SWD WG face-to-face, 8-9 October, Amsterdam

2007-10-08 Monday
    AM   RDFa
    PM   SKOS
          -- Labeling properties (Alistair)
          -- Semantic relation properties (Sean)
    16:00 Vocabulary Management (Vit, with Elisa remotely)
    17:00 Review of Cool URIs (Michael remotely)

2007-10-09 Tuesday
    AM   SKOS
          -- Concept semantics (Antoine)
          -- Label relations (Guus)
          -- Drawing the picture (all)
    PM   Recipes

Agenda
   www.w3.org/2006/07/SWD/wiki/AmsterdamAgenda
   -- cached as e:/u/folders/AMSTERDAM/AmsterdamAgenda.pdf

SKOS Labeling properties (Alistair)
   http://www.w3.org/2006/07/SWD/track/issues/31
   http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0170.html
   http://purl.org/net/skos/2007/10/f2f/labelling-properties.html

SKOS Semantic Relation Properties (Sean)
   http://www.w3.org/2006/07/SWD/track/issues/44

SKOS Concept Semantics (Antoine)
   www.w3.org/2006/07/SWD/wiki/SkosDesign/ConceptSemantics - Antoine's issue summary
   -- cached as e:/u/folders/AMSTERDAM/ConceptSemantics.pdf

SKOS Semantic Relation Properties (Sean)

SKOS Label relations (Guus)
   purl.org/net/skos/2007/10/f2f/label-relations.html
   -- cached as e:/u/folders/AMSTERDAM/Skos-label-relations.pdf
   www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/
   -- cached as e:/u/folders/AMSTERDAM/RelationshipsBetweenLabels.pdf
   www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/ProposalFour
   -- cached as e:/u/folders/AMSTERDAM/ProposalFour.pdf
   www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/ProposalThree
   -- cached as e:/u/folders/AMSTERDAM/ProposalThree.pdf
   http://lists.w3.org/Archives/Public/public-swd-wg/2007Sep/0013.html
   http://lists.w3.org/Archives/Public/public-swd-wg/2007Oct/0019.html

Vocabulary Management (Vit, Elisa)
   www.w3.org/2006/07/SWD/wiki/VocabMgtDraft
   -- cached as e:/u/folders/AMSTERDAM/VocabMgtDraft.pdf

Recipes for publishing RDF vocabularies (Jon)
   madcreek.com/w3c/swdwg_f2f_issues.html - Current list of issues
   -- cached as e:/u/folders/AMSTERDAM/Recipes-consolidated-issues.pdf
   www.w3.org/2006/07/SWD/wiki/BestPracticeRecipe6 - draft of recipe 6
   -- cached as e:/u/folders/AMSTERDAM/Recipes-recipe6.pdf

SKOS specifications
   www.w3.org/2006/07/SWD/wiki/SKOS/Semantics
   -- cached as e:/u/folders/AMSTERDAM/Semantics.pdf
   www.w3.org/TR/skos-ucr/
   -- cached as e:/u/folders/AMSTERDAM/SKOSUseCases.pdf
   http://www.w3.org/TR/swbp-skos-core-spec/
   www.w3.org/TR/swbp-skos-core-guide/ - not included in packet (Adobe format problems)

RDFa (Mark Birbeck)
   www.w3.org/2006/07/SWD/RDFa/primer/20070918/   - RDFa Primer
   -- cached as e:/u/folders/AMSTERDAM/RDFa-primer.pdf
   http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070927 - XHTML1.1 RDFa Syntax
   http://lists.w3.org/Archives/Public/public-swd-wg/2007Oct/0033.html - Diego's review

Review of Cool URIs (Michael)
www.w3.org/2006/07/SWD/wiki/ReviewCoolURIs
-- cached as e:/u/folders/AMSTERDAM/ReviewCoolURIs.pdf
gnowsis.opendfki.de/repos/gnowsis/papers/2006_11_concepturi/html/cooluris_sweo_note.html
-- cached as e:/u/folders/AMSTERDAM/Cool-uris.pdf

Agenda - Semantic Web Deployment WG face-to-face

8-9 October, Amsterdam

- This draft agenda: http://www.w3.org/2006/07/SWD/wiki/AmsterdamAgenda
- Registration: http://www.w3.org/2002/09/wbs/39408/swd-ftf-adam-registration/
- Agenda and logistics: http://www.few.vu.nl/~aisaac/swd/
- Remote participation (telecon) info
- Expected:
  - WG members: Tom, Guus, Diego, Ed, Justin, Antoine, Jon, Alistair, Sean
  - Guests: Steven Pemberton (RDFa), Mark Birbeck (RDFa), Ivan Herman
  - By irc/telecon: Elisa, Ben, Michael, Simone, Daniel, Ralph

2007-10-08 Monday

- 9:00-13:00 (0700 - 1000 UTC)
  - RDFa
- 14:00-18:00 (1100 - 1600 UTC)
  - SKOS: Labeling properties (Alistair) - Tom chairs
  - SKOS: Semantic relation properties (Sean)
  - 16:00 Vocabulary Management (Vit, with Elisa remotely)
  - 17:00 Review of Cool URIs (Michael remotely)

2007-10-09 Tuesday

- AM
  - SKOS Concept semantics (Antoine)
  - SKOS Label relations (Guus)
  - SKOS: Drawing the picture (all)
- PM
  - Recipes (Jon)

RDFa (Mark Birbeck, 3 hours)

- Required reading
  - RDFa Primer
  - XHTML1.1 RDFa Syntax
- PROPOSAL: that the SWD WG agree to bring the following RDFa documents to REC track:

    - XHTML1.1 RDFa Syntax
    - Supporting documentation
        - Test cases - 1
        - Test cases - 2
        - Implementations
- PROPOSAL: that the SWD WG approve publication of RDFa-in-XHTML-1.1 Syntax as a first Public Working Draft
- PROPOSAL: that the SWD WG approve publication of a new Public Working Draft of the RDFa-in-XHTML-1.1 Primer.

