JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2007; **19**:49–75
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/smr.341

Research

Indicating ontology data quality, stability, and completeness throughout ontology evolution

Anthony M. Orme[1,*,†], Haining Yao[2] and Letha H. Etzkorn[3]

[1]*Mathematics and Computer Science, Athens State University, Athens, AL 35611, U.S.A.*
[2]*National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, U.S.A.*
[3]*Computer Science, University of Alabama in Huntsville, Huntsville, AL 35899, U.S.A.*

SUMMARY

Many application areas today make use of ontologies. With the advent of semantic Web technologies, ontology based systems have become widespread. Developing an ontology is part of the necessary early development of an ontology-based system. Since the validity and quality of the ontology data directly affects the validity and quality of the system using the ontology, evolution of the ontology data directly affects the evolution and/or maintenance of the ontology-based systems that depend on and employ the ontology data. Our research examines the quality, completeness, and stability of ontology data as ontologies evolve. We propose a metrics suite, based on standard software quality concepts, to measure the complexity and cohesion of ontology data. First we theoretically validate our metrics. Then we examine empirically whether our metrics determine ontology data quality, by comparing them to human evaluator ratings. We conclude that several of our metrics successfully determine ontology complexity or cohesion. Finally, we examine, over evolving ontology data, whether our metrics determine ontology completeness and stability. We determine that various metrics reflect different kinds of changes. Our experiments indicate our metrics' measure ontology stability and completeness; however, interpretation of specific metrics values and the interaction of different metrics requires further study. Copyright © 2007 John Wiley & Sons, Ltd.

Received 18 April 2006; Revised 15 November 2006; Accepted 15 November 2006

KEY WORDS: ontology completeness metrics; ontology stability metrics; ontology complexity metrics; ontology cohesion metrics

*Correspondence to: Anthony M. Orme, Athens State University, Athens, AL 35611, U.S.A.
†E-mail: tony.orme@athens.edu

1. INTRODUCTION

As the Internet has grown, semantic-Web-based Internet applications such as improved search engines [1,2] and e-commerce [3] have been a driving force for domain-specific ontology development. Ontology-based systems use ontologies to provide a shared semantically enhanced interface for applications. Ontologies are used in intelligent agents and data management [4], and researchers predict additional use in service matching and the dynamic composition of Web services [5,6]. In addition, ontologies are used in text-mining and data-mining applications, such as mining genetic and protein literature in bioinformatics [7,8]. Ontology-based systems are normally developed using similar software lifecycles to those by which other kinds of systems are developed. However, the development of an ontology is part of the necessary early development of an ontology-based system.

Ontology-specific information may be a limiting factor in the development of future applications unless ontology data are prepared correctly prior to use. It is important to examine ontology data for incompleteness, inconsistency, and incorrect values. The stability of each ontology and an analysis of changes to that ontology are also important: as ontology developers update an ontology, it is important to determine whether that ontology is actually improving over time (becoming more complete and correct), or whether changes to the ontology are having a negative effect, making the ontology less cohesive and correct.

Our research focuses on the development of a metrics suite, which measures the complexity and cohesion of ontologies. This metrics suite is compiled from the relationships that exist in the ontology as a whole. In this paper, using our metrics suite, we analyze the changes in the complexity and cohesion of ontologies as the ontologies evolve, through the examination of different ontology releases. We examine ontology stability and discuss how to detect when an ontology is becoming complete, and when an ontology is losing focus on its domain. Until now, the overall quality of the data that make up these ontologies has not been considered in detail, and to the best of our knowledge, analyses of ontology stability, completeness, and domain focus as the ontology evolves have never been done before.

Burton-Jones *et al.* [9] provided a suite of metrics to assess ontology quality. Burton-Jones *et al.* described two different kinds of approaches for analyzing ontology quality: deductive approaches based on a general theory, and inductive approaches that involve empirically testing ontologies to identify characteristics associated with favorable outcomes for an application [9]. Burton-Jones *et al.*'s approach was deductive, based on a semiotic framework derived from linguistics theory that includes general elements of quality based on linguistic-style concepts related to the analysis of signs and symbols [9]. Our approach is inductive, and is based on standard software quality concepts such as complexity and cohesion. Although both Burton-Jones *et al.*'s approach and our approach can be used to analyze ontology quality, we feel that our approach is more suited to analyzing the quality of an ontology that is to be deployed as part of a software system. Also, we feel that our approach can more easily be used in the future to help an application choose between different, similar ontologies at run time. Further, and most importantly, in this paper we analyze the use of ontology metrics to examine ontology quality as the ontology evolves, examining different ontology releases. Burton-Jones *et al.* analyzed a static ontology snapshot.

Our examination of ontology quality across several ontology releases could be considered to be ontology maintenance using some definitions of maintenance. However, using the definitions provided by Chapin *et al.* [10], the term 'evolution' seems more appropriate, since we examined

DOI: 10.1002/smr

public releases of ontologies. However, we note that in many cases the ontology-based systems that employ these ontologies may change their operation as the current release of an ontology changes (because they may always employ the *current* release of the ontology as opposed to a *particular* release of the ontology), but the ontology-based systems may not themselves have a separate public release. Thus, it might be appropriate to discuss ontology-based system 'maintenance' which results from ontology 'evolution'.

The metrics presented in this paper are Number of Properties (NoP), Average Properties per Class (AP-C), Average Fanout per Class (AF-C), Average Fanout of Root Class (AF-R), Average Fanout of Non-Leaf Classes (AF-NL), and Maximum Depth of Inheritance Tree (MaxDIT).

These metrics are validated employing a software metrics theoretical framework. In addition, the metrics are empirically validated in two independent experiments in which the metrics were compared to human evaluators over several different types of ontologies. Potential applications of these metrics are discussed.

The remainder of this paper is organized as follows. Section 2 provides the necessary background to enable an understanding of Web-based ontologies written with the Web Ontology Language (OWL) and DARPA Agent Markup Language (DAML). Section 2 also describes some background in software metrics, and presents a metrics validation framework developed by Kitchenham *et al.* [11]. We validate our set of metrics theoretically (in Section 4) using this validation framework. Section 3 defines our ontology metrics both conceptually and in a formal mathematical notation. In Section 3 we also include example calculations of each metric. The theoretical and empirical validation of the ontology metrics is completed in Section 4 along with a complete summary of our work and results. Section 5 provides conclusions and thoughts for future research in the area of preparing ontology data for ontology-based computing.

2. BACKGROUND

In this section, we first provide a description of OWL: this is necessary to understand the calculation of our metrics. Next we discuss previous work in ontology measurement. Third, we discuss some common object-oriented (OO) metrics that are often used to measure software quality. These are provided since they have some similarities to our ontology-based metrics, and also since these kinds of metrics would most likely be used together with ontology metrics to examine the quality of an ontology-based system (the OO metrics would measure the quality of the software itself, while the ontology-based metrics would measure the quality of the ontology data examined by the software). Finally, we provide a description of the theoretical framework we use to examine our metrics.

2.1. OWL

One of the most recognized definitions of an ontology, as given by Gruber (see [12]), states the following.

'When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. Those sets of objects, and the describable relationship among them, are reflected in the representational vocabulary with which a knowledge-base program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program

DOI: 10.1002/smr

```
<owl:Class rdf:ID="CLASSNAME">
                .
                .
        <rdfs:subClassOf rdf:resource="SOMESUPERCLASS"/>
                .
                .
</owl:Class>
```

Figure 1. OWL class definition.

by defining a set of representational terms. In such ontologies, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrains the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory'.

OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web [13]. OWL has been developed as an extension of RDF (the Resource Description Framework) and is based on the DAML+OIL Web ontology language. OWL is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web-based documents and applications. OWL aids in formalizing a domain by defining classes and asserting properties of those classes. OWL allows reasoning about the classes to a degree that is supported by the formal semantics of the language. The Web Services Ontology (OWL-S), the current standard for describing Web services, was built on DAML+OIL.

An ontology is defined using a mixture of both OWL and RDF tags, prefixed by owl: and rdf: respectively. In defining a class definition in an ontology (see Figure 1), the tag 'owl:Class' tag is used to denote the beginning of a class definition along with the rdf:ID tag to denote the name of the class.

