of an ontology, and the greater the time and effort that are required to build and maintain the ontology. Note that we choose not to include anonymous classes (N_a) and OWL/RDF default properties (P_a) into the calculation of *SOV* as they do not introduce new vocabularies to an ontology.

Example. For the Animals ontology described in Section 3, the *SOV* is 11 (10 named classes and 1 property).

4.1.2. Edge node ratio (ENR)

Definition. For an ontology graph $G = \langle N, P, E \rangle$, the *ENR* is defined as follows:

$$ENR = \frac{|E|}{|N|}$$

as the division of the number of edges ($|E|$) by the number of nodes ($|N|$) (including both named and anonymous) in G.

Rationale. *ENR* measures the connectivity density since it increases as more edges are added between nodes (classes and individuals). The greater the *ENR*, the greater the complexity of an ontology.

Example. For the Animals ontology described in Section 3, the number of nodes is 13 and the number of edges is 19, thus *ENR* is 1.46.

[10] We assume that an inheritance hierarchy is a directed, acyclic graph. For inheritance hierarchy that contains cycles, we suggest identifying equivalent classes and removing the cycles, thus transforming the hierarchy into a semantic equivalent acyclic graph.

4.1.3. Tree impurity (TIP)

Definition. *TIP* measures how far an ontology's inheritance hierarchy $G' = \langle N', P', E' \rangle$ deviates from being a tree. It is defined as:

$$TIP = |E'| - |N'| + 1$$

where $|E'|$ is the number of `rdfs:subClassOf` edges and $|N'|$ is the number of nodes (including both named and anonymous) in an ontology's inheritance hierarchy.

Rationale. A well-structured ontology is composed of classes organized through the inheritance relationship. Single inheritance leads to a "pure" tree hierarchy, while multiple inheritances allow a subclass to inherit from more than one super class, making inheritance hierarchy a graph. *TIP* measures the degree of tree impurity. The value of *TIP* can be seen as the number of extra edges that an inheritance hierarchy differs from a tree structure. A tree with n nodes always has $n-1$ edges, therefore $TIP = 0$. The greater the *TIP*, the more an ontology's inheritance hierarchy deviates from a pure tree structure, and the greater the complexity of an ontology.

Note that in the calculation of *TIP*, we take into consideration the top class `owl:Thing` and anonymous classes created during the translation process. As we stated in Section 3, in the translation, each class node C with no explicit superclass nodes will have an edge added for it: $(C, \texttt{rdfs : subClassOf}, \texttt{owl : Thing})$. Such an addition ensures that *TIP* is always non-negative.

Example. For the Animals ontology described in the previous section, the classes are not organized through single inheritance, evident with the presence of OWL class axioms. Therefore, in the inheritance hierarchy $|E'| = 15$ and $|N'| = 13$, thus $TIP = 3$.

4.1.4. Entropy of graph (EOG)

Definition. The *EOG* is an entropy measure of the ontology graph. It is defined as:

$$EOG = -\sum_i p(i) log_2 p(i)$$

where $p(i)$ is the probability of a node (including both named and anonymous) having i edges (both incoming and outgoing degrees).

Rationale. *EOG* is a metric of uncertainty, which measures how diverse (uncertain) the structure of an ontology is. The maximum value $EOG_{max} = log_2 n$ is obtained for $p(i) = 1/n$, and the minimum value $EOG_{min} = 0$ is obtained when all nodes have the same degree distribution. Lower *EOG* indicates the existence of more structural patterns, therefore the ontology is more regular and less complex.

Example. For the Animals ontology described in the previous section, the probability of a node having one, three, four and seven degrees are 30.77%, 38.46%, 23.08% and 7.69%, respectively, therefore $EOG = 1.83$.

4.2. Class-level metrics

We also propose four metrics (namely *NOC, DIT, CID*, and *COD*) to measure the design complexity of an ontology at the class-level:

4.2.1. Number of children (NOC)

Definition. for a given class *C, NOC* measures the number of its immediate children in the ontology inheritance hierarchy G', as follows:

$$NOC_C = \#\{D | D \in N' \wedge (D, \texttt{rdfs : subClassOf}, \texttt{C}) \in E'\},\quad \text{where} \quad C \in N'$$

where symbol $\#$ denotes the cardinality of $\{D | D \in N' \wedge (D, \texttt{rdfs : subClassOf}, \texttt{C}) \in E'\}$, which is the set of all immediate child class of class C.

Rationale. The *NOC* metric is the same as the *NOC* metric introduced by Chidamber and Kemerer (1994). *NOC* is the number of subclasses that directly inherit from a given class. As inheritance is a form of reuse, the greater the *NOC* value, the greater the reuse. A greater *NOC* value also indicates that, if a change to this class is made, more subclasses may be affected and more efforts are required to test and maintain the subclasses.[11]

Example. For the animals ontology described in Section 3, the *NOC* value for class `Plant` is 2 and for class `Animal` is 4.

4.2.2. Depth of inheritance (DIT)

Definition. *DIT* measures the length of the longest path from a given class C to the root class in an ontology inheritance hierarchy G'.

Rationale. The *DIT* metric is the same as the *DIT* metric introduced by Chidamber and Kemerer (1994). *DIT* is a measure of number of ancestor classes that can potentially affect the class. A greater *DIT* value shows that the class resides deeper in the inheritance hierarchy and reuses more information from its ancestors. A greater *DIT* value also indicates that the class is more difficult to maintain as it is likely to be affected by changes in any of its ancestors.

In the calculation of *DIT*, the class hierarchy G' is traversed only once, in a top-down and depth-first manner, with `owl:Thing` as the starting root node. For each class, its visited descendent classes are recorded for cycle detection so that the traversal is guaranteed to terminate.

Example. For the Animals ontology described in Section 3, the *DIT* values for the `Plant` class and the `Animal` class are both 2.

4.2.3. Class in-degree (CID)

Definition. For a given class *C, CID* measures the number of edges pointing to a node in an ontology graph G:

$$CID_C = \#\{(D, Q, C) \in E | D \in N \wedge Q \in P\},\quad \text{where} \quad C \in N$$

Rationale. The value of in-degree represents the usage of a given class by other nodes. The higher the *CID* value, the more the number of nodes dependent on it. Therefore, changes to this class may affect more classes.

