A Grammar of Dublin Core

Thomas Baker
GMD -- German National Research Center for Information Technology
Scientific Library and Publication Services
Schloss Birlinghoven
D-53754 Sankt Augustin, Germany
thomas.baker@gmd.de

"We begin with the sentence, because the sentence is the unit of discourse, because words can be classified only from their function in the sentence, and because the pupil should, from the outset, see that what determines the words in the sentence and the sentence itself is the thought."

Alonso Reed and Brainerd Kellogg, Higher Lessons in English, 1877

Dublin Core Made Simple

Dublin Core as a Language. Dublin Core is often presented as a modern form of catalog card -- a set of elements (and now qualifiers) that describe resources in a complete package. Sometimes it is proposed as an exchange format for sharing records among multiple collections. The founding principle that "every element is optional and repeatable" reinforces the notion that a Dublin Core description is to be taken as a whole. This paper, in contrast, is based on a much different premise: Dublin Core is a language. More precisely, it is a small language for making a particular class of statements about resources. Like natural languages, it has a vocabulary of word-like terms, the two classes of which -- elements and qualifiers -- function within statements like nouns and adjectives [1]; and it has a syntax for arranging elements and qualifiers into statements according to a simple pattern [2].

A Pidgin for Digital Tourists. Whenever tourists order a meal or ask directions in an unfamiliar language, considerate native speakers will spontaneously limit themselves to basic words and simple sentence patterns along the lines of "I am so-and-so" or "This is such-and-such". Linguists call this pidginization. In such situations, a small phrase book or
translated menu can be most helpful. By analogy, today's Web has been called an Internet Commons where users and information providers from a wide range of scientific, commercial, and social domains present their information in a variety of incompatible data models and description languages. In this context, Dublin Core presents itself as a metadata pidgin for digital tourists who must find their way in this linguistically diverse landscape. Its vocabulary is small enough to learn quickly, and its basic pattern is easily grasped. It is well-suited to serve as an auxiliary language for digital libraries.

**This Grammar.** This grammar starts by defining terms. It then follows a 200-year-old tradition of English grammar teaching by focusing on the structure of single statements (see the Reed and Kellogg quote above). It concludes by looking at the growing dictionary of Dublin Core vocabulary terms -- its registry, and at how statements can be used to build the metadata equivalent of paragraphs and compositions -- the application profile.

**Elements and Qualifiers**

**Vocabulary terms in general.** Strictly speaking, a Dublin Core element or qualifier is a unique identifier formed by a name (e.g., Title) prefixed by the URI of the namespace in which it is defined, as in http://dublincore.org/2000/03/13-dces#title. In this context, a namespace is a vocabulary that has been formally published, usually on the Web; it describes elements and qualifiers with natural-language labels, definitions, and other relevant documentation. Currently there are two namespaces for Dublin Core: the Dublin Core element set and the Dublin Core Qualifiers, denoted here by the conventional abbreviations dc: and dcq: [DCMI 1999, DCMI 2000]. In this paper, as in many application environments, the elements and qualifiers are referred to in a machine-readable short form, such as dc:title.

**Elements.** The fifteen elements of the Dublin Core element set are the defining feature of Dublin Core as a language. In their short form, the elements are dc:title, dc:creator, dc:subject, dc:description, dc:publisher, dc:contributor, dc:date, dc:type, dc:format, dc:identifier, dc:source, dc:language, dc:relation, dc:coverage, and dc:rights. These correspond to fifteen broadly defined properties of resources that are generally useful for searching across repositories in multiple domains.

**Qualifiers.** Qualifiers modify the properties of Dublin Core statements by specifying, in the manner of natural-language adjectives, "what kind" of subject, date, or relation. Qualifiers currently fall into two classes. **Encoding schemes** are pointers to contextual information or parsing rules that aid in the interpretation of an element value. For example, dcq:lcs h qualifies dc:subject to specify that the keywords are a Library of Congress Subject Heading, and dcq:iso8601 qualifies dc:date to specify that the string "2000-06-13" is formatted according to an international standard. **Element refinements** make a property more specific without extending its meaning, such as dcq:revised as a modifier of dc:date (yielding "date revised"). In July 2000, a DCMI Usage Committee recommended a batch of fifty-two qualifiers to exemplify these principles of qualification, but the qualifiers for Dublin Core statements may also come from other namespaces, as in the example yans:author below (where yans: is a hypothetical YetAnotherNameSpace).