The end of a class definition is denoted by the </owl:Class> tag. Subclass relationships are denoted by the rdfs:subClassOf tag and a rdf:resource tag which point to the definition of the superclass. The superclass definition may be defined within the same ontology or defined in another ontology.

The OWL example in Figure 2 is drawn from three ontologies: agent.ont, foaf-basic.ont, and location.ont, defined by Creative Commons [14]. The agent ontology defines a class called Agent and a class called Person, wherein Person is defined to be a subclass of Agent by the statement <rdfs:subClassOf rdf:resource="#Agent"/> in the definition of Person. However, Agent is also defined as a subclass by the following statement in its definition:

```
<rdfs:subClassOf rdf:resource="&foaf;Agent"/>
<rdfs:subClassOf rdf:resource="&loc;ThingHasLocationContext"/>
```

which states the Agent in agent.ont is subclassed from two classes found in ontologies defined in the other OWL documents referred to by "&foaf;Agent"and "&loc;ThingHasLocationContext".

Similarly we may define a class called Agent using the RDFS tag Class and DAML to place a further restriction on the Name property. This is illustrated in Figure 3. In both DAML and OWL, subclass definitions use the RDFS subClassOf tag.

Therefore, class hierarchies may be represented in both OWL and DAML, which allows our metrics suite to be calculated on both types of documents.

DOI: 10.1002/smr

```
<rdfs:Class rdf:ID="Agent">
 <rdfs:subClassOf rdf:resource="="&foaf;Agent"/>
  <daml:restrictedBy>
   <daml:Restriction>
    <daml:onProperty rdf:resource="#Name"/>
    <daml:toClassrdf:resource="http://www.w3.org/TR/xmlschema2/#string"/>
  </daml:Restriction>
 </daml:restrictedBy>
</rdfs:Class>
```

Figure 2. Example Agent and Person classes in OWL.

```
<owl:Class rdf:ID="Agent">
   <rdfs:label>Agent</refs:label>
   <rdfs:subClassOf rdf:resource="&foaf;Agent"/>
   <rdfs:subClassOf rdf:resource="&loc;ThingHasLocationContext"/>
</owl:Class>

<owl:Class rdf:ID="Person">
   <rdfs:label>Person</refs:label>
   <rdfs:subClassOf rdf:resource="#Agent"/>
</owl:Class>
```

Figure 3. RDFS tag class example using DAML.

2.2. Previous ontology quality analysis

As previously discussed in the introduction, Burton-Jones *et al.* provided a suite of metrics for analyzing ontology quality. They used a deductive approach, based on a semiotic framework derived from linguistics theory that includes general elements of quality based on linguistic-style concepts related to the analysis of signs and symbols [9].

Burton-Jones *et al.* measured syntactic quality, semantic quality, pragmatic quality, and social quality, using sub-attributes such as lawfulness (correctness of syntax in terms of markup or html-style errors), richness (number of different properties used within an ontology), interpretability (meaningfulness of terms in relation to WordNet), consistency (whether a term is used in more than one way within an ontology), clarity (number of word senses of each word in an ontology), comprehensiveness (number of classes and properties), accuracy (checking knowledge in an ontology versus knowledge known to be true: currently cannot be automated and must be collected by a domain engineer), relevance (number of statements in the ontology that are used), authority (average number of links between ontologies within a library), and history (number of times an ontology has been used).

In all cases they primarily consider the linguistic aspects of an ontology. Our approach is different in that we primarily consider the ontology as it would appear when deployed as part of a software system.

DOI: 10.1002/smr

Recently, Orme *et al.* [15] examined coupling between ontologies deployed as a part of a software system. However, this study examines cohesion and complexity metrics, whereas Orme *et al.* examined coupling metrics. Also, Orme *et al.* examined a single ontology snapshot, whereas in this study we examine the evolution of an ontology.

Burton-Jones *et al.* [9] also considered only one snapshot of each ontology (from the DAML library), and did not examine how changes in ontology metrics as the ontology evolves could affect ontology stability, completeness, and domain focus. In our approach, we apply our metrics to the ontology as it evolves, examining different releases.

2.3. Software metrics

The metrics described in this paper have some similarity in the way they are calculated with OO software metrics previously proposed to measure the design properties of OO source code [16,17]. The difference between our new metrics and some of the existing OO metrics is that our metrics are calculated on ontology data stored in OWL format, whereas the existing OO metrics were calculated on software written in an OO programming language such as C++ or Java. Our metrics have meaning within the application domain. They are also applicable to the software because ontology-based software uses this application domain data (the ontology), stored in OWL format, to do its job. Since an ontology-based system always requires an ontology in order to work, the ontology-based system will not work well unless it can use a quality ontology.

However, the existing OO metrics measure the software itself. In the future, a combination of our new metrics and existing OO metrics would form a complete picture of the quality of an ontology-based system.

Various such OO metrics have also been employed in the past to examine software maintenance and software evolution [18–21]. Our new metrics focus on evolution of the ontology data used by ontology-based systems, whereas existing metrics focused on the evolution of the software itself. One primary example of OO software metrics that have been used in the past to analyze software evolution [18,20] and maintenance [21] is the metric suite proposed by Chidamber and Kemerer [22].

- Depth of Inheritance Tree (DIT): measures the distance from the base class to the bottommost children in an inheritance tree.
- Number of Children (NOC): measures the number of classes that inherit the current class.
- Response for a Class (RFC): count of all local methods plus all methods called by local methods.
- Coupling Between Objects (CBO): count of all the non-inheritance related couples between classes.
- Lack of Cohesion of Methods (LCOM): The cohesion of a class is characterized by how closely the local methods are related to the local instance variables.
- Weighted Methods per Class (WMC): is the sum of the complexities of all local methods.

To illustrate the similarities in calculation, our Maximum Depth of Inheritance Tree metric (MaxDIT) is a count of the depth of an inheritance tree and DIT is also a count of the depth of an inheritance tree. However, DIT is calculated on a class hierarchy (perhaps C++ or Java classes) in OO software while our MaxDIT is calculated on a class hierarchy in ontology data in OWL format. Our Number of Properties metric counts the number of properties in an ontology (a collection of

DOI: 10.1002/smr

ontology classes, each of which describes a term in an application domain), while WMC (basic or default version) counts the number of methods in one, for example, C++ or Java class in OO software.

To clarify, a class in ontology data is quite different from a C++ or Java class. A C++ or Java class represents a software entity that has meaning within the software itself, and is related to how the software operates. A class in ontology data represents an entity that has meaning within the application domain of the software.

2.4. Metrics validation framework

Kitchenham *et al*. [11] proposed a framework for evaluating software metrics. In this framework, they described the structure of any measure as containing the entities being analyzed (such as classes or modules), the attribute being measured (such as size), the unit used (such as lines of code), and the data scale (nominal, ordinal, interval, or ratio). Units are valid only for interval or ratio data, but they can be adapted for use with ordinal data. In order for a value to have any meaning, the entity, the attribute being measured, and the units must be specified. The measure must be defined over a specified set of permissible values (discrete or continuous).

In order to be valid, a measure must have [11]:

- attribute validity—the entity being analyzed has the attribute;
- unit validity—the unit is appropriate for the attribute;
- instrumental validity—the underlying model is valid and the instrument was calibrated;
- protocol validity—the protocol used for the measurement was valid and prevented errors such as double counting.

Furthermore, in order to be theoretically valid, a direct measure must have the following properties:

- the attribute has different values for different entities;
- the measure works in a way that makes sense with respect to the attribute and its values for different entities;
- any of the attribute's units can be used if the attribute is part of a valid measure;
- the attribute can have the same value for different entities [11].

In addition, for an indirect measure, the following properties apply:

- a model of relationships among entities' attributes is the basis for the measure;
- no improper use of dimensionality occurs in the measure;
- no unexpected discontinuities occur in the measure;
- the units used are appropriate for the scale of data available [11].

In addition to theoretical validation, Kitchenham, Pfleeger, and Fenton recommend empirical validation using statistical analysis of metric values compared to evaluators' assessment of software.