Example. For the Animals ontology described in Section 3, the *CID* value for class `Plant` is 3 and for class `Animal` is 5.

4.2.4. Class out-degree (COD)

Definition. For a given class *C, COD* measures the number of edges leaving C in the ontology graph G:

$$COD_C = \#\{(C, Q, D) \in E | D \in N \wedge Q \in P\},\quad \text{where} \quad C \in N$$

[11] Currently for NOC we only consider immediate subclasses. We should note that the impact of a class may propagate to its indirect subclasses too. Understanding and measuring the propagation of change impact will be an interesting future work.

Rationale. The value of out-degree represents the number of nodes referred to by a given class. The higher the *COD* value, the more the number of classes a class depends on. Therefore, if any of these nodes are changed, this class needs to be re-examined.

Example. For the Animals ontology described in the previous section, the *COD* value for the `Plant` class is 1 and for the `Animal` class is 2.

5. Analytical evaluation of the complexity metrics

Weyuker has proposed a set of properties for evaluating the usefulness of software complexity metrics (Weyuker, 1988). Although some researchers offered critique (especially on its property 9) (Zuse, 1991; Cherniavsky and Smith, 1991; Gursaran and Roy, 2001; Zhang and Xie, 2002), these properties do provide formal criteria for evaluating the behavior of a metric and are therefore widely adopted (Chidamber and Kemerer, 1994; Harrison, 1992). For a software complexity metric *M*, Weyuker's properties are paraphrased as follows:

Property 1. *The complexity measure should not rate all programs as equally complex. Formally, there are programs P and Q for which* $M(P) \neq M(Q)$.

Property 2. *There are only finitely many programs of a given complexity. Formally, if c is a non-negative number, there are only finitely many programs P for which* $M(P) = c$.

Property 3. *There exist two different programs of the same complexity. Formally, there are distinct programs P and Q such that* $M(P) = M(Q)$.

Property 4. *Two different programs which have the same functionality need not have the same complexity. Formally, there are functionally equivalent programs P and Q such that* $M(P) \neq M(Q)$.

Property 5. *The complexity of a program segment should be less than or equal to the complexity of the whole program. Formally, for all programs P and Q, the following must hold:* $M(P) \leqslant M(P+Q)$ *and* $M(Q) \leqslant M(P+Q)$.

Property 6. *The resulting complexity of the composition of two programs P and R is not necessarily the same as the composition of programs Q and R, even though P and Q have the same complexity. Formally, there exist programs P, Q, and R such that* $M(P) = M(Q)$ *and* $M(P+R) \neq M(Q+R)$.

Property 7. *If the statements within a program are permutated, the complexity of the resulting program is not necessarily equal to the complexity of the original program. Formally, there are programs P and Q such that Q is formed by permuting the order of the statements of P and* $M(P) \neq M(Q)$.

Property 8. *Renaming has no effect on the measure. Formally, if P is a renaming of Q, then* $M(P) = M(Q)$.

Property 9. *The complexity of the composition of two programs may be greater than the sum of the complexities of the two taken separately. Formally, there exist programs P and Q such that* $M(P) + M(Q) < M(P+Q)$.

5.1. Evaluation of the proposed metrics

Although Weyuker's properties were originally proposed to evaluate software metrics, we analyze the applicability of these properties in the context of ontology and analytically evaluate our proposed metrics against them. Of Weyuker's nine properties, six will be dealt with only briefly here as the proof is obvious.

Obviously, for two ontologies *P* and *Q*, they could contain different sets of classes and properties, resulting in different graph representations and different measurement values for *SOV, ENR, TIP, EOG, NOC, DIT, CID* and *COD*, therefore, Property 1 is satisfied by all metrics. For an application domain, there are only a finitely many number of classes and properties, therefore there are only a finitely many number of ontologies with the same measurement values, so Property 2 is met by all metrics. It is always possible that two different ontologies having the same size or the same graphical structure, therefore satisfying Property 3. Concepts in a domain could be designed/organized in different ways (e.g., see the normalization examples as illustrated by Rector, 2002), leading to different measurement values, therefore Property 4 is satisfied.

The original intent of Property 7 was to ensure that the measurement value changes with the permutation of statements in programs. This property pertains to traditional programming languages. As ontology languages RDFS and OWL are declarative languages, the change in the order of the elements does not affect the semantics of the ontology nor its graphical representation. Hence, the measurement values are not affected. Therefore, Property 7 is not satisfied as it is not applicable to ontologies.

The eighth property states that when the name of the ontology (or its elements) changes, the metric value should remain unchanged. As all proposed metrics are independent of the name of the ontology (or elements), they also satisfy this property.

For each of the metrics, we will provide a detailed analysis for the remaining three properties (Properties 5, 6 and 9) below.

5.1.1. Evaluation of ontology-level metrics

SOV. Let *P* and *Q* be two ontologies, which have the amount of vocabulary *p* and *q*, respectively (i.e., $M(P) = p$ and $M(Q) = q$). Combining *P* and *Q* will yield a single ontology $(P+Q)$ with vocabulary size $p + q - \alpha$, where α is the amount of vocabulary *P* and *Q* have in common. Clearly $0 \leqslant \alpha \leqslant p$ and $0 \leqslant \alpha \leqslant q$, therefore, $M(P) \leqslant M(P+Q)$ and $M(Q) \leqslant M(P+Q)$, satisfying Property 5. As $\alpha \geqslant 0, M(P) + M(Q) \geqslant M(P+Q)$ for any *P* and *Q*, thus Property 9 is not satisfied.

ENR. Let *P* and *Q* be two ontologies, *P* has n_p nodes and e_p edges, and *Q* has n_q nodes and e_q edges, then $M(P) = e_p/n_p$ and $M(Q) = e_q/n_q$. Assuming *P* and *Q* are composed at the root node and there is no further overlapping between *P* and *Q*, then $M(P+Q) = (e_p + e_q)/(n_q + n_p - 1)$, and also assume that $n_q \gg 1$ and $n_p \gg 1$, thus: $M(P+Q) \approx (e_p + e_q)/(n_q + n_p)$.