SKOS

General reading

- http://www.w3.org/2006/07/SWD/wiki/SKOS/Semantics
- http://www.w3.org/TR/skos-ucr/
- http://www.w3.org/TR/swbp-skos-core-spec/
- http://www.w3.org/TR/swbp-skos-core-guide/

Three central issues (Alistair, 2007-06-26 posting)

- "'Issue-31 - BasicLexicalLabelSemantics"' (open): defining the semantics of the three basic labelling properties, skos:prefLabel, skos:altLabel and skos:hiddenLabel -- see Topic "Labeling Properties" (Alistair)
- "'Issue-44 - BroaderNarrowerSemantics"' (open): defining the semantics of skos:broader and skos:narrower -- see Topic "Semantic Relation Properties" (Sean)
- "'Issue-54 ConceptSemantics"' (open): defining the semantics of skos:Concept -- see Topic "Concept Semantics" (Antoine)

"Labeling properties" (Alistair)

Background information and discussion for this topic is given in the following document:

- http://purl.org/net/skos/2007/10/f2f/labelling-properties.html

This document is required reading, along with the SKOS Core Guide (section: Labelling Properties) and the SKOS Core Vocabulary Specification (only sections skos:prefLabel, skos:altLabel and skos:hiddenLabel).

"Semantic Relation Properties" (Sean)

What are the semantics of skos:broader and skos:narrower? There are several open questions:

- Are they transitive/intransitive?
- Are they reflexive/irreflexive?

- Is the transitive closure irreflexive (i.e. a cycle is a contradiction)?

The transitivity of the relationships could be enshrined in the vocabulary, e.g. through making skos:broader and skos:narrower instances of owl:transitiveProperty. Or we can say nothing and leave it up to applications to decide whether to treat the relationships as transitive. Specific vocabularies may also be able to define transitive subproperties of broader/narrower if required.

Similarly for reflexivity, although this requires expressivity outside of OWL (but which is in the proposed OWL 1.1 spec).

Questions to consider:

- Which use cases exhibit a necessity for transitive broader/narrower?
- If the relationships are transitive, what impact does this have on presentations? For example, hierarchical views may want to only show "direct" broader terms, as is the case with most tools and sub/superclass relationships.
- Do we expect interaction between skos:related and skos:broader/narrower. For example, if x related y and y broader z, do we expect x related z? Again, such functionality is now supported in OWL 1.1 extensions.

Potential solutions:

- Say nothing about characteristics of broader/narrower.
- Include characteristics of broader/narrower.

Relevant issues:

- '"Issue-44 - BroaderNarrowerSemantics"' (open)

Background reading:

- OWL 1.1 Member Submission
- "SKOS Queries" Thread on List

Related material:

- Discussion about SKOS Validator

"Concept Semantics" (Antoine)

Problems to discuss:

- Defining the semantics of skos:Concept
- Mixing with OWL concepts

Required reading:

- ConceptSemantics - Antoine's issue summary

Relevant issues:

- "'Issue-54 ConceptSemantics'" (open)

What are the semantics of skos:Concept?

skos:Concept is currently declared as { skos:Concept rdf:type rdfs:Class . } ... is this ok? Is this enough? What about the OWL universe?

"Label relations" (Guus)

SKOS allows to represent semantic relationships (broader, related) between concepts. It also allows to represent relationships between concepts and labels (prefLabel, altLabel). However, there is nothing proposed in SKOS to capture links between labels themselves, a configuration which sometimes happens in concept schemes.

Relevant issues:

- "'Issue-26 - RelationshipsBetweenLabels'" (open)
  - Issue: SKOS allows to represent semantic relationships (broader, related) between concepts. It also allows to represent relationships between concepts and labels (prefLabel, altLabel). However, there is nothing proposed in SKOS to capture links between labels themselves, a configuration which sometimes happens in concept schemes.

Required background reading:

- SkosDesign RelationshipsBetweenLabels

Three options will be discussed in Amsterdam

- Alistair's "Minimal Label Relation" proposal
- Guus "Simple Extension" proposal
- Guus "remove range restriction for skos:prefLabel" proposal

Drawing the picture (discussion leader?)

Problems to discuss:

- Are relationships part of a concept scheme?
- Are the concepts part of a concept scheme?
- Capture underlying intuitions: UML diagrams? Blobs and lines?
- What do we mean by "containment", "aggregation"...?
- Are relationships of a concept somehow part of the concept?
- Can concepts "exist" independently of a scheme?
- Goal: see how things fit together in the larger picture.

Relevant issues:

- ISSUE-36 http://www.w3.org/2006/07/SWD/track/issues/36 - ConceptSchemeContainment (open)

    - Guus thoughts on ISSUE-36: http://lists.w3.org/Archives/Public/public-swd-wg/2007Jul/0164.html
    - Alistair: http://www.w3.org/2006/07/SWD/wiki/SKOS/UserStories useful for ISSUE-36?

Reading:

- http://lists.w3.org/Archives/Public/public-swd-wg/2007Aug/0016.html
- http://isegserv.itd.rl.ac.uk/public/ajm65/dc2007/tutorial.pdf
- http://zthes.z3950.org/model/zthes-model-1.0.html
- bs8273 - UML diagrams available?
- Elisa/Clay/Ed/Jon: OMG, ISO 1087...?