3. METRIC DEFINITIONS

In this section we present the following metrics: Number of Properties (NoP), Average Properties per Class (AP-C), Average Fanout per Class (AF-C), Average Fanout of Root Class (AF-R),

DOI: 10.1002/smr

Average Fanout of Non-Leaf Classes (AF-NL), and Maximum Depth of Inheritance Tree (MaxDIT). Section 3.1 defines the common mathematical notation for the development of our metrics. Section 3.2 provides a conceptual representation along with the formal definition for our metrics suite.

3.1. Formal notation

In order to define metrics, we introduce the following formal notation. This formal notation is used in the mathematical definition of our metrics.

- Let $C_1, C_2, \ldots, C_m$ be the set of m classes defined in an ontology.
- Let $P_1, P_2, \ldots, P_n$ be the set of n properties defined in an ontology.
- Let $F_{c1}, F_{c2}, \ldots, F_{cm}$ be the Fanout of each class C_i (see below for the definition of Fanout).
- Let O_1 be the ontology of interest.
- Let $\rightarrow$ be a mapping from set C_i, to C_j such that $C_i \rightarrow C_j$ if class C_j is a subclass of class C_i.
- Let & be a mapping from set C_i to set P_j such that C_i & P_j if class C_i includes property P_j as part of its definition.

OWL defines many constructs that allow the definition of a tree-based graph to determine the relationship between entities defined in the ontology. The following terminology is used to describe the tree-based relationships in our paper.

- Let A be a finite set, and let T be a relation on A. We say that T is a tree if there is a vertex V_0 in A with the property that there exists a unique path in T from V_0 to every other vertex in A (see [23]).
- Let A be a finite set, and let T be a relation on A. A vertex V_n has a fanout of degree m if there exist m relationships to other vertices in A (see [23]). Here 'relationship' is defined recursively: if vertex V_2 is a subvertex of V_1, then the relationships of V_2 are included in the fanout of V_1. That is, the children of V_1 are included in the count, and the grandchildren and great-grandchildren of V_1 are also included in the count. We refer to the relationships of children, grandchildren, etc., as *sublinks*.
- Let A be a finite set, and let T be a relation on A. A vertex V_q is called a leaf if it has a fanout of degree 0.

3.2. Metrics definitions

In this section we present the conceptual and mathematical basis for the following metrics: Number of Properties (NoP), Average Properties per Class (AP-C), Average Fanout per Class (AF-C), Average Fanout of Root Class (AF-R), Average Fanout of Non-Leaf Classes (AF-NL), and Maximum Depth of Inheritance Tree (MaxDIT).

The Number of Properties metric provides an indication both of how complex an ontology is and also how completely the terms in the ontology have been defined. A class possessing a large number of properties represents a class that has been fully described, thus an ontology consisting of several well-defined classes should contain a large number of properties. The Average Properties per Class gives an overall indication of how well individual classes within the ontology have been defined. The Average Fanout per Root Class metric measures how many different kinds of terms are being

DOI: 10.1002/smr

defined within the ontology (this provides an indication of both complexity and cohesion), while the Average Fanout per Class and Average Fanout of Non-Leaf Classes measure the complexity and descriptiveness of individual portions of the ontology. The Maximum Depth of Inheritance Tree provides another complexity indication; it also provides an indication of how fine grained the terms in the ontology are. For example, an ontology which says a field mouse is-a mouse is-a mammal provides a more fine-grained definition than an ontology which did not contain field mouse. The Maximum Depth of Inheritance tree would be a larger number for the field mouse hierarchy than for the mouse hierarchy. Thus, these metrics can provide a practitioner with a useful indication of various aspects of ontology quality.

Definitions of the metrics Number of Classes (NoC), Number of Root Classes (NoR), Number of Leaf Classes (NoL), Number of External Classes (NEC), Reference to External Classes (REC), Reference Includes (RI), Number of Classes (NoC), NoF (Number of Fanouts) and Average Depth of Inheritance Tree of All Leaf Nodes (ADIT-LN) are provided elsewhere [15].

3.2.1. *Definition of NoP (Number of Properties)*

Number of Properties (NoP) is the number of properties explicitly defined in the ontology O_i. Mathematically, NoP can be formulated as follows:

$$\mathrm{NoP}(O_i) = \sum P_j$$

for all $1 \leq j \leq n$ (number of properties defined in O_i).

Example. In the ontology shown in Figure 4, each class can possess properties (but is not required to). Class 1 has one property, class 2 has two properties, class 3 has no properties, class 4 has three properties, and class 5 has one property. Thus, NoP = 7.

3.2.2. *Definition of AP-C (Average Properties per Class)*

The Average Properties per Class (*AP-C*) is the number of properties explicitly defined in the ontology O_i divided by the number of classes explicitly defined in the ontology O_i. Mathematically, AP-C can be formulated as follows:

$$\mathrm{AP-C}(O_i) = \sum P_j \Big/ \sum C_k$$

for all $1 \leq j \leq m$ (number of properties defined in O_i) and for all $1 \leq k \leq n$ (number of classes defined in O_i). From Figure 4, $\mathrm{AP\text{-}C}(O_i) = 7/5 = 1.4$.

3.2.3. *Definition of NoF (Number of Fanouts)*

We previously defined Number of Fanouts (NoF) to be the sum of the fanouts of each node explicitly defined in the ontology O_i. Mathematically, NoF can be formulated as follows:

$$\mathrm{NoF}(O_i) = \sum_j F_j$$

for all $1 \leq j \leq n$, where n is the number of classes in O_i.

DOI: 10.1002/smr

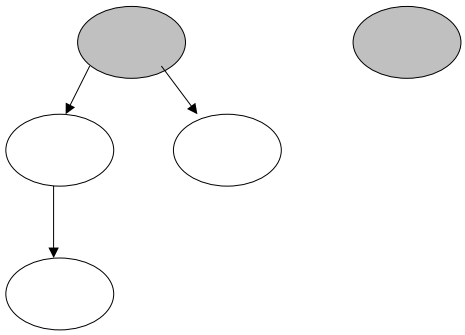


Figure 4. Conceptual view of NoP. Ellipses represent ontology classes. Shaded ellipses are root classes.

Note that F_j is defined recursively, the F_j for class C_1 counts the immediate relationships between C_1 and the immediate children of C_1, but it also counts the relationships of the subchildren of the immediate children of C_1 (grandchildren and great-grandchildren, etc.).

3.2.4. *Definition of AF-C (Average Fanout per Class)*

The Average Fanout per Class (AF-C) is the Number of Fanouts (NoF) divided by Number of Classes (NoC) explicitly defined in the ontology O_i. Mathematically, AF can be formulated as follows:

$$\mathrm{AF}(O_i) = \sum F_j \Big/ \sum C_j$$

for all $1 \leq j \leq n$ (number of classes in O_i).

Example. In Figure 5, $\sum F_j = (5+4+3+0+0+0)+0 = 12$ and $\sum C_j = 7$, therefore AF $=$ 12/7.

3.2.5. *Definition of AF-R (Average Fanout of Root Classes)*

The Average Fanout of Root Classes (AF-R) is defined to be the Number of Fanouts (NoF) divided by the Number of Root Classes in the ontology O_i. Mathematically, AF-R can be formulated as follows:

$$\mathrm{AF}(O_i) = \sum F_j \Big/ \sum \mathrm{R}(O_i)$$

for all $1 \leq j \leq n$ (j is number of classes in O_i). $\mathrm{R}(O_i)$ is defined as the number of root nodes in O_i.

Example. In Figure 5, $\sum F_j = 12$ and $\sum \mathrm{R}(O_i)$, therefore AF $-$ R $= 12/2$.

DOI: 10.1002/smr

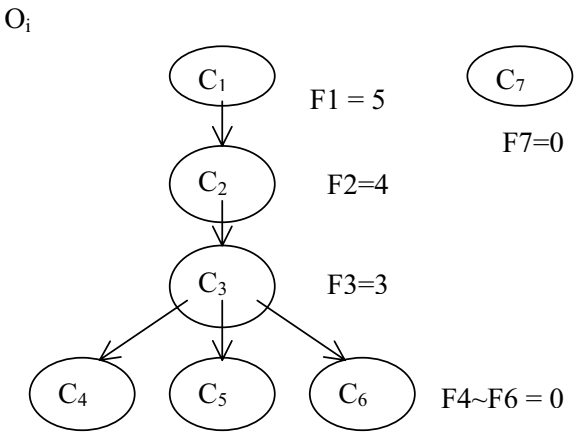


Figure 5. Conceptual view of NoF.