Again assuming $M(P) \leqslant M(Q)$, i.e., $e_p/n_p \leqslant e_q/n_q$, we can infer that: $e_p n_q \leqslant e_q n_p \Rightarrow e_p n_q + e_q n_q \leqslant e_q n_p + e_q n_q \Rightarrow ((e_p + e_q)/(n_q + n_p)) \leqslant (e_q/n_q) \Rightarrow M(P+Q) \leqslant M(Q)$ therefore, Property 5 is not satisfied.

Let *P* and *Q* be two ontologies with the same *ENR* values (i.e., $e_p/n_p = e_q, n_q$). For *P* and *Q*, there exists an ontology *R* such that it has α number of nodes and β number of edges in common with *P* and *Q*, respectively. Let *R* has n_r nodes and e_r edges, therefore $(e_p + e_r - \beta)/(n_p + n_r - \alpha) \neq (e_q + e_r - \beta)/(n_q + n_r - \alpha)$ if $e_p \neq e_q$, therefore $M(P+R) \neq M(Q+R)$, satisfying Property 6. For Property 9, it can be proved that $((e_p + e_q)/(n_q + n_p)) < (e_q/n_q + e_p/n_p)$, therefore $M(P+Q) < M(P) + M(Q)$ for any *P* and *Q*, hence Property 9 is not satisfied.

TIP. Let *P* and *Q* be two ontologies with *TIP* values *p* and *q*, respectively. Combining *P* and *Q* will yield a single inheritance hierarchy, which has *TIP* value $p + q - \alpha$, where α = (the number of overlapping edges – the number of overlapping nodes + 1). Clearly, $0 \leqslant \alpha \leqslant p$ and $0 \leqslant \alpha \leqslant q$, therefore, $M(P) \leqslant M(P+Q)$ and

$M(Q) \leqslant M(P+Q)$ for any P and Q, satisfying Property 5. As $\alpha \geqslant 0, M(P)+M(Q) \geqslant M(P+Q)$, thus Property 9 is not satisfied.

Let P and Q be two ontologies and $M(P)=M(Q)=n$, there exists an ontology R that overlaps P and Q. The number of overlapping nodes could be different, resulting different hierarchical structures after composition, therefore $M(P+R) \neq M(Q+R)$ and Property 6 is satisfied.

EOG. Let P and Q be two ontologies. Assume both P and Q contain three nodes n_1, n_2, and n_3. n_1 and n_2 have α degrees in P and β degrees in Q ($\alpha \neq \beta$). n_3 has β degrees in P and α degrees in Q. Therefore, $M(P)=M(Q)$ as P and Q have the same degree distribution pattern. Clearly after composing P and Q, the resulting ontology may have $\alpha+\beta$ degrees for each of the three nodes, making $M(P+Q)=0$, thus $M(P+Q)<M(P)$ and $M(P+Q)<M(Q)$, Property 5 is not satisfied. Now, considering an ontology R which contains nodes n_1, n_2 and n_3 with degrees 1, 1, and k, respectively. After composing P and R, the resulting ontology may have $k+1$ degrees for each of the three nodes. After composing Q and R, the resulting ontology may have three nodes with degrees 2, 2, and $k+1$, respectively. Clearly, $M(P+R) \neq M(Q+R)$ even though $M(P)=M(Q)$, satisfying Property 6.

Let P and Q be two ontologies. Assume P contains three nodes n_1, n_2, and n_3 with the same degree k, and Q contains two nodes n_1 and n_3 with the same degree 1. Therefore, $M(P)=M(Q)=0$. After composing P and Q, the resulting ontology may have three nodes n_1, n_2, and n_3 with degrees $k+1, k, k+1$, respectively. Therefore, $M(P)+M(Q)<M(P+Q)$, satisfying Property 9.

5.1.2. Evaluation of class-level metrics

NOC. The NOC metric proposed in this paper is the same as the CK NOC metric, which is proved to satisfy Properties 5 and 6. Property 9 is not satisfied. We refer the reader to Zuse (1991) for the detailed analytical evaluation of this metric.

DIT. The DIT metric proposed in this paper is the same as the CK DIT metric, which is proved to satisfy Properties 5 and 6. Property 9 is not satisfied. We refer the reader to Zuse (1991) for the detailed analytical evaluation of this metric.

CID. Let P and Q be two class in an ontology, which have the number of in-degrees p and q, respectively (i.e., $M(P)=p$ and $M(Q)=q$). Combining P and Q will yield a single class ($P+Q$) with in-degree $p+q-\alpha$, where α is the number of incoming edges that P and Q have in common. Clearly $0 \leqslant \alpha \leqslant p$ and $0 \leqslant \alpha \leqslant q$, therefore, $M(P) \leqslant M(P+Q)$ and $M(Q) \leqslant M(P+Q)$, satisfying Property 5. As $\alpha \geqslant 0, M(P)+M(Q) \geqslant M(P+Q)$ for any P and Q, thus Property 9 is not satisfied.

Let P and Q be two classes and $M(P)=M(Q)=n$, there exists a class R such that it has α number of incoming edges in common with P and β in common with Q, where $\alpha \neq \beta$. Let $M(R)=r$, then $M(P+R)=n+r-\alpha$ and $M(Q+R)=n+r-\beta$, therefore $M(P+R) \neq M(Q+R)$ and Property 6 is satisfied.

COD. Let P and Q be two classes in an ontology, which have the number of out-degrees p and q, respectively (i.e., $M(P)=p$ and $M(Q)=q$). Following the reasoning for the *CID* metric above, we can prove that for *COD*, the Properties 5 and 6 are satisfied, and the Property 9 is not satisfied.

5.2. Summary of the analytical evaluation

Table 1 shows the summary of analytical evaluation results. All the metrics satisfy the majority of the properties presented by Weyuker, with the two strong exceptions Properties 7 and 9. As discussed previously, Property 7 is not applicable to ontology metrics due to the declarative nature of ontology languages.