**Elements and qualifiers defined in languages other than English.** Strictly speaking, elements and qualifiers are represented by machine-readable tokens that stand for general
concepts such as "title," "subject," and "date." The Dublin Core Metadata Initiative discusses and approves their definitions in English. In principle, however, they can be labelled and defined equally well in any other language, such as Dutch or Arabic or Thai. For example, dc:creator may be labeled "Creatore" in Italian, "Pencipta" in Bahasa Indonesian, or "Verfasser" in German [Baker 1998]. To date, the element set has been translated into twenty-six languages. Bear in mind as you read that although this grammar is written in English, a Japanese version could translate every English word here into Japanese -- all grammar terms and example sentences included -- except for the English-like names of the tokens themselves.

**Dublin Core Statements**

**Diagramming statements.** Since the 1870s, the grammar of sentences has been taught in (American) high schools using sentence diagrams [Reed and Kellogg 1886, House and Harman 1950, Warriner et al. 1973]. This style has a binary flavor -- the sentence baseline is intersected to divide the subject, "that of which something is said," from the predicate, that which is said of the subject. Within the predicate, a smaller line separates the object (in Dublin Core terms, the property) from the objective complement (in Dublin Core, the property value). This style is nicely expressive of Dublin Core because the qualifiers, hanging below the baseline on slanted lines, are visibly subordinate to the properties they modify (see Figures 1 and 2).

![Diagram of a Dublin Core statement](image)

**Parts of a Statement.** Dublin Core is in effect a class of statements of the pattern "Resource has property X," where "resource" is the implied subject; followed by an implied verb ("has"); followed by one of fifteen properties from the Dublin Core element set; followed by a property value -- an appropriate literal such as a person's name, a date, some words, or an URL. For example: "Resource has dc:creator 'Tom Baker,'" and "Resource has dc:date '2000-06-13'." Optional qualifiers may make the meaning of a property more definite, as in "Resource has dc:date dcq:revised '2000-06-13'."
**Principles of Qualification.** The qualification of Dublin Core properties is guided by a rule known colloquially as the Dumb-Down Principle. According to this rule, "a client should be able to ignore any qualifier and use the description as if it were unqualified. While this may result in some loss of specificity, the remaining element value (minus the qualifier) must continue to be generally correct and useful for discovery" [DCMI 2000]. Qualification is therefore supposed only to refine, not extend the semantic scope of a property. In borderline cases, qualification should not result in a literal that could be misleading.

**Appropriate Literals.** Whether a property value is "useful for discovery" is at the heart of the notion of appropriateness. A property value should be a string of an expected type -- usually, for example, some sort of name for dc:creator, dc:contributor, dc:publisher, or dc:title; a URL for dc:relation, dc:identifier, or dc:source; full-text sentences for dc:description; short text strings or keywords for dc:subject, dc:type, dc:format, and dc:language; and a recognizable combination of years, months, and days for dc:date. Both in theory and in practice, the range of expected data types varies from property to property; which types are appropriate for a given property is open to interpretation and debate (see below).

**Evaluating Statements.** To test whether a Dublin Core statement is conceptually solid, cover the qualifiers with your hand ("dumbing down"), read the statement above the line, and ask:
- Does it make sense?
- Is it factually and logically correct?
- Is the literal "appropriate" for the given property?