3.2.6. *Definition of AF-NL (Average Fanout of Non-Leaf Classes)*

The Average Fanout of Non-Leaf Classes (AF-NL) is defined as the Number of Fanouts (NoF) divided by Number of Non-leaf Classes explicitly defined in the ontology O_i.

Mathematically, AF-NL can be formulated as follows:

$$\mathrm{AF}(O_i) = \sum F_j \Big/ \sum \mathrm{NL}(O_i)$$

for all $1 \leq j \leq n$ (j is number of classes in O_i). F_j is the fanout of the jth node ($\mathrm{NL}(O_i)$ is the number of non-leaf nodes in O_i).

Example. In Figure 5, $\sum F_j = 12$ and $\sum \mathrm{NL}(O_i) = 3$, therefore $\mathrm{AF} - \mathrm{NL} = 12/3 = 4$.

3.2.7. *Definition of MaxDIT (Maximum Depth of Inheritance Tree)*

The Maximum Depth of Inheritance Tree (MaxDIT) of the ontology O_i is defined as follows. A depth is the total number of nodes in a path. A path starts from the root node to the leaf node. All distinct paths in the ontology O_i include every path from each root node to each leaf node if there exists an inheritance path from the root node to the leaf node. The root node is the first level in each path. Mathematically, MaxDIT is formulated as follows:

$$\mathrm{MaxDIT}(O_i) = \mathrm{Max}(D_j)$$

for all $1 \leq j \leq n$ (number of paths in O_j). D_j is the depth of the jth path.

Example. in In Figure 6, MaxDIT = 3 (from C_1 to C_4).

DOI: 10.1002/smr

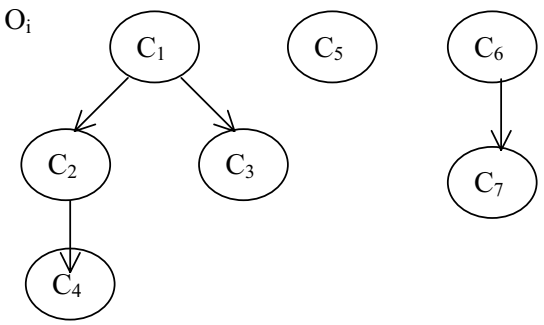


Figure 6. Conceptual view of MaxDIT.

4. ANALYSIS OF METRICS

In this section we present a theoretical and empirical analysis of the ontology metrics suite. First, in Section 4.1, each metric is examined theoretically using the Kitchenham *et al.* [11] framework. Second, in Section 4.2 we perform an empirical analysis of these metrics.

4.1. Theoretical analysis

4.1.1. Analysis of the Number of the Properties (NoP)

To analyze the Number of Properties (NoP) within the framework proposed by [11], we first define the following:

- the entity being analyzed as O_i, the ontology of interest;
- the attribute being measured as the Properties;
- the unit as the Property;
- the data scale is the interval;

NoP is a direct measure of counting the number of properties in the ontology. This measure meets the first four of Kitchenham *et al.*'s eight properties for validity as follows.

(1) *Attribute validity*: the entity (ontology) has the attribute (properties), a measure of the number of properties in the ontology.
(2) *Unit validity*: the idea is an appropriate measure of the Number of Properties in the ontology.
(3) *Instrumental validity*: the instrument is valid as long as our tool parses and counts the number of properties in the ontology correctly.
(4) *Protocol validity*: the measurement as defined in the formal notation and shown conceptually is consistent and prevents double counting.

DOI: 10.1002/smr

4.1.2. Analysis of the Average Properties per Class (AP-C)

To analyze the Average Properties per Class (AP-C) within the framework proposed by [11], we first define the following:

- the entity being analyzed as O_i, the ontology of interest;
- the attribute being measured as the Average Properties per Class;
- the unit as the Properties per Class;
- the data scale is continuous.

Average Properties per Class (AP-C) is an indirect measure, counting the properties and classes in the ontology. This measure meets Kitchenham *et al.*'s eight properties for validity as follows.

(1) *Attribute validity*: the entity (ontology) has the attribute (properties), which enables us to provide an indirect measure of the Average Properties per Class (AP-C) in the ontology.
(2) *Unit validity*: the idea is an appropriate measure of properties per class in the ontology.
(3) *Instrumental validity*: the instrument is valid as long as our tool parses and counts the properties and classes in the ontology and averages correctly.
(4) *Protocol validity*: the measurement as defined in the formal notation and shown conceptually is consistent and prevents double counting.
(5) *The model of relationship among the entities' attributes is the basis for the metric*: a relationship exists between the class and properties; classes must exist for properties to exist.
(6) *No improper use of dimensionality*: averaging the properties per class assures the proper use of dimensionality to derive the properties per class.
(7) *No discontinuities*: the measurement is assured of no discontinuities, in that classes must exist for properties to exist.
(8) *Unit scale*: the unit scale is appropriate for measuring ontologies that contain properties defined within a class.

4.1.3. Analysis of the Average Fanout per Class (AF-C)

To analyze the Average Fanout per Class (AF-C) within the framework proposed by [11], we first define the following:

- the entity being analyzed as O_i, the ontology of interest;
- the attribute being measured as the fanout;
- the unit as the Average Fanout per Class;
- the data scale is continuous.

The Average Fanout per Class (AF-C) is an indirect measure, counting the fanout and classes in the ontology. This measure meets Kitchenham *et al.*'s eight properties for validity as follows.

(1) *Attribute validity*: the entity (ontology) has the attribute (fanout), which enables us to provide the average number of links and sublinks per class. A link is a direct relationship between a class and another class. A sublink is defined as links to classes which are children (or grandchildren, recursively) of the current class.

DOI: 10.1002/smr

(2) *Unit validity*: the idea is an appropriate measure of the Average Fanout per Class in the ontology.
(3) *Instrumental validity*: the instrument is valid as long as our tool parses and correctly counts the fanout and classes in the ontology, and averages them correctly.
(4) *Protocol validity*: the measurement as defined in the formal notation and shown conceptually is consistent and prevents double counting.
(5) *The model of relationship among the entities' attributes is the basis for the metric*: a relationship exists between the class and fanout; classes must exist for fanout to exist.
(6) *No improper use of dimensionality*: averaging the fanout per class assures the proper use of dimensionality to derive the average fanout per class.
(7) *No discontinuities*: the measurement is assured of no discontinuities, in that classes must exist for fanout to exist.
(8) *Unit scale*: the unit scale is appropriate in measuring ontologies that contain classes.

4.1.4. Analysis of the Average Fanout of Root Classes (AF-RC)

To analyze the Average Fanout of Root Classes (AF-RC) within the framework proposed by [11], we first define the following:

- the entity being analyzed as O_i, the ontology of interest;
- the attribute being measured as the fanout;
- the unit as the fanout per root class;
- the data scale is continuous.

The Average Fanout of Root Classes (AF-RC) is an indirect measure, counting the fanout and root classes in the ontology. This measure meets Kitchenham *et al.*'s eight properties for validity as follows.

(1) *Attribute validity*: the entity (ontology) has the attribute (fanout), a measure of the number of links and sublinks of a class in the ontology. A link is a direct relationship between a class and another class. A sublink is defined as links to classes which are children (or grandchildren, recursively) of the current class.
(2) *Unit validity*: the idea is an appropriate measure of fanout per root class in the ontology.
(3) *Instrumental validity*: the instrument is valid as long as our tool parses and counts correctly the fanout and root classes in the ontology.
(4) *Protocol validity*: the measurement as defined in the formal notation and shown conceptually is consistent and prevents double counting.
(5) *The model of relationship among the entities' attributes is the basis for the metric*: a relationship exists between the root classes and fanout; root classes must exist for fanout to exist.
(6) *No improper use of dimensionality*: averaging the fanout assures the proper use of dimensionality to derive the average fanout of root classes.
(7) *No discontinuities*: the measurement is assured of no discontinuities, in that root classes must exist for fanout to exist.
(8) *Unit scale*: the unit scale is appropriate in measuring ontologies that contain fanout within a class.