Weyuker's Property 9 is not satisfied by seven metrics (all except *EOG*). Property 9 implies that the interactions between programs can increase complexity. This is the property about which many researchers have raised questions (Gursaran and Roy, 2001; Zhang and Xie, 2002). In our study, failing to satisfying this property by the seven metrics indicates that unlike programs, the complexity of ontologies/classes is reduced after individual ontologies/classes are composed.

Table 1
The summary of analytical evaluation results.

Prop.	Ontology-level metrics				Class-level metrics			
	SOV	ENR	TIP	EOG	NOC	DIT	CID	COD
1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Yes	No	Yes	No	Yes	Yes	Yes	Yes
6	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
7	No	No	No	No	No	No	No	No
8	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9	No	No	No	Yes	No	No	No	No

Other violations of Weyuker's properties are in the cases of *ENR* and *EOG* on Property 5. Property 5 implies that the complexity should be increased monotonically. The *ENR* and *EOG* measures show that the complexity is subject to change during the development of an ontology, which is reasonable when we view the complexity in terms of structure, instead of size. Interestingly, other researchers also observed the exceptions in applying this property (Zuse, 1991).

6. Empirical evaluation of the complexity metrics

We have applied the proposed metrics to measure a set of real-world ontologies collected from online sources (as shown in Table 2). The Jambalaya[12] tool was used to visualize the graphical representation of the ontology. As an example, Fig. 5 below shows the ontology graph for the Full-Galen ontology (Rector et al., 1993), which describes comprehensive anatomy and drug related terms.

To facilitate automated data collection, we have also developed a metric tool based on the Protégé-OWL[13] Java API. The tool we developed traverses the graph of each ontology, collects and stores relevant information and finally calculates the metrics.[14] In this section, we briefly discuss some observations from the empirical analysis.

6.1. Evaluation of ontology-level metrics

Table 3 shows the measurement values for the collected ontologies. The numbers marked with (*) indicate the largest measurement values among the studied ontologies.

The *SOV* values range from 52 (the Amino-acid ontology) to 134K (the Go_daily-termdb ontology), showing different amounts of vocabulary used. Although the Amino-acid ontology has the smallest amount of vocabulary, it has the highest edge-node density (*ENR*) (about 3 edges associated with one node), therefore is more complex when *ENR* is considered. The high *ENR* value also indicates that further modularization is needed to ease the understanding and maintenance efforts. The empirical results also show that some ontologies are designed with strict single inheritance (with *TIP* = 0), while others adopt multiple inheritance and their inheritance hierarchy deviates heavily from a pure tree structure (e.g., *TIP* = 33136 for the NCI Thesaurus ontology). The ontology

[12] http://www.thechiselgroup.org/jambalaya.
[13] http://protege.stanford.edu/overview/protege-owl.html.
[14] Available for download at:http://www.itee.uq.edu.au/liyf/ontComplexity.

Table 2
The descriptions of the collected ontologies.

Ontology	Description	Version	Size (KB)
AllMonet	An ontology for mathematical algorithms	2005-06-07	756
Amino-acid	An ontology about amino acids	1.2	91
Biopax-level2	Biological pathways exchange language	1.0	118
Bio-zen	An ontology for life sciences	1.01	188
CL	An ontology for cell types	1.26	784
Full-Galen	The full GALEN ontology of medical terms, anatomy and drugs translated into OWL	2006-09-12	20100
Go_daily-termdb	The Gene Ontology project, which provides a controlled vocabulary to describe gene and gene product attributes in any organism	4.270	39200
MGED	An ontology for microarray experiments in support of MAGE v.1	1.3.1.1	556
nciOntology	National Cancer Institute's ontology of cancer	03.09d	32800
NCI Thesaurus	NCI Thesaurus, a controlled vocabulary in support of NCI administrative and scientific activities	07.04e	81500
NMR	NMR-instrument specific component of metabolomics investigations	0.1	124
OBI	Ontology for Biomedical Investigations	0.6.7	104
Parkinsons disease	An ontology about Parkinsons disease	n/a	13
Po	Protein ontology	2.0	114
ProPreO	A comprehensive Proteomics data and process provenance ontology	0.5	229
SBO	The Systems Biology Ontology, a controlled vocabulary tailored specifically for Systems Biology problems, especially in the context of computational modeling	2006-10-15	247
Semweb-glossary	A glossary of information about the web	2007-10-17	426
Snpontology_full	A domain representation of genomic variations	1.4	67
STC	An ontology about space-time Coordinates	1.3	188
Swpatho2	Semantic Web for Pathology	2006-02-20	166
tambis	A biological science ontology developed by the TAMBIS project	2006-09-27	214

Table 3
The measurement values of ontology-level metrics

Ontology	SOV	ENR	TIP	EOG
AllMonet	2065	1.01	52	1.47
Amino-acid	52	**3.10**(*)	200	1.56
Biopax-level2	111	1.48	69	2.54
Bio-zen	263	1.29	89	**2.74**(*)
CL	2616	1.23	453	1.49
Full-Galen	24092	1.68	12414	2.21
Go_daily-termdb	**134142**(*)	1.13	13981	0.97
MGED	1029	1.35	226	2.29
nciOntology	27723	1.79	19165	2.42
NCI Thesaurus	58741	1.79	**33136**(*)	2.13
NMR	326	1.05	0	1.62
OBI	234	1.07	0	1.85
Parkinsons-Disease	86	0.74	0	1.86
po	211	0.92	62	2.37
ProPreO	478	1.38	144	2.37
SBO	331	1.15	46	2.14
Semweb-glossary	1805	1.35	632	2.27
snpontology_full	123	1.12	24	2.47
STC	549	0.79	39	1.74
Swpatho2	628	1.21	100	1.69
tambis	493	1.44	109	1.92

Go_daily-termdb has the most regular structure as it has the smallest value of *EOG*. While the structure of Bio-zen ontology is most irregular, therefore it is "more complex" if *EOG* is considered. In summary, the measurement values in Table 3 show that the ontologies may be "more complex" in one aspect but "less complex" in other aspects. Through the measurement, we can achieve a more complete understanding of the complexity of theses ontologies.