**Examples**

```
<table>
<thead>
<tr>
<th>Resource</th>
<th>has</th>
<th>DC:Date</th>
<th>&quot;2000-06-13&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DC:Revised</td>
<td>DCQ:ISO861</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>has</th>
<th>DC:Subject</th>
<th>&quot;Languages -- Grammar&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DC:Q:1234h</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 2: Some statements, diagrammed

Resource has dc:title 'A Grammar of Dublin Core.'
Does it make sense? Yes. Is it correct? Yes. Is the literal "appropriate"? Yes, a sequence of words is normal and expected for the property dc:title.

Resource has dcq:iso8601 dcq:revised dc:date '2000-06-13.'
This means that a resource was revised on 6 June 2000. The statement dumbs down to "Resource has dc:date '2000-06-13,'" which means that the date 6 June 2000 has something to do with the life-cycle of the resource. This is less specific than the qualified statement, but still correct.

Resource has dcq:lcsh dc:subject 'Languages -- Grammar.'
This says that the resource is about the subject "grammar of languages," and that these words are a controlled term from the Library of Congress Subject Headings. The statement dumbs down to "Resource has dc:subject 'Languages -- Grammar,'" which makes sense even if we do not know that the term comes from the Library of Congress.

Resource has yans:cerif dc:subject 'H352 Grammar, semantics, semiotics, syntax.'
This literal includes a language-independent abbreviation, "H352," which will be useful for applications that understand yans:cerif. The string "H352" may confuse some users, but otherwise does no harm.

Resource has yans:author dc:creator 'Tom Baker.'
Users of the yans: namespace will recognize that yans:author is being used here as an adjective modifying dc:creator, just as talk is an adjective modifying show in the phrase talk show. It is an awkward but correct way to say that Tom Baker is "the author sort of creator" -- i.e., the author -- of the resource. This statement dumbs down to "Resource has dc:creator 'Tom Baker,'" -- less specific but still correct.

Resource has dc:relation 'http://www.dlib.org/dlib/december98/12baker.html.'
The statement asserts that the resource is somehow related to an article in D-Lib Magazine (a URL is appropriate as a literal for dc:relation).

Some not-so-good examples

Resource has dc:creator 'name.given:Thomas; name.family:Baker; employer:GMD; contact:Schloss Birlinghoven, D-53754 Sankt Augustin.'
A reader can see what this compound or structured value is saying. But a search engine would need to know how to parse out the components and suppress the tags in order to index this cleanly, lest a search for creators named "Augustin" should yield false hits. Generally speaking, things like affiliations and addresses -- properties of the creator of a resource -- do not belong in Dublin Core statements about the resource itself. Metadata providers that used such compound values within specific usage communities could "speak Dublin Core" to the rest of the world by exporting just the name ("Resource has dc:creator 'Tom Baker'").

Resource has yans:creator 'Tom Baker.'
This statement is useful for applications that recognize the yans: namespace, but it is not a "Dublin Core" statement per se. If the yans: and dc: definitions of Creator were compatible, either the metadata provider or an indexing application could use a crosswalk to translate this into the Dublin Core statement "Resource has dc:creator 'Tom Baker.'"

Ongoing Issues
Fifteen fuzzy buckets. The properties of Dublin Core are like fifteen big buckets, and the rules about which types of literals may be placed in those buckets are somewhat fuzzy. This fuzziness is intentional -- the Internet is a diverse and chaotic place where a more disciplined, top-down approach to standardization is unrealistic, especially for use across multiple domains and languages. If the rules of Dublin Core were more precise, people would inevitably bend them. In the jargon of computer science, then, Dublin Core is "weakly typed" as a language. A search engine may find a variety of information types in any given bucket -- from URLs to non-textual, alphanumeric strings to full text in any language.

The "appropriateness" of literals. Requirements for the appropriateness of a literal are in practice somewhat contradictory. Ideally, a literal should be useful for discovery, which means it should make sense "as is" to the average user. Yet it should also be processable in an expected way by search engines. Programmers need to be know, for example, when to index on strings separated by white space, minus punctuation and stop words, when to expect a URL, and when to expect an alphanumeric date string. Some elements are particularly ambiguous in this regard. Dc:rights, for example, can be free text or a URL. Dc:coverage can be a place name, the name of a time period, a numeric identifier for a place or time, or even a compound value -- in effect, a miniature schema with multiple sub-components separated by semicolons or XML tags. Where such a range of data types is permissible, should it be acceptable to relax the Dumb-Down Principle? Should qualifiers in effect modify the expected data type of the literal? Or should the presence of, say, XML angle brackets be expected to trigger, automatically, a change in parsing algorithms? Would such a complexification of a property compensate for the corresponding loss of Dublin Core's overall simplicity? Or can the need for complex description be resolved in a broader framework, outside Dublin Core per se?