DOI: 10.1002/smr

4.1.5. *Analysis of the Average Fanout of Non-Leaf Classes (AF-NL)*

To analyze the Average Fanout of Non-Leaf Classes (AF-NL) within the framework proposed by [11], we first define the following:

- the entity being analyzed as O_i, the ontology of interest;
- the attribute being measured as fanout;
- the unit as the fanout per non-leaf class;
- the data scale is continuous.

The Average Fanout of Non-Leaf Classes (AF-NL) is an indirect measure of counting the fanout and non-leaf classes in the ontology. This measure meets Kitchenham *et al.*'s eight properties for validity as follows.

(1) *Attribute validity*: the entity (ontology) has the attribute (fanout), a measure of the number of links and sublinks in the ontology. A link is a direct relationship between a class and another class. A sublink is defined as links to classes which are children (or grandchildren, recursively) of the current class.
(2) *Unit validity*: the idea is an appropriate measure of the fanout per non-leaf class in the ontology.
(3) *Instrumental validity*: the instrument is valid as long as our tool parses and counts the fanout and non-leaf classes in the ontology correctly.
(4) *Protocol validity*: the measurement as defined in the formal notation and shown conceptually is consistent and prevents double counting.
(5) *The model of relationship among the entities' attributes is the basis for the metric*: a relationship exists between the non-leaf classes and fanout, non-leaf classes must exist for fanout to exist.
(6) *No improper use of dimensionality*: averaging the fanout assures the proper use of dimensionality to derive the average fanout per non-leaf class.
(7) *No discontinuities*: the measurement is assured of no discontinuities, in that non-leaf classes must exist for fanout to exist.
(8) *Unit scale*: the unit scale is appropriate in measuring ontologies that contain fanout within a class.

4.1.6. *Maximum Depth of Inheritance Tree (MaxDIT)*

To analyze the Maximum Depth of Inheritance Tree (MaxDIT) within the framework proposed by [11], we first define the following:

- the entity being analyzed as O_i, the ontology of interest;
- the attribute being measured as the inheritance;
- the unit as the depth of inheritance;
- the data scale is interval.

The Maximum Depth of Inheritance Tree (MaxDIT) is a direct measure of counting the depth of inheritance in the ontology. This measure meets the first four of Kitchenham *et al.*'s eight properties for validity as follows.

DOI: 10.1002/smr

(1) *Attribute validity*: the entity (ontology) has the attribute (inheritance), a measure of the depth of inheritance within the ontology.
(2) *Unit validity*: the idea is an appropriate measure of the depth of inheritance in the ontology.
(3) *Instrumental validity*: the instrument is valid as long as our tool parses and counts the depth in the ontology correctly .
(4) *Protocol validity*: the measurement as defined in the formal notation and shown conceptually is consistent and prevents double counting.

4.2. Empirical analysis

We performed two separate experiments to analyze the ontology metrics suite. The purpose of experiment 1 was to examine the use of ontology data complexity and cohesion metrics to determine ontology data quality. The purpose of experiment 2 was to employ ontology data complexity and cohesion metrics to examine the impact of change on ontology data quality. The potential benefits of ontology complexity and cohesion metrics in determining ontology data completeness and stability were also examined in experiment 2. In this context, cohesion means the degree to which the data items that together form an ontology are related to each other (the degree to which the ontology refers to a single domain-related area).

4.2.1. Methodology employed in empirical analysis

In experiment 1, we collected ontology complexity metrics over 33 independent ontologies developed by Creative Commons. Then we assembled a panel of 18 evaluators to assess each of the 33 ontologies to determine each ontology's complexity. In experiment 1, our evaluators' average experience in developing software projects was 3.5 years. For experiment 1 we were interested in how well our different metrics measured complexity and cohesion over this data set of 33 ontologies.

In experiment 2 we measured 12 separate ontology instances from a single domain, that of tourism. These tourism ontologies were defined as three separate ontologies, each of which was modified three other times (thus resulting in four instances each of a particular ontology): Tourism 1, Tourism 2, and Tourism 3. In experiment 2, we also used a panel of 18 evaluators. Our evaluators' average experience in developing software projects was 2.5 years.

For this experiment we are interested in how our metrics detect the extent of changes, and the types of changes, to a single ontology, and in how these changes vary from ontology to ontology. We are interested in detecting the stability of an ontology. If the ontology is changing regularly, it might not be stable enough for an application to depend on. Also, are the changes that are being made to the ontology desirable? If the ontology is getting larger and more complex then that is desirable from the standpoint of completeness; but when we require a lack of ambiguity it might not be desirable, particularly if the ontology is becoming substantially less cohesive as it becomes more complex.

In both experiments the evaluators were given an electronic copy of each ontology and asked to rate the complexity of each. The evaluators rated the complexity or cohesion of each of the 33 ontologies on the following scale:

DOI: 10.1002/smr

- 0 = low;
- 0.25 = moderate;
- 0.50 = average;
- 0.75 = high;
- 1.0 = extreme.

In the case of cohesion, low (0) meant that the sections of the ontology were not very related to other sections of the ontology and high (1.0) meant all sections of the ontology were related; whereas for complexity, low (0) meant the ontology was not very intricate or complicated while high (1.0) meant the ontology was very intricate or complicated.

We used SPSS software to compute interrater reliability over the human evaluators ratings, to determine how well our evaluators agreed with one another. This measurement is reported on a per-experiment basis.

4.2.2. *Empirical analysis results from experiment 1*

In experiment 1, after performing a test for the normality of data, we used Pearson's correlation coefficient to compare the evaluators' complexity ratings with the metrics values.

The following hypotheses were employed:

- H_0: $\rho = 0$ (there is no correlation between the metric value and the evaluators' rating);
- H_1: $\rho \neq 0$ (there is a correlation between the metric value and the evaluators' rating).

Correlation coefficients range from -1.0 to 1.0 with numbers closer to either end stating a higher degree of correlation, meaning that the values being correlated are not independent of one another. If the correlation coefficient is closer to 0, then the variables being correlated are more independent of one another. The p-values calculated were in relation to the hypothesis test of the correlation coefficient being zero.

To understand the meanings of these values, Cohen [24] proposed the following scale:

- < 0.01 = trivial;
- 0.10–0.30 = minor;
- 0.30–0.50 = moderate;
- 0.50–0.70 = large;
- 0.70–0.90 = very large;
- 0.90–1.0 = almost perfect.

4.2.2.1. Complexity empirical analysis results Table I summarizes the results of comparing each of the ontology complexity metrics with the evaluators' ratings of complexity in experiment 1. Here, the metrics NoC, NoL, NoR, NoP, NoF, AF-C, MaxDIT, and ADIT-LN showed a statistically significant large or very large correlation with the evaluators ratings at $\alpha = 0.05$ (95% significance level). AP-C and AF-NL had no statistically significant correlation with the evaluators ratings, while AF-R had a moderate correlation at somewhat less than $\alpha = 0.01$ (95% significance level). Thus, several of these metrics successfully indicate the complexity of an ontology. The interrater reliability is 0.9160, which indicates a consistent agreement between the evaluators.

DOI: 10.1002/smr

Table I. Statistical complexity analysis over 33 independent ontologies: experiment 1.

Correlated metrics	Correlation	p-value
NoC to evaluators	0.885	<0.001
NoL to evaluators	0.836	<0.001
NoR nodes to evaluators	0.611	<0.001
NoP to evaluators	0.735	<0.001
AP-C to evaluators	−0.129	0.497
NoF evaluators	0.802	<0.001
AF-C to evaluators	0.596	0.001
AF-R to evaluators	0.451	0.012
AF-NL to evaluators	−0.076	0.789
MaxDIT to evaluators	0.808	<0.001
ADIT-LN to evaluators	0.588	0.001

Table II. Statistical cohesion analysis over 33 independent ontologies: experiment 1.