Moreover, note an interesting comparison between the NCI Thesaurus ontology and the Go_daily-termdb ontology. The former is more than double in terms of file size than the latter. However, the latter has the largest *SOV* (twice that of the former) among all the ontologies we evaluated. This reiterates our point that a single metric is not sufficient to analyze the complexity of ontologies.

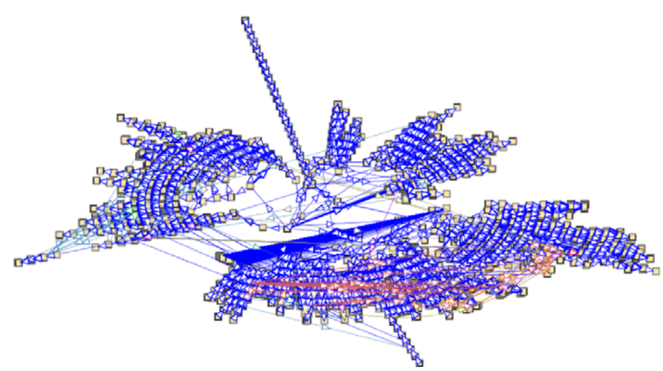

Fig. 5. The ontology graph of the Full-Galen ontology.

6.2. Evaluation of class-level metrics

Table 4 shows the collected data for the class-level metrics. For all metrics, the minimum measurement value is 0 so we omit the Minimum column in Table 4. The maximum measurement values are much larger than the median (Med) and third-quartile (Q3) values, suggesting that the distributions of the metric data (except *DIT*) are highly skewed – that most of the ontology classes have small measurement values with a few have large values. For *NOC*, this means that the classes in general have few immediate children and only a small number of classes have many immediate subclasses. For *CID* and *COD*, the skewed distributions mean that many classes refer to (or are referred to by) a few other classes, while a small number of classes refer to (or are referred to by) a large number of other classes. As an example, Table 5 shows the top 10 classes in the Full-galen ontology that have the largest *CID* and *COD* values.

The classes with large *CID* values indicate that more classes are depending on them, therefore special cares need to be taken when

Table 4
The descriptive statistics of the metrics data at the class-level.

Ontology	NOC			DIT			CID			COD		
	Med	Q3	Max	Med	Q3	Max	Med	Q3	Max	Med	Q3	Max
AllMonet	0	0	154	3	5	8	0	0	154	1	1	4
Amino-acid	0	1	20	3	3	3	2	11	38	2	18.5	30
Biopax-level2	0	1	11	3	3.5	6	0	2	11	4	6	13
Bio-zen	0	1	19	6	7	9	1	2	19	1.5	3	12
CL	0	1.25	67	8	10	15	0	2	67	1	2	6
Full-Galen	0	1	1492	9	12	25	1	2	1694	1	2	95
Go_daily-termdb	0	1	1121	6	8	15	0	2	1121	1	2	7
MGED	0	0	31	5	7	9	1	1	32	1	3	12
nciOntology	0	1	2761	8	10	19	0	1	2761	1	2	24
NCI Thesaurus	0	1	2633	7	9	17	0	2	3390	1	2	31
NMR	0	0	22	6	6	8	0	1	22	1	1	5
OBI	0	1	38	4	6	9	0	2	38	1	1	5
Parkinsons-Disease	0	1	7	3	5	9	0	1	7	1	1	1
po	0	1	8	4	4	6	1	3	13	3	4.5	8
ProPreO	0	2	25	6	7	8	1	2	27	1	2	5
SBO	0	2	18	5	6.75	8	0	2	18	1	1	3
Semweb-glossary	0	1	180	12	14	19	0	1	180	1	2	5
snpontology_ full	0	1	12	2	3	4	0	2	13	1	2	6
STC	0	1	36	3	4	7	0	1	36	1	1	7
Swpatho2	0	0	30	1	2	4	0	0	40	1	1	3
tambis	1	1	51	3	4	8	1	3	162	1	2	14

Table 5
The top 10 Full-Galen classes that have largest CID and COD values.

CID	NAMEDActiveDrugIngredient, BodyStructure, pathological, Device, Level, Pathological Phenomenon, nonNormal, mirrorImaged, ClinicalAct,InflammationLesion
COD	AbdominalCavity, Face, Skull, Neck, CranialCavity, Knee, PelvicCavity, Thigh, Heart, Orbit

changes to these classes are made as the changes may be propagated to a large proportion of the ontology. The classes with large *COD* values indicate that more inputs from other classes are required to understand these classes, therefore more learning and maintenance efforts need to be allocated.

For the *DIT*, the median values range from 1 to 12, which are close to the Q3 and max values. These values represent a relatively less skewed distribution, indicating good adoption of taxonomy principles. The ontologies are designed into hierarchies via the inheritance relationship.

The metric data of *NOC, DIT, CID* and *COD* can help ontology engineers identify potential problematic areas. For example, in the ontology "nciOntology", we find that one class (the Chemotherapy_Regimen class) has *NOC* value 2761. A closer look at the class shows that none of its 2761 subclasses has subclasses of its own. This suggests that the designer may not be following good classification principles when designing these classes. These classes should be further examined during review meetings, which may lead to possible re-design/re-structure of the ontology.

To further analyze the distribution of metric data in large-scale ontologies, we sort the classes in descending order (from the largest value to the smallest value), and plot the value against its rank on a log–log diagram. As an example, Fig. 6 below shows the distributions of the *NOC, DIT, CID*, and *COD* data for the Full-Galen ontology. We can see that all distributions (except *DIT*) have "long tails", indicating most of the data have smaller values. The distributions of *NOC, CID* and *COD* form a straight line in log–log diagrams, indicating the power law behavior (Newman, 2005), that is, the distribution of the metric data can be described by the following equation:

$$f = Cr^{-a}$$

where f is the measurement value of a class, r is the rank of the class when the values are sorted in descendant order, C is a constant and a is the exponent of the power law. Taking the logarithm on both sides of the above equation, we get:

$$ln(f) = ln(C) - a ln(r)$$

So a power law distribution is seen as a straight line on a log–log plot. The slope of the line is $-a$ and the intercept is $ln(C)$. Table 6 shows the power law parameters for the Full-Galen, Go_daily-termdb and NCI Thesaurus ontologies. The corresponding R^2 values range from 0.74 to 0.96, indicating good fitness of the data (with significance value 0.000).