Vocabulary Management (Vit locally, Elisa remotely): 60 minutes

To be discussed late afternoon Monday 16:00+ so that Elisa can join.

Discussion points

- VocabMgtDraft structure, section requirements, status (Elisa, 30 minutes)
- Version management of ontologies (Vit, 30 minutes)
    - Vit will present:
        - Versioning Questionnaire Evaluation - preliminary report on the results at [X]
        - Reference Implementation of RDF-based Ontology Versioning - SemVersion
          (partially funded by the Knowledge Web EU project); the tool description in a paper at [Y], the tool web site at [Z]
        - Versioning best practices based on early results of diff capabilities and multi-version ontology reasoning
    - The WG should discuss which of this topics should be reflected in VocabMgtDraft (and how)

Required Reading

- http://www.w3.org/2006/07/SWD/wiki/VocabMgtDraft

Suggested Reading

- http://lists.w3.org/Archives/Public/public-semweb-lifesci/2007Aug/0001.html
- https://gnowsis.opendfki.de/repos/gnowsis/papers/2006_11_conceptturi/html/index.html
- http://esw.w3.org/topic/HCLSIG_BioRDF_Subgroup/Tasks/URI_Best_Practices
    - Daniel is liaison, SWD may be asked for review before publication
- http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/drafts/LinkedDataTutorial/
- [X] http://smile.deri.ie/resources/2007/09/ovs-eval.pdf (10 pages, only 3 of them

significantly relevant for the group)

- [Y] http://xam.de/2006/10-SemVersion-ICIW2006.pdf (8 pages)

Background reading:

- [Z] http://semweb4j.org/site/semversion/

Recipes for publishing RDF vocabularies (Jon)

- Required reading:
  - Current list of issues: http://madcreek.com/w3c/swdwg_f2f_issues.html
  - Draft of Recipe 6: http://www.w3.org/2006/07/SWD/wiki/BestPracticeRecipe6
- Suggested reading:
  - "How to publish linked data" tutorial by Bizer, Cyganiak and Heath: http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/
  - "Cool URIs for the Semantic Web": http://www.dfki.uni-kl.de/%7Esauermann/2006/11/cooluris/
  - TAG finding: "Associating Resources with Namespaces": http://www.w3.org/2001/tag/doc/nsDocuments/
  - Latest Working Draft (2006-03-14): http://www.w3.org/TR/swbp-vocab-pub/
- Discussion points:
  - Discuss and resolve an strategy to close ISSUE-58:
    - Should we fix the general case?
    - Is it enough to fix the most common cases?
  - Discuss the current draft of Recipe 6: http://www.w3.org/2006/07/SWD/wiki/BestPracticeRecipe6
    - Do we have the time now, or will someone(s) volunteer to review?
  - Briefly review status of issues 19 to 24:
    - ISSUE-19 http://www.w3.org/2006/07/SWD/track/issues/19 - Recipes should supply a general server configuration template (open)
      - Should the recipes supply this in order to better support non-Apache servers?
    - ISSUE-20 http://www.w3.org/2006/07/SWD/track/issues/20 - Online server testing (open) There's a sourceforge project: http://vapour.sourceforge.net/
      - If we develop a web interface for it, who can host it?
    - ISSUE-22 http://www.w3.org/2006/07/SWD/track/issues/22 - Questioning reference to 'IE6 hack' (open) in Content Negotiation section http://www.w3.org/TR/swbp-vocab-pub/#negotiation
      - Should we "remove from the document any suggestion that the reader should delete the directive and just insert an explanation as to why it's needed;"?
    - ISSUE-23 http://www.w3.org/2006/07/SWD/track/issues/23 - There should be some discussion of alternatives to .htaccess (open)
      - Should we do this?

- Discuss the relationship between the Recipes, current issues, and the more recently published 'Linked Data', 'Cool URIs', and 'Associating Resources' documents: http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/, http://www.dfki.uni-kl.de/%7Esauermann/2006/11/cooluris/, http://www.w3.org/2001/tag/doc/nsDocuments/
  - Should we expand or revise the discussion of the content of documents returned by dereferencing a URI in the
    - minimum requirements: http://www.w3.org/TR/swbp-vocab-pub/#minimumrequirements
    - URI namespaces: http://www.w3.org/TR/swbp-vocab-pub/#naming
    - Appendix B. Vocabulary URIs based on a 303-redirect service: http://www.w3.org/TR/swbp-vocab-pub/#redirect
  - This discussion also impacts these issues...
    - ISSUE-24 http://www.w3.org/2006/07/SWD/track/issues/24 - Additional text explaining redirect choices in the recipes (open)
    - ISSUE-30 http://www.w3.org/2006/07/SWD/track/issues/30 - Determine how and if RDDL relates to the Recipes (raised)
    - ISSUE-60 http://www.w3.org/2006/07/SWD/track/issues/60 - Guidelines needed for proper construction of vocabulary scheme and 'term' URIs (raised)

SKOS issues (as of 2007-08-21)