Correlated metrics	Correlation	p-value
NoC to evaluators	0.686	<0.001
NoL to evaluators	0.687	<0.001
NoR nodes to evaluators	0.386	0.035
NoP to evaluators	0.695	<0.001
AP-C to evaluators	0.065	0.733
NoF evaluators	0.658	<0.001
AF-C to evaluators	0.680	<0.001
AF-R to evaluators	0.440	0.015
AF-NL to evaluators	0.271	0.438
MaxDIT to evaluators	0.765	<0.001
ADIT-LN to evaluators	0.641	<0.001

4.2.2.2. Cohesion empirical analysis results Table II summarizes the results of comparing each of the ontology cohesion metrics with the evaluators' ratings of complexity in experiment 1. Here, the metrics NoC, NoL, NoP, NoF, AF-C, MaxDIT, and ADIT-LN have statistically significant large or very large correlations with the evaluators' ratings at a 95% significance level. NoR, AP-C, AF-R, and AF-NL had no statistically significant correlation with the evaluators. Thus, several of these metrics successfully indicate the cohesion of an ontology. The interrater reliability is 0.9014, which indicates a consistent agreement between the evaluators.

DOI: 10.1002/smr

Table III. Individual metrics data, with evaluators' values: Tourism 1.

Evaluators' complexity rating	NoC	NoL	NoR	NoP	AP-C	NoF	AF-C	AF-R	AF-NL	MaxDIT	ADIT-LN	Evaluators' cohesion rating
63.33	339	258	4	64	0.189	343	1.012	85.75	4.235	7	5.316	50.00
96.66	473	351	4	61	0.129	499	1.055	124.75	4.090	8	5.346	40.00
53.33	337	249	4	64	0.190	343	1.018	85.75	3.898	9	5.965	58.33
25.00	299	227	4	38	0.127	311	1.040	77.75	4.319	7	5.442	65.00

4.2.3. Empirical analysis results from experiment 2

As mentioned above, in experiment 2 we measured 12 separate ontology instances from a single domain, that of tourism. These tourism ontologies were defined as three separate ontologies, each of which was modified three other times (thus resulting in four instances each of a particular ontology): Tourism 1, Tourism 2, and Tourism 3. Thus, there are four instances of the Tourism 1 ontology, four instances of the Tourism 2 ontology, and four instances of the Tourism 3 ontology. The modifications to each ontology represent additional information being added to the ontology, as well as changes to the ontology to fix previous problems.

For this experiment we are interested in how our metrics detect the extent of changes, and the types of changes, to a single ontology, and in how these changes vary from ontology to ontology. We are interested in detecting the stability of an ontology. If the ontology is changing regularly, it might not be stable enough for an application to depend on. Also, are the changes that are being made to the ontology desirable? If the ontology is getting larger and more complex, then that is desirable from the standpoint of completeness; but if we require a lack of ambiguity it might not be desirable.

This section is divided into analysis 1, an examination of individual metrics, and analysis 2, a statistical comparison of metrics versus human evaluators. In analysis 1 we look at individual metrics values, and draw conclusions based on the individual values ('visual inspection' comparisons). In analysis 2, we perform statistical comparisons to determine whether our earlier 'visual inspection' conclusions have strength in statistical terms, and whether any additional statistically based conclusions can be drawn from these data.

4.2.3.1. Analysis 1: examination of individual metrics—complexity and cohesion Examining Table III, for the Tourism 1 ontology, the Number of Leaf Classes (NoL) varies quite a lot with changes in that ontology. We note that it went up considerably on the second iteration, but actually reduced NoL on the third and fourth iterations. Presumably this resulted from some refactoring of inefficiencies in the ontology.

It is quite interesting, however, that the number of leaf nodes in the fourth iteration is actually smaller than that in the first iteration. We suspected originally that NoL might be used to indicate ontology completeness. The idea is that a more complex ontology includes more of the domain-level

DOI: 10.1002/smr

Table IV. Individual metrics data, with evaluators' values: Tourism 2.

Evaluators' complexity rating	NoC	NoL	NoR	NoP	AP-C	NoF	AF-C	AF-R	AF-NL	MaxDIT	ADIT-LN	Evaluators' cohesion rating
65.00	309	231	4	88	0.285	310	1.003	77.50	3.974	7	4.736	35.00
78.33	298	226	4	102	0.342	302	1.013	77.50	4.194	7	4.777	38.33
63.33	307	245	5	69	0.225	306	0.997	61.20	4.935	10	5.762	50.00
48.33	309	237	6	68	0.220	304	0.984	50.66	4.222	8	4.857	63.33

Table V. Individual metrics data, with evaluators' values: Tourism 3.

Evaluators' complexity rating	NoC	NoL	NoR	NoP	AP-C	NoF	AF-C	AF-R	AF-NL	MaxDIT	ADIT-LN	Evaluators' cohesion rating
46.66	304	248	5	64	0.211	302	0.993	60.4	5.392	8	4.891	46.66
55.00	304	248	6	76	0.250	304	1.000	60.8	5.429	8	4.851	54.00
71.66	304	248	5	81	0.266	302	0.993	60.4	5.393	8	4.891	51.66
35.00	304	248	5	64	0.211	304	1.000	60.8	5.429	8	4.850	63.33

information required, and thus is closer to complete. Further study is required to determine whether this indicates that the ontology has actually improved. We note a similar pattern (in Table IV) in the Tourism 2 ontology, in that NoL reduces with certain iterations. However, for the Tourism 3 ontology (in Table V), the number of leaf nodes did not change at all over the four iterations.

Our original expectation was that an increase in the NoL would reflect an increase in ontology complexity. When comparing individual values of NoL with the evaluators' complexity rating for the Tourism 1 ontology, that would seem to be the case. However, when comparing NoL with the evaluators' complexity rating for the Tourism 2 ontology, the relationship expected was not so clear: NoL went from 231 down to 226, while the evaluators' complexity rating went up from 65.00 to 78.33. Similarly, for the Tourism 3 ontology, evaluators' complexity ratings changed over the various ontology iterations while NoL did not change.

The (tentative) conclusion we draw is that NoL measures one kind of complexity, but that there is more than one kind of complexity in an ontology. For example, one type of ontology complexity results from simply adding new low-level nodes to an existing node hierarchy. If one has cat is-a mammal is-an animal as part of your hierarchy, and the new term 'dog' is added to the hierarchy, such that one has dog is-a mammal is-an animal, this would be an increase in complexity that would be reflected in the NoL metric.

DOI: 10.1002/smr

Our original expectation was that an increase in NoL would correspond to a reduction in ontology cohesion (lower cohesion). The argument is that the longer the list of nodes, the less likely that each node would be heavily related to the other nodes in the list. This does seem to be the case in Tourism 1. However, it does not seem to be strictly the case in Tourism 2 and, of course, in Tourism 3 NoL did not change.

It is interesting that the Number of Root Classes (NoR) varies very little among the Tourism complexity values. In Tourism 1, it does not vary at all; in Tourism 2, it varies by a total of 2 (one step per iteration), and in Tourism 3 it varies by 1. Presumably this means that the basic areas covered by these ontologies change very little. This is important, if true, because it may mean that the changes to these ontologies do not include adding additional domain areas to the ontology. This could be an indication that the ontology is complete for the particular domain it addresses.

However, our expectation had been that NoR would reflect complexity. As it changed very little while the evaluators' complexity ratings did change, from an examination of these tables this does not appear to be the case. Similarly, our expectation had been that NoR would be a very good indicator of cohesion, since adding additional type hierarchies to the ontology (a new root node is a new base node in a type hierarchy) would result in an overall reduction in cohesion. From Tourism 2, this does not seem to be the case: as NoR increases, the evaluators' cohesion ratings actually improve.

The Number of Fanouts (NoF) varied quite a bit for Tourism 1, while it varied very little for Tourism 3, and had a limited variation for Tourism 2. We note that this corresponds to the NoL analysis, in that NoL varied considerably for Tourism 1, not at all for Tourism 3, and not much for Tourism 2. Also, for Tourism 1 NoF went up considerably for iteration 2, similarly to what the number of leaf nodes did (see the earlier analysis). This possibly indicates a heavy refactoring of Tourism 1. These two metrics together (NoL and NoF) tend to indicate that the data in Tourism 1 are much less stable than that in Tourism 3, and somewhat less stable than that in Tourism 2. Such large changes to an ontology would result in varying results for any software package that employed these ontology data. Thus, from this standpoint, Tourism 2 and Tourism 3 would appear to be more desirable ontologies for the Tourism area than would Tourism 1.