Power law is a universal law that is behind many natural and social phenomena, such as earthquake magnitudes, income distribution and word frequencies in a text (Barabási and Bonabeau, 2003). Our study shows that some properties of large ontologies also follow power law distribution. One possible explanation of this behavior is the "Preferential Attachment" principle (Barabási and Albert, 1999), which says that new nodes are more likely to be attached to the more "important" nodes that already have many attachments. This principle also applies to ontology construction, as new knowledge is often derived from existing knowledge that are well understood and established, thus generating the "scale-free" ontology graph. More implications of the power law behavior in ontology development remain to be studied.

7. Possible applications of the metrics to ontology engineering

Using the four proposed ontology-level metrics, ontology engineers could achieve a better understanding of the overall complexity of the ontology. For example, by examining the *SOV* and *ENR* values, the ontology engineers can check if the ontology under development is too large and if further modularization is needed. By means of the *TIP* metric, the ontology engineer can check if the design of the ontology follows good classification (or object-oriented) principles. Through the *EOG* metric, the

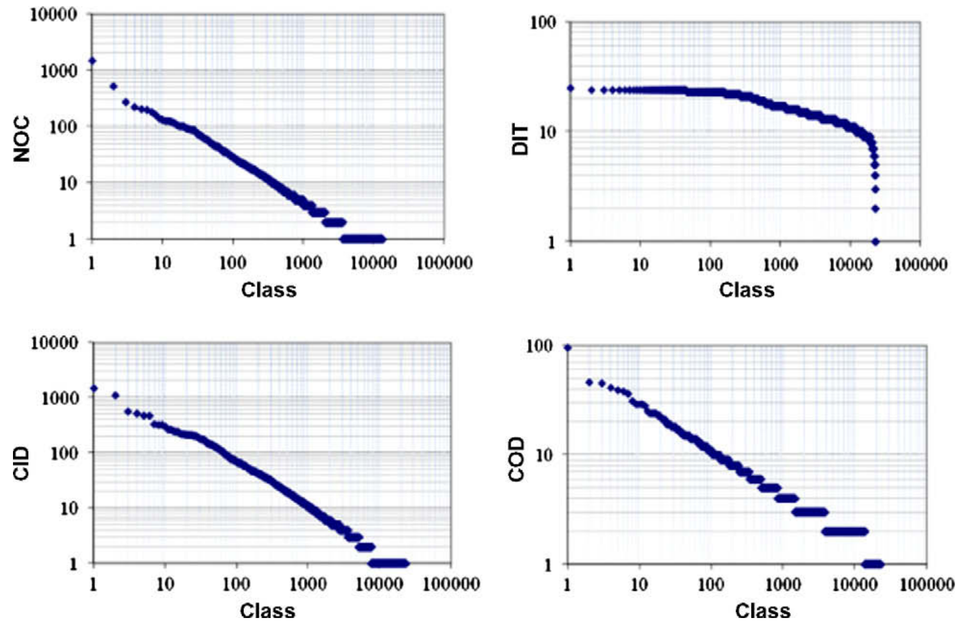

Fig. 6. The distribution of metrics data for the Full-Galen ontology (classes are sorted by measurement values).

Table 6
The power law parameters of the three large ontologies.

		a	*C*	R^2	Std. Error	Significance
Full-Galen	*NOC*	0.67	414.29	0.90	0.22	0.000
	DIT	–	–	–	–	–
	CID	0.91	5619.80	0.96	0.18	0.000
	COD	0.46	111.65	0.85	0.20	0.000
Go_daily-termdb	*NOC*	0.80	1480.30	0.95	0.18	0.000
	DIT	–	–	–	–	–
	CID	0.79	1551.54	0.95	0.19	0.000
	COD	0.35	34.78	0.74	0.21	0.000
NCI Thesaurus	*NOC*	0.88	4798.22	0.95	0.21	0.000
	DIT	–	–	–	–	–
	CID	1.02	28254.27	0.94	0.26	0.000
	COD	0.49	194.81	0.88	0.18	0.000

ontology engineer can check how regular (or irregular) an ontology's structure is.

Using the four proposed class-level metrics, ontology engineers could have better insights of the quality of the internal design of the ontology. By means of *NOC* and *DIT*, they could check whether the internal design is a good decomposition of the problem space. The power law distribution of *CID* and *COD* means that the ontology graph is very inhomogeneous. By using *CID* and *COD*, ontology engineers could understand the inter-connectivity among the classes, and identify the more "important" classes that require more attentions.

The proposed metrics could also be useful for project managers who may not be able to review the detailed ontology design materials. The metrics could serve as "indicators" of the ontology quality, helping managers understand the development status, gain an overall picture of ontology complexity, and identify potential problematic areas. The managers could then have a better control of the ontology development process, which may involve changes in ontology design and project schedule.

Very often there are many possible designs for an ontology. The proposed metrics can be applied to evaluate different design alternatives. For example, Rector (Rector, 2002; Rector, 2003) observes that many large ontologies use existing classifications that are usually tangled and heterogenous, often mixing subsumption and partonomy. Rector then suggests a normalized ontology design, which decomposes an ontology into independent disjoint homogeneous taxonomies (each taxonomy has a tree structure). By using our metrics, we find that the tangled version has a smaller set of vocabulary, shorter inheritance tree, and less diverse structure than the normalized version. From these points of view, the normalization leads to a more complex design. However, the tangled version has higher edge density and is more deviated from the pure tree structure; therefore the normalization leads to a "conceptually clearer" design if *ENR* and *TIP* are considered. By using the proposed metrics, we can achieve a better understanding of an ontology design and an improved decision-making process.