- Open
  - ISSUE-26 - RelationshipsBetweenLabels (open)
  - ISSUE-27 - AnnotationOnLabel (open)
  - ISSUE-31 - BasicLexicalLabelSemantics (open)
  - ISSUE-36 - ConceptSchemeContainment (open)
  - ISSUE-44 - BroaderNarrowerSemantics (open)
  - ISSUE-54 - ConceptSemantics (open)
- Raised
  - ISSUE-32 - ConceptSchemeLabellingInteractions (raised)
  - ISSUE-35 - RulesAndConformance (raised)
  - ISSUE-37 - SkosSpecialization (raised)
  - ISSUE-39 - ConceptualMappingLinks (raised)
  - ISSUE-40 - ConceptCoordination (raised)
  - ISSUE-41 - UseLangTagsInExamples (raised)
  - ISSUE-45 - NaryLinksBetweenDescriptorsAndNonDescriptors (raised)
  - ISSUE-46 - IndexingAndNonIndexingConcepts (raised)
  - ISSUE-47 - MappingProvenanceInformation (raised)
  - ISSUE-48 - IndexingRelationship (raised)
  - ISSUE-49 - LexicalMappingLinks (raised)
  - ISSUE-50 - CompatibilityWithDC (raised)

    - ISSUE-51 - CompatibilityWithISO11179 (raised)
    - ISSUE-52 - CompatibilityWithISO2788 (raised)
    - ISSUE-53 - CompatibilityWithISO5964 (raised)
    - ISSUE-56 - ReferenceSemanticRelationshipSpecializations (raised)
- Postponed
    - ISSUE-38 - CompatibilityWithOWL-DL (postponed)

SKOS requirements (as of 2007-08-21)

- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-AnnotationOnLabel
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-CompatibilityWithDC
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-CompatibilityWithISO11179
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-CompatibilityWithISO2788
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-CompatibilityWithISO5964
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-CompatibilityWithOWL-DL
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-ConceptCoordination
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-ConceptSchemeContainment
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-ConceptSchemeExtension
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-ConceptualMappingLinks
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-ConceptualRelations
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-ConsistencyChecking
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-GroupingInConceptHierarchies
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-IndexingAndNonIndexingConcepts
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-IndexingRelationship
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-LabelRepresentation
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-LexicalMappingLinks
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-MappingProvenanceInformation
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-MultilingualLexicalInformation
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-RelationshipsBetweenLabels
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-SkosSpecialization
- http://www.w3.org/TR/2007/WD-skos-ucr-20070516/#R-TextualDescriptionsForConcepts

ISSUE-31

BasicLexicalLabelSemantics

State:
OPEN

Product:
SKOS

Raised by:
Alistair Miles

Opened on:
2007-03-19

Description:

```
Can a resource have two "preferred lexical labels"? Can a lexical label be both
"preferred" and "alternative" for the same resource? If a lexical label is
"hidden", can it also be "preferred" or "alternative" for the same resource?

See also <http://www.w3.org/2006/07/SWD/wiki/SkosDesign/BasicLexicalLabelSemantics>

Imagine the SKOS vocabulary consisted only of skos:prefLabel, skos:altLabel and
skos:hiddenLabel. I.e. forget everything else for the moment.

Now, consider the following RDF graph:

@prefix ex: <http://www.example.com/examples#>.
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.

ex:foo skos:prefLabel "foo"@en.
ex:foo skos:prefLabel "bar"@en.

The resource ex:foo has two different preferred lexical labels in the same
language. Is this graph inconsistent?

What about the following:

@prefix ex: <http://www.example.com/examples#>.
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.

ex:foo skos:prefLabel "foo"@en.
```

```
ex:foo skos:altLabel "foo"@en.

The label "foo"@en is given as both a preferred and an alternative lexical label
for the resource ex:foo. Is this graph inconsistent?

Consider another graph:

@prefix ex: <http://www.example.com/examples#>.
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.

ex:foo skos:prefLabel "foo"@en.
ex:foo skos:hiddenLabel "foo"@en.

The label "foo"@en is given as both a preferred and a hidden lexical label for
the resource ex:foo. Is this graph inconsistent?

Finally, consider:

@prefix ex: <http://www.example.com/examples#>.
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.

ex:foo skos:altLabel "foo"@en.
ex:foo skos:hiddenLabel "foo"@en.

The label "foo"@en is given as both an alternative and a hidden lexical label
for the resource ex:foo. Is this graph inconsistent?
```

Related emails:

1. ISSUE-31: BasicLexicalLabelSemantics (from dean+cgi@w3.org on 2007-03-19)
2. RE: [SKOS] Possible issue: Uniqueness of skos:prefLabel [was Re: [SKOS] inconsistency between Guide and Specification (from A.J.Miles@rl.ac.uk on 2007-03-19)
3. [SKOS] central issues (from A.J.Miles@rl.ac.uk on 2007-06-26)
4. Re: [SKOS] central issues (from bernard.vatant@mondeca.com on 2007-06-26)
5. [SKOS] ISSUE-31 BasicLexicalLabelSemantics proposed resolution (from A.J.Miles@rl.ac.uk on 2007-06-26)
6. [SKOS] question about SKOS semantics document (was Re: [SKOS] ISSUE-31 BasicLexicalLabelSemantics proposed resolution) (from aisaac@few.vu.nl on 2007-06-26)
7. Re: [SKOS] ISSUE-31 BasicLexicalLabelSemantics proposed resolution (from aisaac@few.vu.nl on 2007-06-26)
8. Re: [SKOS] ISSUE-31 BasicLexicalLabelSemantics proposed resolution (from bernard.vatant@mondeca.com on 2007-06-27)
9. Re: [SKOS] central issues (from aisaac@few.vu.nl on 2007-06-27)
10. Re: [SKOS] central issues (from bernard.vatant@mondeca.com on 2007-06-27)
11. Re: [SKOS] central issues (from schreiber@cs.vu.nl on 2007-07-02)
12. [ALL] agenda July 3 telecon 1500 UTC (from schreiber@cs.vu.nl on 2007-07-02)
13. Re: [ALL] agenda July 3 telecon 1500 UTC (from Bernard.Horan@sun.com on 2007-07-02)
14. Re: [SKOS] central issues (from jphipps@madcreek.com on 2007-07-02)
15. Re: [SKOS] central issues (from aisaac@few.vu.nl on 2007-07-03)
16. Re: [SKOS] central issues (from jphipps@madcreek.com on 2007-07-03)
17. Re: [ALL] agenda July 3 telecon 1500 UTC (from ekendall@sandsoft.com on 2007-07-03)
18. Re: [ALL] agenda July 3 telecon 1500 UTC (from simone.onofri@gmail.com on 2007-07-03)
19. Re: [ALL] agenda July 3 telecon 1500 UTC (from juth@loc.gov on 2007-07-03)
20. Re: [SKOS] central issues (from schreiber@cs.vu.nl on 2007-07-04)
21. belated regrets (was RE: [ALL] agenda July 3 telecon 1500 UTC) (from A.J.Miles@rl.ac.uk on 2007-07-04)
22. Minutes from July 3 tcon (from rubin@med.stanford.edu on 2007-07-05)
23. [ALL] Agenda - July 10 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-07-09)
24. Re: [ALL] Agenda - July 10 2007 SWD telecon - 1500 UTC (from ekendall@sandsoft.com on 2007-07-09)
25. Re: [ALL] Agenda - July 10 2007 SWD telecon - 1500 UTC (from cred@loc.gov on 2007-07-09)
26. [ALL] SWD telecon - July 17 2007 - agenda (from baker@sub.uni-goettingen.de on 2007-07-16)
27. Re: [ALL] SWD telecon - July 17 2007 - agenda (from baker@sub.uni-goettingen.de on 2007-07-16)
28. [ALL] agenda July 24 telecon - 1500 UTC (from schreiber@cs.vu.nl on 2007-07-23)
29. Re: [ALL] agenda July 24 telecon - 1500 UTC (from ben@adida.net on 2007-07-23)
30. Regrets - Re: [ALL] agenda July 24 telecon - 1500 UTC (from juth@loc.gov on 2007-07-23)
31. [ALL] agenda Aug 7 telecon - 1500 UTC (from schreiber@cs.vu.nl on 2007-08-07)
32. regrets for Aug 7 telecon - 1500 UTC (from rubin@med.stanford.edu on 2007-08-06)
33. Agenda - Aug 21 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-08-20)

34. Re: Agenda - Aug 21 2007 SWD telecon - 1500 UTC (from Bernard.Horan@sun.com on 2007-08-20)
35. Scribing - Re: Agenda - Aug 21 2007 SWD telecon - 1500 UTC (from juth@loc.gov on 2007-08-21)
36. [ALL] SWD face-to-face - draft agenda (from baker@sub.uni-goettingen.de on 2007-08-22)
37. RE: [ALL] SWD face-to-face - draft agenda (from A.J.Miles@rl.ac.uk on 2007-08-24)
38. [ALL] agenda 28 Aug telecon - 1500 UTC (from schreiber@cs.vu.nl on 2007-08-27)
39. Re: [ALL] SWD face-to-face - draft agenda (from aisaac@few.vu.nl on 2007-08-27)
40. Re: [ALL] agenda 28 Aug telecon - 1500 UTC (from ekendall@sandsoft.com on 2007-08-27)
41. Re: [ALL] agenda 28 Aug telecon - 1500 UTC (from Bernard.Horan@sun.com on 2007-08-28)
42. Agenda - Sep 04 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-03)
43. Meeting record: 2007-09-04 (from michael.hausenblas@joanneum.at on 2007-09-05)
44. LATE REGRETS - Re: Meeting record: 2007-09-04 (from juth@loc.gov on 2007-09-05)
45. [ALL] agenda 11 Sep telecon - 1500 UTC (from schreiber@cs.vu.nl on 2007-09-11)
46. RE: [ALL] agenda 11 Sep telecon - 1500 UTC (from A.J.Miles@rl.ac.uk on 2007-09-11)
47. RE: [SKOS] Amsterdam topic 'Labeling properties' (from A.J.Miles@rl.ac.uk on 2007-10-02)

Related notes:

2007-03-19: Issues Sandbox: http://www.w3.org/2006/07/SWD/wiki/SkosIssuesSandbox#head-5593998a47bf818e6bbcf40d289d95f6390c320a

[SKOS] ISSUE-31 BasicLexicalLabelSemantics proposed resolution

From: Miles, AJ \(Alistair\) <A.J.Miles@rl.ac.uk>
Date: Tue, 26 Jun 2007 19:23:11 +0100
Message-ID: <677CE4DD24B12C4B9FA138534E29FB1D02EC0BF3@exchange11.fed.cclrc.ac.uk>
To: "SWD WG" <public-swd-wg@w3.org>
Cc: <public-esw-thes@w3.org>

Hi all,

I'd like to propose the following section of the SKOS Semantics as a resolution for [ISSUE-31]:

[1] <http://www.w3.org/2006/07/SWD/wiki/SKOS/Semantics/Labelling?action=recall&rev=5>

This proposal deals with the intuitive semantics of "preferred", "alternative" and "hidden" at the syntactic level, which avoids complicated semantics conditions and allows for error-tolerant strategies to be defined.

This idea of handling some conditions at a syntactic level is new to me, and I have very limited experience of writing normative specifications of application behaviour, so I'd very much welcome comments and thoughts on this proposal.

Cheers,

Alistair.