We also note that MaxDIT varies the most for Tourism 2. This would appear to mean that Tourism 2 is adding more low-level data nodes (adding new content), whereas Tourism 1 is changing existing higher-level data content, rather than simply adding new content. Our suspicion is that, in general, adding new content with new ontology data releases is desirable, whereas changing existing higher-level content indicates that the richness of expression of the existing data was insufficient. This again tends to indicate, in our opinion, that Tourism 2 is a more desirable ontology than Tourism 1, from the standpoint of stability. Note that Tourism 3 does not change at all. This may mean that Tourism 3 is not being improved, which is a negative; but it could also indicate that Tourism 3 does not need to be improved, that it is sufficient for its use, or that it is complete. Further study is required.

Although the ratings from the human evaluator teams were provided for comparison with the individual metrics data, it is difficult to tell from such a comparison (that we earlier called a 'visual inspection' comparison) what the relationship between each metric and the evaluators' ratings actually is. For this reason, although the data set is fairly small, we performed a statistical comparison of each of the metrics versus the human evaluators' ratings of complexity, and similarly versus the human evaluators' ratings of cohesion.

DOI: 10.1002/smr

4.2.3.2. Analysis 2: a statistical comparison of metrics versus human evaluators In the statistical comparison of metrics versus human evaluators, which was performed as the second part of experiment 2, after performing a test for normality of data, we used Pearson's correlation coefficient to compare the evaluators' complexity ratings with the metrics values. This statistical comparison was performed twice, once for complexity and once for cohesion.

The following hypotheses were employed:

- H_0: $\rho = 0$ (there is no correlation between the metric value and the evaluators' rating);
- H_1: $\rho \neq 0$ (there is a correlation between the metric value and the evaluators' rating).

Correlation coefficients range from -1.0 to 1.0 with numbers closer to either end stating a higher degree of correlation, meaning that the values being correlated are not independent of one another. If the correlation coefficient is closer to 0, then the variables being correlated are more independent of one another. The p-values calculated were in relation to the hypothesis test of the correlation coefficient being zero.

To understand the meanings of these values, Cohen [24] proposed the following scale:

- < 0.01 = trivial;
- 0.10–0.30 = minor;
- 0.30–0.50 = moderate;
- 0.50–0.70 = large;
- 0.70–0.90 = very large;
- 0.90–1.0 = almost perfect.

The correlations comparing evaluators' ratings of complexity to the various metrics are shown in Table VI. The correlations comparing evaluators' ratings of cohesion to the various metrics are shown in Table VII. The interrater reliability for complexity for Tourism 1 was 0.6915, Tourism 2 was 0.2013 and Tourism 3 was 0.7667. Thus, there was reasonable agreement among evaluators on Tourism 1 and Tourism 3 and some agreement on Tourism 2. The interrater reliability for cohesion for Tourism 1 was 0.0322, Tourism 2 was 0.1600 and Tourism 3 was 0.0040. Thus, there was some agreement among evaluators on Tourism 2 and little agreement on Tourism 1 and Tourism 3.

The results from the complexity comparison (see Table VI) show that for Tourism 1, the correlations of NoC, NoL, NoF, and AF-R with the evaluators' ratings are statistically significant with very large values (at above a 90% significance level). For Tourism 2, only AF-C is statistically significant; however, it has a very large value (at a 95% significance level). For Tourism 3, AP-C and NoF have very large statistically significant correlations (at above a 90% significance level). These statistical results tend to bear out our earlier 'visual inspection' comparisons on individual metrics values for NoL and NoF.

The results from the cohesion analysis (see Table VII) show that for Tourism 1, there are statistically significant very large correlations of NoC and NoL with the evaluators cohesion ratings at above a 90% significance level (NoF and AF-R were significant at above an 89% significance level). For Tourism 2, NoR, AF-C, and AF-R had very large statistically significant correlations (above an 85% significance level) with the evaluators' cohesion ratings. For Tourism 3, there were no statistically significant correlations.

DOI: 10.1002/smr

Table VI. Statistical complexity analysis: experiment 2.

	Tourism 1		Tourism 2		Tourism 3	
Correlated metrics	Correlation coefficient	p-value	Correlation coefficient	p-value	Correlation coefficient	p-value
NoC to evaluators	0.941	0.059	−0.825	0.175	0.991	0.009
NoL to evaluators	0.942	0.058	−0.586	0.432	*	*
NoR to evaluators	*	*	−0.886	0.114	*	*
NoP to evaluators	0.706	0.294	0.865	0.135	0.126	0.874
AP-C to evaluators	−0.029	0.971	0.877	0.123	0.923	0.077
NoF to evaluators	0.917	0.083	−0.179	0.821	0.923	0.077
AF-C to evaluators	0.329	0.671	0.987	0.013	0.530	0.470
AF-R to evaluators	0.917	0.083	0.867	0.133	0.530	0.470
AF-NL to evaluators	−0.378	0.622	−0.066	0.934	0.530	0.470
MaxDIT to evaluators	0.289	0.711	−0.320	0.680	0.530	0.470
ADIT-LN to evaluators	0.272	0.728	−0.095	0.905	*	*

*Column data were the same, therefore correlation cannot be computed.

Table VII. Cohesion correlation analysis: experiment 2.

	Tourism 1		Tourism 2		Tourism 3	
Correlated metrics	Correlation coefficient	p-value	Correlation coefficient	p-value	Correlation coefficient	p-value
NoC to evaluators	−0.961	0.084	0.420	0.580	*	*
NoL to evaluators	−0.928	0.072	0.598	0.402	*	*
NoR nodes to evaluators	*	*	0.994	0.006	0.008	0.992
NoP to evaluators	−0.646	0.354	−0.816	0.184	−0.206	0.794
AP-C to evaluators	0.073	0.927	−0.792	0.208	−0.206	0.794
NoF evaluators	−0.894	0.106	−0.355	0.645	0.786	0.214
AF-C to evaluators	−0.332	0.668	−0.901	0.099	0.786	0.214
AF-R to evaluators	−0.894	0.106	−0.989	0.011	0.786	0.214
AF-NL to evaluators	0.206	0.794	0.342	0.658	0.786	0.214
MaxDIT to evaluators	−0.108	0.892	0.490	0.510	*	*
ADIT-LN to evaluators	0.440	0.560	0.271	0.729	−0.786	0.214

*Column data were the same, therefore correlation cannot be computed.

DOI: 10.1002/smr

4.2.4. *Discussion of results*

In experiment 1, we examined the use of ontology data complexity and cohesion metrics to determine ontology data quality. In experiment 1, the correlations of several of the ontology data metrics were shown to be statistically significant, and in the range of large to very large when compared with evaluators' complexity ratings over a reasonably large data set of 33 ontologies. Thus, these metrics successfully indicate the complexity of ontologies, and can be employed in the future to indicate ontology quality in terms of complexity.

Also in experiment 1, the correlations of several of the ontology data metrics were shown to be statistically significant, and in the range of large to very large when compared with evaluators' cohesion ratings. Thus, several of these metrics successfully indicate the cohesion of ontologies, and can be employed in the future to indicate ontology quality in terms of cohesion.

In experiment 2, we examined the use of ontology data metrics to examine the stability and extent of changes over time to a single ontology (and how these types of changes vary from ontology to ontology as each ontology evolves). We also examined the use of ontology data metrics to determine ontology completeness. Through examination of changes in individual metrics, we detected trends in these metrics that could potentially indicate ontology stability. We detected other trends that could potentially indicate ontology completeness. However, the specific interpretation of metrics values as to how they indicate stability and completeness requires further examination. Particularly, the interaction between these different metrics might be more important that individual metrics values for an indication of stability or completeness.

In experiment 2, we compared the ontology data metrics to evaluators' ratings of complexity and cohesion over data sets consisting of different instances of the same ontology. Since this was difficult to perform well by examining individual metrics values, we performed statistical comparisons. Although the data set involved was fairly small, there were statistically significant correlations between several of the metrics and evaluators' ratings of complexity; similarly, there were statistically significant correlations between several of the metrics and evaluators' ratings of cohesion. Although additional work is required in this area, this experiment provides an initial indication that ontology data metrics can be used to indicate changes in complexity and cohesion over a single ontology. From this observation, we surmise indirectly that these metrics may also potentially measure ontology stability and completeness. Further study is required: as additional work on ontologies progresses (such as with data mining and the semantic Web), additional datasets should become available that will enable larger versions of this study.