8. Conclusions

Semantic Web ontologies are believed to be an ideal candidate for representing domain knowledge because of their wide adoption and formal semantics (Smith et al., 2007; Gardner, 2005; Searls, 2005; Ruttenberg et al., 2009). With the proliferation of Semantic Web technologies, more and more large ontologies are being developed in a wide variety of domains. As design complexity has impact on human understanding, measuring the complexity of ontologies has become a very important task for ontology development, maintenance, and reuse. In this paper, we have proposed a suite of metrics for ontology measurement, including ontology-level metrics (*SOV, ENR, TIP, EOG*) and class-level metrics (*NOC, DIT, CID, COD*). We evaluated the proposed metrics analytically using Weyuker's criteria and empirically using data collected from large, public ontologies. The analytical evaluation shows that the proposed metrics satisfy most of Weyuker's properties, and the empirical evaluation shows that the proposed metrics can differentiate ontologies with distinct degrees of complexity. The evaluation results confirm that the proposed metrics can be applied in practices to evaluate the design complexity of ontologies. In this paper, we also discuss the possible applications of the proposed metrics to ontology quality control.

In the future, we plan to apply the proposed metrics to the development of large-scale ontologies in practices, and to collect data to investigate how the metrics can be used to improve the process of ontology engineering. We will then further refine/enhance the proposed metrics. We also plan to further investigate the relationship between the complexity of an ontology and its quality. We assume that more complex ontologies are harder to maintain and are more defect-prone, therefore more quality assurance (QA) and maintenance efforts are needed. An empirical study on the correlations between the proposed metrics and ontology reliability and maintainability is an important future research direction. Moreover, theoretical and empirical research are also required to identify exactly how each metric is associated with increased cognitive complexity.

Acknowledgement

This research is supported by the Chinese NSF grant 60703060, and the state 863 project 2007AA01Z480 and 2007AA01Z122. We also acknowledge the support from the MOE Key Laboratory of High Confidence Software Technologies at Peking University. We thank Jeff Pan at the University of Aberdeen and Jane Hunter at the University of Queensland for their valuable comments. We would also like to thank the anonymous reviewers for their detailed and insightful comments.

References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z., 2008. DBpedia: a nucleus for a web of open data. In: Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007), 2008, pp. 722–735. URL <http://dx.doi.org/10.1007/978-3-540-76298-0_52>.

Barabási, A.-L., Albert, R., 1999. Emergence of scaling in random networks. Science 286 (5439), 509–512. URL: <http://view.ncbi.nlm.nih.gov/pubmed/10521342>.

Barabási, A.-L., Bonabeau, E., 2003. Scale-free networks. Scientific American 288, 50–59.

Basili, V.R., Briand, L.C., Melo, W.L., 1996. A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering 22 (10), 751–761. <http://dx.doi.org/10.1109/32.544352>.

Berners-Lee, T., Hendler, J., Lassila, O., 2001. The semantic web. Scientific American 284 (5), 35–43.

Brickley, D., Guha, R.V. (Eds.), 2004. Resource Description Framework (RDF) Schema Specification 1.0, (Feb. 2004). <http://www.w3.org/TR/rdf-schema/>.

Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P., 2005. A semiotic metrics suite for assessing the quality of ontologies. Data Knowledge Engineering 55 (1), 84–102.

Cherniavsky, J.C., Smith, C.H., 1991. On Weyuker's axioms for software complexity measures. IEEE Transactions on Software Engineering 17 (6), 636–638. http://dx.doi.org/10.1109/32.87287.

Chidamber, S., Kemerer, C., 1994. A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20 (6), 476–493. <http://doi.ieeecomputersociety.org/10.1109/32.295895>.

DeMarco, T., 1986. Controlling Software Projects: Management, Measurement, and Estimates. Prentice-Hall, PTR, Upper Saddle River, NJ, USA.

Fenton, N., Melton, A., 1990. Deriving structurally based software measures. Journal of Systems and Software 12 (3), 177–187. <http://dx.doi.org/10.1016/0164-1212(90)90038-N>.

Fenton, N., Pfleeger, S.L., 1998. Software Metrics: A Rigorous and Practical Approach. PWS Publishing Co., Boston, MA, USA.

Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J., 2006. Modelling ontology evaluation and validation. In: Proceedings of the 3rd European Semantic Web Conference (ESWC'06), Budva, Montenegro, pp. 140–154.

Gardner, S.P., 2005. Ontologies and semantic data integration. Drug Discovery Today 10 (14), 1001–1007. doi:10.1016/S1359-6446(05)03504-X.

Gruber, T., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5 (2), 199–220.

Gursaran, Roy, G., 2001. On the applicability of Weyuker Property 9 to object-oriented structural inheritance complexity metrics. IEEE Transactions on Software Engineering 27 (4), 381–384. <http://dx.doi.org/10.1109/32.917526>.

Harrison, W., 1992. An entropy-based measure of software complexity. IEEE Transactions on Software Engineering 18 (11), 1025–1029. <http://dx.doi.org/10.1109/32.177371>.

Horrocks, I., Patel-Schneider, P.F., van Harmelen, F., 2003. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. Journal of Web Semantics 1 (1), 7–26.

Kang, D., Xu, B., Lu, J., Chu, W.C., 2004. A complexity measure for ontology based on UML. In: Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04), IEEE CS, Washington, DC, USA, pp. 222–228.

Koru, A.G., Tian, J., 2003. An empirical comparison and characterization of high defect and high complexity modules. Journal of Systems and Software 67 (3), 153–163. <http://dx.doi.org/10.1016/S0164-1212(02)00126-7>.

Law, K.S., Wong, C.-S., Mobley, W.H., 1998. Toward a taxonomy of multidimensional constructs. The Academy of Management Review 23 (4), 741–755. URL: <http://www.jstor.org/stable/259060>.

Li, H.F., Cheung, W.K., 1987. An empirical study of software metrics. IEEE Transactions on Software Engineering 13 (6), 697–708. <http://dx.doi.org/10.1109/TSE.1987.233475>.