[ISSUE-31] <http://www.w3.org/2006/07/SWD/track/issues/31>

--
Alistair Miles
Research Associate
Science and Technology Facilities Council
Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Didcot
Oxfordshire OX11 0QX
United Kingdom
Web: http://purl.org/net/aliman
Email: a.j.miles@rl.ac.uk
Tel: +44 (0)1235 445440

Received on Tuesday, 26 June 2007 18:23:31 GMT

SWDWG Amsterdam F2F October 2007

Topic: SKOS Labelling Properties

This document gives background information and suggested resolutions for the "Labelling Properties" topic at the October 2007 SWDWG F2F meeting in Amsterdam.

Latest Version: http://purl.org/net/skos/2007/10/f2f/labelling-properties.html

$Revision: 1.2 $ on $Date: 2007/10/02 12:09:21 $

Contents

Introduction

Scope

This topic concerns three URIs from the SKOS vocabulary:

- skos:prefLabel
- skos:altLabel

- skos:hiddenLabel

These URIs denote SKOS's "lexical labelling properties". This topic does not include or concern or have any dependencies on any other URIs from the SKOS vocabulary.

Goals

I suggest our goals for this topic be to agree:

1. a semantics for these three URIs, and
2. how to specify those semantics.

Background

The most recent specifications concerning these properties are the section on "Labelling Properties" in [SKOS-GUIDE] (ignore the sub-section on symbolic labelling) and the relevant 3 sections of the properties table in [SKOS-SPEC].

A while ago I raised [ISSUE-31], which asks some questions about the intended semantics of these three properties.

Approach

You would have thought three simple properties like these would be easy to deal with, but this topic is surprisingly complicated, and takes us deep into the "nitti gritti" of both RDF and OWL!

The intended semantics are -- I think -- fairly obvious, and agreeing informally on the semantics should (hopefully) be easy.

However, the issue is complicated because there are:

- several different possible ways to formally specify those semantics, and
- several ways to implement applications which are aware of those semantics.

To try and make things easier, I've broken this topic down into sub-topics below, each of which should be the focus of a decision by the WG. Some topics discuss what the semantics should be, other topics discuss alternatives for formally specifying and implementing those semantics:

Sub-Topic A: Range Semantics?

The three SKOS lexical labelling properties were, originally, intended to be used with RDF plain literals in the object position of a triple. (RDF plain literals are formally defined in [RDF-CONCEPTS].)

So, for example, [SKOS-GUIDE] gives the following example:

```
ex:shrubs
  skos:prefLabel "shrubs"@en;
  skos:altLabel "bushes"@en;
  skos:prefLabel "arbuste"@en;
  skos:altLabel "buisson"@fr.
```

I.e. the notion of a "lexical label" in SKOS was, originally, tightly coupled to the notion of a plain literal in RDF. The phrase "lexical label" was chosen, rather than e.g. "plain literal label", because it was thought that most people would be coming to SKOS with little prior knowledge of RDF and without a computer science background, and would recognise and understand "lexical" (of or relating to words) better than "literal" (a computer science concept).

However, it has been suggested that the range of these properties might be allowed to include not only plain literals, but also other types of resource.

Resolution

There are two options here:

1. State that the range of these properties *is* the class of RDF plain literals (i.e. the range is closed).
2. State that the range of these properties *contains* the class of RDF plain literals, but may also contain other classes of resource (i.e. the range is left open).

I prefer the first option. My main reason for this preference is that, as will become obvious from the discussion below, the semantics already get quite complicated with the range closed as the class of RDF plain literals. Allowing for an open-ended range will introduce additional complexity, which I'm not sure I know how to handle.

Sub-Topic B: Disjoint Properties?

Originally, these three properties were intended to be pairwise disjoint, although this was never stated explicitly. That is, a preferred label cannot also be an alternative label or a hidden label; and an alternative label cannot also be a hidden label.

I.e. there is something intuitively wrong with each of the following graphs:

```
ex:foo skos:prefLabel "foo"@en; skos:altLabel "foo"@en.
```

...

```
ex:foo skos:prefLabel "foo"@en; skos:hiddenLabel "foo"@en.
```

...

```
ex:foo skos:altLabel "foo"@en; skos:hiddenLabel "foo"@en.
```

Resolution

There is really only one option here:

1. State that these three properties are pairwise disjoint.

Sub-Topic C: Cardinality?

Intuitively, it's obvious that a resource can have no more than one preferred lexical label per language. In other words, two different labels in the same language cannot both be "preferred". This is already stated informally in [SKOS-GUIDE].

I.e. there is something intuitively wrong with the following graph:

```
ex:shrubs skos:prefLabel "shrubs"@en; skos:prefLabel "bushes"@en.
```

The matter of cardinality is complicated by languages spoken differently in different regions (e.g. the English Language,

British English, US English), by the usage of different scripts in written languages (E.g. the Japanese Language, Japanese Hiragana, Japanase Katakana), and by other variants of a natural language.

Because there is a natural relationship between a language spoken in a specific region, such as British English, and its "parent" language, it *could* be argued that for example, the graph below...

```
ex:shrubs skos:prefLabel "shrubs"@en-GB.
```

... should entail ...

```
ex:shrubs skos:prefLabel "shrubs"@en.
```

If this entailment were allowed, this would lead to subtle issues with the cardinality skos:prefLabel, because for example the following graph would lead to a violation of the intuitive cardinality constraint given above...

```
ex:shrubs skos:prefLabel "shrubs"@en-GB; skos:prefLabel "bushes"@en.
```

However, if we treat the English Language, British English and US English as 3 distinct "languages", then there is no problem. Similarly, if we treat the Japanese Language, Japanese Hiragana and Japanese Katakana as 3 distinct "languages", then there is no problem.