Application profiles. One broader framework for such a resolution is the application profile. As currently defined, application profiles are the metadata equivalent of regional idioms or creoles (complexified pidgins). Implementors who need an application language more expressive than a pidgin may combine elements and qualifiers from Dublin Core with elements from other namespaces into a richer vocabulary or embed them into a syntactically more sophisticated data model. Such linguistic innovation is considered by many people to be reasonable as long as implementors respect a distinction between namespaces, where elements and qualifiers are given standard definitions, and profiles, where elements from multiple namespaces are (only) reused, combined, adapted, and constrained [Heery 2000]. The profile, then, is the natural locus for full descriptions -- the catalog card or metadata package taken as a whole.

For example, the Collection Description Schema of the Research Support Libraries Programme (RSLP) in the UK uses dc:title -- officially defined as a "name given to the resource" -- but defines it more narrowly as a "name given to the collection." Alongside such Dublin Core elements, it uses elements from other namespaces, such as cld:accessConditions (for the hours of access and classes of permitted users) from a local "Collection Level Description" namespace. These elements are framed in a data model that specifies typical relationships between a collection, its individual items, a collector, an owner, a location, and the constituent parts of a collection -- each of which may be described with multiple attributes [RSLP]. An RSLP description does not talk just about information resources per se, but also about the people, organizations, and access frameworks related to those resources.
Developing profiles and coining new elements. Some working groups of DCMI are developing domain-specific profiles of Dublin Core, surveying the descriptive needs of domains such as education and government to determine an appropriate mix of Dublin Core elements and elements from other namespaces and perhaps to coin additional elements for concepts not covered in existing standards. These working groups need to consider that literals appropriate to domain specialists may not make much sense to general users, especially in statements that have been "dumbed down." As the example above makes clear, moreover, core elements are needed for classes of resources other than document-like objects, such as people and organizations (generically, agents) and spatially and temporally grounded events. Urgently required are data-model conventions for combining multiple entities within an application profile -- for example, to include an author's affiliation and address in the description for a resource -- and stable formats for the encoding of profiles as XML or RDF schemas.

Building a dictionary (registry) for Dublin Core. Historically, the standardization of national languages such as English has been helped by the compilation of dictionaries. Good dictionaries often strike a balance between prescribing guidelines for good style and describing a living language with examples of actual usage. Metadata languages like Dublin Core have hitherto been developed prescriptively, in standards committees, as there have been no convenient ways to track local innovations in usage and feed them back into the standardization process. However, several related developments are now enabling the collective construction of metadata dictionaries, or registries. The new Resource Description Framework (RDF) Schemas standard of the World Wide Web Consortium provides a format for publishing schemas that can be harvested by metadata search engines [W3C 2000]. Eric Miller has developed an open-source software toolkit for indexing a distributed corpus of RDF schemas as one huge database, with an interface for following hyperlinked cross-references between related terms in namespaces and profiles -- in effect, a metadata schema browser [EOR]. The Dublin Core Metadata Initiative is using this toolkit to manage its namespace [Open Metadata Registry], and a working group is formulating technical and policy guidelines for its ongoing management [DC-Registry]. A European project, SCHEMAS, is promoting the use of RDF schemas to help harmonize metadata practice among EU-funded projects and is using RDF to build a layer of annotated pointers to namespaces, profiles, and metadata activities generally [SCHEMAS].