5. CONCLUSIONS AND FUTURE RESEARCH

Our first experiment indicated that several of our metrics reflect ontology complexity and several reflect ontology cohesion. Our second experiment provided an initial indication that ontology metrics can measure ontology stability and completeness. We also determined that different metrics indicate different aspects of ontology stability and completeness. A complete examination of stability or completeness would entail examining interacting metrics values. We were able to identify some initial indications of how such interactions could be interpreted, but further study will be required in this area.

Complexity and cohesion are two factors of ontology quality. Complexity, for example, can potentially be a measure of ontology completeness (the more complex the ontology, we anticipate the

DOI: 10.1002/smr

more complete the ontology becomes). However, if an ontology goes up in complexity but its cohesion becomes markedly worse, it could indicate that the ontology is losing its focus on a single domain: that the ontology's complexity is not increasing due to more complicated relationships or new items within the current domain, but that new items not related to this domain are being added and the ontology is becoming unfocused. One of the strengths of our approach is that we analyzed how the relationship between different quality attributes affects quality, such as cohesion and complexity taken together. This has not been done for ontology metrics before.

Changes in complexity can indicate a lack of stability of an ontology. For example, over several iterations of a particular ontology, if complexity goes up in one iteration then down in another, that could indicate refactoring (tightening up) which could be a positive development. However, if complexity changes substantially on every iteration, one should be suspicious that the ontology developers do not have a good understanding of the domain.

To summarize, by using these ontology metrics, ontology developers will have an indication as to which ontologies should be examined more closely for inconsistencies or even errors (those ontologies that are very complex or very incohesive) prior to the ontologies being made available to the public for use in ontology-based applications. Also, ontology developers may be able to employ these metrics as ontology stability metrics, to determine whether changes over time during ontology evolution are adversely affecting the ontologies. These ontology metrics can also potentially be used as completeness measures: complexity measures could possibly determine when an ontology is sufficiently rich and interesting to be useful, while cohesion measures determine, when compared with complexity metrics on the same ontology, whether the ontology has lost its restrictions to a particular domain. An initial indication of this potential for ontology metrics was given here, but further study is required on this topic.

Our approach can provide an indication of ontology quality and also possibly completeness that can potentially be used in the future, on the fly, by ontology-based applications on the Web to choose between similar ontologies.

ACKNOWLEDGEMENTS

We wish to thank the anonymous reviewers of our paper for their generous and thoughtful comments.

REFERENCES

1. Swoogle. Ebiquity frequently asked questions. *Swoogle Semantic Web Search Engine Manual*. University of Maryland in Baltimore County Ebiquity Research Group: Baltimore MD, 2006; 5 pp.
   Available at: http://swoogle.umbc.edu/index.php?option=com_swoogle_manual&manual=faq [15 November 2006].
2. Berners-Lee T. Semantic Web road map. *Design Issues Draft*. World Wide Web Consortium: Cambridge MA, 2006; 9 pp. Available at: http://www.w3.org/DesignIssues/Semantic.html [15 November 2006].
3. Pistore M. Supporting the composition of distributed business processes: Research challenges. *WWW Service Composition with Semantic Web Services Workshop, Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*. University of Technology of Compiegne: Compiegne, France, 2005; 1.
4. W3C. OWL Web Ontology Language use cases and requirements. *W3C Recommendation 10 February 2004*. World Wide Web Consortium: Cambridge MA, 2006; 19 pp. Available at: http://www.w3.org/TR/webont-req/ [15 November 2006].
5. Meyer H, Overdick H, Weske M. Plaengine: A system for automated service composition and process enactment. *WWW Service Composition with Semantic Web Services Workshop, Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*. University of Technology of Compiegne: Compiegne, France, 2005; 3–12.

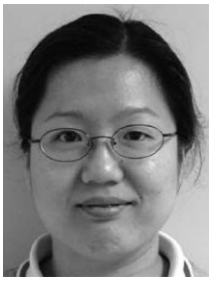
DOI: 10.1002/smr

6. Dogac A, Kabac Y, Laleci GB. Enriching ebXML registries with OWL ontologies for efficient service discovery. *Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications*. IEEE Computer Society Press: Los Alamitos CA, 2004; 69–76.
7. Cannataro M, Comito C. Semantics and knowledge Grids: Building the next-generation Grid. *IEEE Intelligent Systems* 2004; **19**(1):56–63.
8. Harte H, Lu Y, Osborn S, Dehoney D, Chin C. Refining the extraction of relevant documents from biomedical literature to create a corpus for pathway text mining. *Proceedings IEEE Bioinformatics Conference*. IEEE Computer Society Press: Los Alamitos CA, 2003; 644–645.
9. Burton-Jones A, Storey VC, Sugumaran V, Ahluwalia P. A semiotic metrics suite for assessing the quality of ontologies. *Data and Knowledge Engineering* 2005; **55**(1):84–102.
10. Chapin N, Hale JE, Khan KM, Ramil JF, Tan W. Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice* 2001; **13**(1):3–30.
11. Kitchenham B, Pfleeger S, Fenton N. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering* 1995; **21**(12):929–944.
12. Corazzon R. Descriptive and formal ontology, a resource guide to contemporary research. *Ontology: A Resource Guide for Philosophers*, 2006. Available at: http://www.formalontology.it/ [15 November 2006].
13. W3C. OWL Features. *W3C Recommendation 10 February 2004*. World Wide Web Consortium: Cambridge MA, 2006; 14 pp. Available at: http://www.w3.org/TR/owl-features/ [15 November 2006].
14. Chen H. Creative Commons. *Daml Tools for Supporting Intelligent Information Annotation, Sharing, and Retrieval*. University of Maryland in Baltimore County: Baltimore MD, 1 pp.
Available at: http://daml.umbc.edu/ontologies/cobra/ [15 November 2006].
15. Orme A, Yao H, Etzkorn L. Coupling metrics for ontology-based systems. *IEEE Software* 2006; **23**(2):102–108.
16. Briand L, Daly J, Wust J. A unified framework for cohesion measurement. *Empirical Software Engineering* 1998; **3**(1):65–115.
17. Briand L, Daly J, Porter V, Wust J. A comprehensive empirical validation of design measures for object-oriented systems. *Proceedings of the 5th International Software Metrics Symposium*. IEEE Computer Society Press: Los Alamitos CA, 1998; 246–257.
18. Alshayeb M, Li W. An empirical validation of object-oriented metrics in two different iterative software processes. *IEEE Transactions on Software Engineering* 2003; **29**(11):1043–1049.
19. Chapin N. An entropy metric for software maintainability. *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*. Western Periodicals: North Hollywood CA, 1989; 522–523.
20. Li W, Etzkorn L, Davis C, Talburt J. An empirical study of design evolution in an object-oriented system. *Information and Software Technology* 2000; **42**(6):373–381.
21. Li W, Henry S. Object-oriented metrics which predict maintainability. *Journal of Systems and Software* 1993; **23**(1):111–122.
22. Chidamber S, Kemerer C. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering* 1994; **20**(6):476–493.
23. Kolman B, Busby RC. *Discrete Mathematical Structures for Computer Science*. Prentice-Hall: Englewood Cliffs NJ, 1987; 401pp.
24. Cohen J. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum: Mahwah NJ, 1988; 567 pp.

AUTHORS' BIOGRAPHIES

Anthony M. Orme is an Assistant Professor at Athens State University in Athens AL. His primary research interests are in software engineering (software metrics), distributed systems, and knowledge engineering. He received his PhD in computer science from the University of Alabama in Huntsville.

DOI: 10.1002/smr



Haining (Irene) Yao is employed at the National Library of Medicine (NLM) at the National Institutes of Health (NIH), in Bethesda MD. Her research interests are in ontology development and metrics, knowledge representation, and software engineering (software reuse). She received her PhD degree in computer science from the University of Alabama in Huntsville.

Letha H. Etzkorn is an Associate Professor in the Computer Science Department at the University of Alabama in Huntsville. Her primary research interests are in software engineering (including program understanding, object-oriented software metrics, and artificial intelligence applications to software engineering) and mobile agents. She has Bachelor's and Master's Degrees in Electrical Engineering from the Georgia Institute of Technology, and a PhD in Computer Science from the University of Alabama in Huntsville.

DOI: 10.1002/smr