This is equivalent to assuming that each distinct tag allowed by [RFC-4646] denotes a distinct "language".

Resolution

I think there is only one option here:

1. Assume that each distinct tag allowed by [RFC-4646] denotes a distinct "language", and use this special meaning for the word "language" when stating the cardinality constraint for skos:prefLabel (i.e. that there cannot be more than one preferred lexical label per "language").

Sub-Topic D: Super-Property?

In the most recent specifications, these three properties are stated as sub-properties of rdfs:label.

Is this a useful statement? Should it be retained?

Resolution

This still seems valid, and I cannot see any reason to drop it, so I suggest we:

1. Retain the statement that SKOS lexical labelling properties are sub-properties of rdfs:label.

Sub-Topic E: Formally Stating the Range Semantics?

If we adopt my suggested resolution to the range semantics (that the range of these three properties be closed as the class of RDF plain literals), how should this then be formally stated?

We could use RDF triples (e.g. skos:prefLabel rdfs:range ex:PlainLiteral.) but unfortunately there is no official URI

denoting the class of RDF plain literals.

Note that the class of RDF plain literals is a sub-class of rdfs:Literal, which also contains XML literals and typed literals. Therefore, the following graph...

```
skos:prefLabel rdfs:range rdfs:Literal.
skos:altLabel rdfs:range rdfs:Literal.
skos:hiddenLabel rdfs:range rdfs:Literal.
```

...whilst not incorrect, does **not** fully express our intended semantics.

Resolution

There are two options:

1. Coin a URI for the class of RDF plain literals, and use it to formally declare the range of the SKOS lexical labelling properties in axiomatic RDF triples.
2. State in normative prose that the range of the SKOS lexical labelling properties is actually the class of RDF plain literals; retain the current declaration in RDF triples that the range of these properties is rdfs:Literal.

I don't think there would be any value in option 1, so I suggest option 2.

Sub-Topic F: Literal Object Syntax Constraint?

There is a subtle but important complication here, due to the semantics of rdfs:range in both the RDF and OWL Full semantics.

Even if the range of these three properties is the class of RDF plain literals, this would **not** make the following graph inconsistent:

```
ex:foo skos:prefLabel ex:bar.
```

The only thing to be gained by the range semantics is the inference that the URI ex:bar denotes an RDF plain literal, which is probably not very useful.

If, however, we believe there **is** something wrong with the graph above, because we believe only RDF plain literals should ever appear in the object position of such a triple in an RDF graph, then we would need to do something completely different.

We would need to state constraints on the RDF abstract syntax. However, as far as I know, there is no standard way to formally express constraints over the RDF abstract syntax.

So there are two problems here: (1) how do you state syntax constraints for RDF? and (2) what should those constraints be for SKOS lexical labelling properties (if any)?

Before diving into the detail, I should say that the possible benefit of having syntax constraints is that it allows applications to do some simple "validation" at the graph syntax level. This makes for a stricter specification, which promotes interoperability (at the cost of flexibility) and makes implementation simpler (because there are less options to deal with). More on this below.

The only precedent I know of for declaring constraints over the RDF abstract syntax is the set of constraints imposed by the OWL DL language. Those constraints are, however, implicit in the mapping between the OWL abstract syntax and RDF triples. I would **not** recommend this approach for SKOS (i.e. to define a separate abstract syntax for SKOS) because it would seriously complicate the specifications.

One way to express an RDF syntax constraint is to use normative prose, with the keywords MUST, SHOULD, MAY etc.

For example, we could state that:

> Where skos:prefLabel, skos:altLabel or skos:hiddenLabel appears in the predicate position of a triple, the object MUST be an RDF plain literal.

A syntax constraint like this gives a stricter specification. This might be good, because it would lead to tighter interoperability and simpler implementations. This might be bad, because it gives less room for manouevre where people might want to experiment, for example by using URIs to denote plain literals.

Personally, I favour a stricter specification. If people want to experiment, they can always invent new properties.

Another way to express an RDF syntax constraint is to use a SPARQL graph pattern, where the binding of any variables in the pattern indicates a violation of the constraint in the matching graph.

For example:

```
{
  ?x skos:prefLabel ?y.
  FILTER (!isLiteral(?y))
}
```

(...although note that the SPARQL operator isLiteral() returns true for any RDF literal, included typed literals, so the syntax constraint expressed by this pattern is **not** equivalent to the normative prose stated above.)

There are further complications.

Suppose we adopt the syntax constraint stated in normative prose above. How should applications implement this constraint? Should an application consuming SKOS data generate a fatal error if it encounters a graph which violates the constraint? Or should it quietly recover according to some standard error-handling strategy?

Also, given that RDF data can be merged from multiple graphs, each individual graph might be valid, but the merge might be invalid. In practice, this means that 2 different applications might each **produce** perfect SKOS data, but an application **consuming** the merged data might find problems.

Initially, I thought it would be better to define ways in which applications could quietly handle any syntax problems, rather than forcing them to generate fatal errors. This was the motivation for the proposal I made at [LABEL-SEMANTICS], in which the syntax constraint stated above is instead given by the following statement:

> An application MAY ignore any triple in an RDF graph where the predicate is either skos:prefLabel, skos:altLabel or skos:hiddenLabel and the object is NOT a plain literal.

However, this proposal is complicated, and has a number of potential problems as highlighted in [VATANT].

Resolution

I don't have any clear options for this at present.

I only know that, if we do want syntax constraints, then we should also state how applications MUST/SHOULD/MAY handle violations of those constraints.