Does your application speak Dublin Core? Pidgins are inherently limited in what they can express, but they are easy to learn and enormously useful. In real life, we talk one way to our professional colleagues and another way to visitors from other cultures. Our digital library applications need to do this as well. Simplicity and complexity are both appropriate, depending on context. If Dublin Core is too simple or generic to use as the native idiom of a particular application, its elements can be embedded in a richer local vocabulary from which pidgin statements can be generated or extracted as needed. This output should also be filtered to keep the fifteen buckets clear of encoding debris and semantic silt. One should treat digital tourists with courtesy and hide from them the complexities of a local application vocabulary or grammar. However sophisticated its local idiom may be, an application might also speak a pidgin that general users and generic search engines will understand. Simple, semantically clean, computationally obvious values will help us negotiate our way through a splendidly diverse and heterogeneous Internet.
Appendix: Dublin Core and RDF grammar compared

Directed Labelled Graphs. The Resource Description Framework (RDF), a relatively new standard of the World Wide Web Consortium, is emerging as an information model and encoding format of choice for metadata and application profiles that use Dublin Core [W3C 1999, W3C 2000]. RDF is a grammar for expressing relationships among resources located or represented somewhere on the Internet. These relationships are depicted graphically with Directed Labelled Graphs (DLGs), which use arcs (predicates expressing properties) to establish a relationship between multiple nodes (resources). Nodes are seen as subjects or objects depending on the direction of the arrow.

Figure 3: The general pattern of RDF statements ("triples")

RDF triples. In contrast to the binary flavor of the pedagogical English grammars cited above, the model of RDF statements is a tripartite one of subject, predicate, and object (see Figure 3). RDF statements, accordingly, are called "triples":

A property is a specific aspect, characteristic, attribute, or relation used to describe a resource. ... A specific resource together with the named property plus the value of that property for that resource is an RDF statement. These three individual parts of a statement are called, respectively, the subject, the predicate, and the object. The object of a statement (i.e., the property value) can be another resource or it can be a literal; i.e., a resource (specified by a URI) or a simple string or other primitive datatype defined by XML. In RDF terms, a literal may have content that is XML markup but is not further evaluated by the RDF processor [W3C 1999].

Figure 4 shows an RDF triple consisting of the subject "http://www.w3.org/Home/Lassila," predicate "has creator," and object "Ora Lassila."

Figure 4: Directed Labelled Graph of a triple

Subject of an RDF statement. The subject of an RDF statement is anything that can be named by a Universal
Resource Identifier (URI). In principle, this is anything from a Web page or museum artifact to an abstract concept or event. The subject of a Dublin Core statement, in contrast, is an anonymous "information resource," perhaps a "document-like object" [3]. In Dublin Core, the subject ("resource") is always implied, never named as in RDF. The RDF statement in Figure 4, then, would be expressed in Dublin Core as two statements: "Resource has dc:identifier 'http://www.w3.org/Home/Lassila'," and "Resource has dc:creator 'Ora Lassila'." As long as it is clear from the context that the two statements refer to the same resource -- for instance, they are both in the same record -- one can infer that Ora Lassila has created the Web page [4].

<table>
<thead>
<tr>
<th>English</th>
<th>Dublin Core</th>
<th>RDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>words (have classes)</td>
<td>vocabulary terms (have classes)</td>
<td>nodes and arcs</td>
</tr>
<tr>
<td>→ nouns</td>
<td>→ elements</td>
<td></td>
</tr>
<tr>
<td>→ adjectives</td>
<td>→ qualifiers</td>
<td></td>
</tr>
<tr>
<td>sentences (have parts)</td>
<td>statements (have parts)</td>
<td>statements (have parts)</td>
</tr>
<tr>
<td>→ subject</td>
<td>→ implied subject</td>
<td>→ subject</td>
</tr>
<tr>
<td>→ predicate</td>
<td>→ implied verb</td>
<td>→ predicate</td>
</tr>
<tr>
<td>→ → verb</td>
<td>→ → implied verb</td>
<td>→ → implied verb</td>
</tr>
<tr>
<td>→ → object</td>
<td>→ → property</td>
<td>→ → property</td>
</tr>
<tr>
<td>→ → objective complement</td>
<td>→ → property value</td>
<td>→ object (property value)</td>
</tr>
<tr>
<td>→ modifiers</td>
<td>→ → qualifiers</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Grammar Terms Compared

Predicate of an RDF statement. The predicate of an RDF statement is an implied verb plus a property. This is more restricted than the standard definition of a predicate in high-school English grammar, which includes (in effect) "everything to the right of the vertical intersector in a sentence diagram" -- that is, everything within a clause that is said about a subject. However, since most people have only the vaguest recollection of predicates from high school, this grammar avoids using the term at all.

Property value of RDF statements. In RDF, these are called objects. But again, there is risk of confusion with English grammar. In the sentence "This paper has the title 'A Grammar of Dublin Core'," some high-school English grammars see 'A Grammar of Dublin Core' as an objective complement with respect to the object of the sentence, "title" (see Figure 5). Compounding the confusion, computer scientists are oriented to objects of a much different sort, and the "resource" of a Dublin Core statement might even be a physical object. This grammar avoids the term object altogether.