Lindsay, P.H., Norman, D.A., 1977. Human Information Processing: An Introduction to Psychology, second ed. Academic Press, New York. URL: <http://millenium.itesm.mx/record= i143498&searchscope=0>.

McCabe, T.J., 1976. A complexity measure. IEEE Transactions on Software Engineering 2 (4), 308–320.

Miller, G.A., 1956. The magical number seven plus or minus two: some limits on our capacity for processing information. The Psychological Review 63, 81–97. URL: <http://users.ecs.soton.ac.>, <http://uk/harnad/Papers/Py104/Mille r/miller.html>.

Newman, M., 2005. Power laws pareto distributions and Zipf's law. Contemporary Physics 46 (5), 323–351.

Opdahl, A.L., Henderson-Sellers, B., 2001. Grounding the OML metamodel in ontology. Journal of Systems and Software 57 (2), 119–143. http://dx.doi.org/10.1016/S0164-1212(00)00123-0.

Rector, A.L., 2002. Normalisation of ontology implementations: towards modularity, re-use, and maintainability. In: Proceedings of the Workshop on Ontologies for Multiagent Systems (OMAS), Siguenza, Spain.

Rector, A.L., 2003. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In: K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture, ACM Press, New York, USA, pp. 121–128. <http://doi.acm.org/10.1145/945645.945664>.

Rector, A.L., Nowlan, W., Glowinski, A., 1993. Goals for concept representation in the GALEN project. In: Proceedings of the 17th Annual Symposium on Computer Applications in Medical Care (SCAMC'93), pp. 414–418.

Rector, A.L., Noy, N., Drummond, N., Musen, M., 2005. Ontology design patterns and problems: practical ontology engineering using Protégé-OWL. In: Proceedings of the 4th International Semantic Web Conference (ISWC'05). Springer-Verlag, Galway, Ireland (tutorial notes).

Ruttenberg, A., Rees, J., Samwald, M., Marshall, M.S., 2009. Life sciences on the semantic web: the neurocommons and beyond. Briefings in Bioinformatics 10 (2), 193–204. doi:10.1093/bib/bbp004.

Searls, D.B., 2005. Data integration: challenges for drug discovery. Nature Reviews Drug Discovery 4 (1), 45–58. doi:10.1038/nrd1608.

Sheridan, T., 1980. Mental workload – What is it? Why bother with it? Human Factors Society Bulletin 23 (2), 1–2.

Simon, H.A., 1974. How big is a chunk? Science 183 (4124), 482–488.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R.H., Shah, N., Whetzel, P.L., Lewis, S., 2007. The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25 (11), 1251–1255. doi:10.1038/nbt1346.

Subramanyam, R., Krishnan, M.S., 2003. Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects. IEEE Transactions on Software Engineering 29 (4), 297–310. <http://dx.doi.org/10.1109/TSE.2003.1191795>.

Vrandečić, D., Sure, Y., 2007. How to design better ontology metrics. In: ESWC '07: Proceedings of the 4th European conference on The Semantic Web. Springer-Verlag, Innsbruck, Austria, pp. 311–325. <http://dx.doi.org/10.1007/978-3-540-72667-8_23>.

Wang, T.D., Parsia, B., Hendler, J.A., 2006. A survey of the web ontology landscape. In: International Semantic Web Conference, Lecture Notes in Computer Science, vol. 4273. Springer, pp. 682–694. doi:10.1007/11926078_49. URL: <http://dx.doi.org/10.1007/11926078_49>.

Weyuker, E.J., 1988. Evaluating software complexity measures. IEEE Transactions on Software Engineering 14 (9), 1357–1365. <http://dx.doi.org/10.1109/32.6178>.

Wilde, N., Matthews, P., Huitt, R., 1993. Maintaining object-oriented software. IEEE Software 10 (1), 75–80. <http://dx.doi.org/10.1109/52.207232>.

Yao, H., Orme, A.M., Etzkorn, L., 2005. Cohesion metrics for ontology design and application. Journal of Computer Science 1 (1), 107–113.

Zhang, L., Xie, D., 2002. Comments on the applicability of Weyuker Property 9 to object-oriented structural inheritance complexity metrics. IEEE Transactions on Software Engineering 28 (5), 526–527. <http://dx.doi.org/10.1109/TSE.2002.1000454>.

Zhang, H., Zhang, X., Gu, M., 2007. Predicting defective software components from code complexity measures. In: PRDC '07: Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, IEEE Computer Society, Washington, DC, USA, pp. 93–96. <http://dx.doi.org/10.1109/PRDC.2007.56>.

Zuse, H., 1991. Software Complexity: Measures and Methods. Walter de Gruyter & Co., Hawthorne, NJ, USA.

Hongyu Zhang received the PhD degree in computer science from the School of Computing, National University of Singapore in 2003, the MS degree in communication and networks from Nanyang Technological University, Singapore in 1998. He is currently an Associate Professor at the School of Software, Tsinghua University, Beijing, China. Before joining Tsinghua, he was a lecturer at the School of Computer Science and Information Technology, RMIT University, Australia. His research is mainly in the area of software engineering, in particular, software metrics, software quality, and software reuse. He has published more than 40 research papers in international journals and conferences proceedings.

Yuan-Fang Li is a research fellow at the School of ITEE, the University of Queensland, Australia. He received both his Bachelor of Computing (with honors) and PhD's degrees from National University of Singapore. His main research interests include the Semantic Web, ontology languages, semantic querying and inference, large-scale information processing, information visualization and formal methods.

Hee Beng Kuan Tan received his B.Sc.(First Class Hons) in Mathematics in 1974 from the Nanyang University (Singapore). He received his M.Sc. and Ph.D. degrees in Computer Science from the National University of Singapore in 1989 and 1996 respectively. He has thirteen years of experience in IT industry before moving to academic. He is currently an Associate Professor with the Division of Information Engineering in the School of Electrical and Electronic Engineering, Nanyang Technological University. His current research interest is in software testing and analysis, and software security.