The lack of qualifiers in RDF. In Dublin Core, qualifiers depend on and modify one of Dublin Core's fifteen elements (properties). The basic RDF model does not express this type of dependency. Properties may relate to another as narrower to broader terms (using the relation "SubPropertyOf"), as "Author" relates to "Creator." In native RDF, however, both "Author" and "Creator" are full properties in their own right.

The compatibility of Dublin Core and RDF. The differences between Dublin Core and RDF outlined above are largely terminological; at issue is what the parts of a statement such as 'Resource has Property X' should be called. The difficulty of choosing the right words for this grammar should not obscure the basic compatibility and complementarity of Dublin Core and RDF. RDF offers a general model for statements, while Dublin Core offers a particular type of pidgin-like statement about information resources and privileges a small set of
special words. RDF offers a specific encoding in XML for expressing its conceptual model, while Dublin Core is by design independent of any particular encoding format. RDF, then, is just one of the possible information models that can use tokens from Dublin Core, while Dublin Core is just one of the languages expressible in RDF.

Footnotes

[1] It would be desirable to have a word for Dublin Core vocabulary terms in general -- elements and qualifiers as a whole -- just as natural-language nouns, verbs, adjectives, and conjunctions are all called words. Unfortunately, entities, concepts, and symbols are all too abstract and vague; words are associated too closely with natural language; and lexemes are too obscure. The first draft of this paper spoke generically of elements and distinguished between core elements and qualifiers, but veterans of Dublin Core found this confusing. The next draft introduced tokens, which conveys the notion that Dublin Core vocabulary terms stand for general concepts that are defined and labeled in many natural languages, but people also found this confusing. Vocabulary terms seems a bit cumbersome, so I avoid the term whenever possible and speak simply of elements and qualifiers.

[2] Early in the workshop series, before it was formally called a Metadata Initiative, the Dublin Core effort was declared to be primarily about semantics in contrast to syntax, and the latter was declared to be out of scope. In that context, however, syntax referred to the encoding of metadata in HTML, database, and (later) XML or RDF formats; it involved questions such as which tags to place where, within what angle brackets or punctuation, and how to group or nest related elements. This grammar, in contrast, presents the syntax of Dublin Core statements in a linguistic sense, as the rules governing how the words of a sentence are related to each other -- which words modify other words, and which words are of central importance in the statement.

[3] In practice, the reasoning behind this is circular: Dublin Core properties are appropriate for any entity that has such properties. This circularity has spared us a perhaps futile attempt to seek philosophical consensus on a universal ontology of entity classes and allowed us to get on with the task of describing whatever it is we are describing.

[4] Dan Brickley points out that software tools could exploit additional information about entities and vocabularies to translate sequences of RDF-encoded Dublin Core statements into a natural-language style that flows more elegantly and reads less like a pidgin.

Acknowledgements

I have discussed this paper with many colleagues from both DCMI and other communities. Special thanks to Bill Arms, Jose Borbinha, Dan Brickley, David Bearman, Rachel Heery, Diane Hillmann, Erik Jul, Carl Lagoze, Sigfrid Lundberg, Eric Miller, Paul Miller, Andy Powell, Shigeo Sugimoto, and Stuart Weibel for valuable comments on the drafts. Many thanks to Michael Kasper for advice regarding the literature on English grammar.

References


[DCMI 2000] Dublin Core Metadata Initiative, "Dublin Core Qualifiers,"

[DC-Registry] Dublin Core Registry Working Group,
http://www.mailbase.ac.uk/lists/dc-registry/.

[EOR] EOR Tookit: Web-based search interface tools for RDF RDMS systems,

http://www.ariadne.ac.uk/issue25/app-profiles.

[House and Harman 1950] Homer C. House and Susan Emelyn Harman, Descriptive English

[Open Metadata Registry] The Open Metadata Registry,
http://wip.dublincore.org/registry/OpenRegistry.

[Reed and Kellogg 1886] Alonso Reed and Brainerd Kellogg, Higher lessons in English, New
York: Clark and Maynard, 1886 [1877]; Delmar (New York): Scholars' Facsimiles and


[SCHEMAS] SCHEMAS: A Forum for Metadata Schema Implementers,

Model and Syntax Specification," [W3C Recommendation],
http://www.w3.org/TR/REC-rdf-syntax.

Schema Specification 1.0," [W3C Candidate Recommendation],
http://www.w3.org/TR/WD-rdf-schema.

[Warriner et al. 1973] John E. Warriner et al., English Grammar and Composition, Third