We should also separate syntax constraints from application behaviour. I.e. we should state syntax constraints without talking about application behaviour, **then** state how applications MUST/SHOULD/MAY implement those constraints. I think I made the mistake of confusing these two considerations in the proposal I made at [LABEL-SEMANTICS].

Sub-Topic G: Formally Stating Disjointness?

Unfortunately, neither RDF nor OWL has a concept of disjoint properties.

We could use normative prose to state a semantic condition on the interpretation of the three properties, for example:

> The property extensions of skos:prefLabel, skos:altLabel and skos:hiddenLabel are pairwise disjoint.

We could also go a bit further, and follow the conventions and definitions set out in [RDF-SEMANTICS] for stating semantic conditions on the interpretation of some RDF vocabulary, for example:

> For any resource x, the sets { y | <x,y> is in IEXT(I(skos:prefLabel)) }, { y | <x,y> is in IEXT(I(skos:altLabel)) } and { y | <x,y> is in IEXT(I(skos:hiddenLabel)) } are pairwise disjoint.

Note that expressing the disjointness of the SKOS lexical labelling properties in semantic conditions requires a notion of SKOS-interpretation and SKOS-inconsistency.

We could also handle disjointness entirely at the syntax level (see next sub-topic).

Resolution

I'm not sure what to recommend here. We could state a semantic condition for completeness, but I would not expect anyone to implement it in a reasoner -- it's possible, and easier, to test for "inconsistencies" by looking for graph patterns (i.e. by checking the syntax).

Sub-Topic H: Disjointness Syntax Constraint?

As mentioned above, it would be possible to state syntax constraints which effectively capture our intension for the three properties to be disjoint.

For example, as normative prose:

> A graph MUST NOT contain a triple with skos:prefLabel as predicate and a triple with skos:altLabel as predicate, where both triples share the same subject and object.

The same constraint, expressed as a SPARQL graph pattern matching an "invalid" graph:

```
{
  ?x skos:prefLabel ?y.
  ?x skos:altLabel ?y.
}
```

The advantage of syntax constraints is that they are (arguably) easier to implement than a reasoner, e.g. by using SPARQL.

Note that we could have both semantic conditions and syntax constraints -- these options are not mutually exclusive.

However, we still have the problem of stating how applications should handle a violation of these constraints. Should they generate a fatal error? Should they quietly recover, and if so, how?

Resolution

Again, I'm not sure what to recommend here. Syntax constraints are undoubtedly useful, but should they be part of the normative specification, or should they be informative? How should applications handle violations?

Sub-Topic I: Formally Stating Cardinality?

Our intuitive cardinality constraint -- that there cannot be more than one preferred lexical label per "language" for any given resource -- cannot be expressed as axiomatic triples using either RDF or OWL vocabularies. This is because the cardinality has to take into account the value of the language tag on an RDF plain literal.

This can be stated as a semantic condition, following the conventions and definitions set out in [RDF-SEMANTICS], for example:

> For any resource x, all members of the set { y | <x,y> is in IEXT(I(skos:prefLabel)) } are RDF plain literals and no two members of this set share the same language tag.

However, again, this semantics can also be effectively captured in a syntax constraint (see sub-topic below).

Resolution

Again, I'm not sure what to recommend here. As with disjointness, we could state a semantic condition for completeness, but I would not expect anyone to implement it in a reasoner -- it's possible, and easier, to test for "inconsistencies" by looking for graph patterns (i.e. by checking the syntax).

Sub-Topic J: Cardinality Syntax Constraint?

As mentioned above, it would be possible to state a syntax constraint which effectively captures our intension that there cannot be more than one preferred lexical label per "language" for any given resource.

For example, as normative prose:

> A graph MUST NOT contain two or more triples with the same subject, where the predicate is skos:prefLabel, and where the objects are plain literals with the same language tag.

For example, as a SPARQL graph pattern matching an "invalid" graph:

```
{
  ?x skos:prefLabel ?y, ?z.
  FILTER ( (str(?y) != str(?z)) && (lang(?y) = lang(?z)) )
}
```

Resolution

Again, I'm not sure what to recommend. Syntax constraints are undoubtedly useful, but should they be part of the normative specification, or should they be informative? How should applications handle violations?

Sub-Topic K: OWL Property Type?

OWL has the notions of Datatype Properties and Object Properties. See section 4 of [OWL-REFERENCE] for some important details on these two categories of property.

OWL also has a notion of Annotation Properties, although note that these are only needed for reasoning in OWL DL. If we are building a semantics for SKOS on OWL Full, we can ignore Annotation Properties.

Note also that, in OWL Full, object properties and datatype properties are not disjoint. Because data values can be treated as individuals, datatype properties are effectively subclasses of object properties. In OWL Full owl:ObjectProperty is equivalent to rdf:Property.

Should the SKOS lexical labelling properties be instances of owl:ObjectProperty or owl:DatatypeProperty?

Resolution

The resolution depends on our decision for sub-topic A (Range Semantics). If we choose to set the range as the class of RDF plain literals, then these properties should be instances of owl:DatatypeProperty. If we choose an open-ended range, then we can only say they are instances of owl:ObjectProperty.

Summary

Wow! As I said at the beginning, who would have thought three little properties could cause so many problems!

Hopefully, the first step will be to agree on the actual semantics of these three properties, stated informally. I.e. to agree a resolution on sub-topics A, B, C and D.

Then we can discuss how best to state the semantics formally (if at all), and whether syntax constraint might be useful, as either normative or informative parts of the specification.

References

[SKOS-GUIDE]
http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102

[SKOS-SPEC]
http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102

[ISSUE-31]
http://www.w3.org/2006/07/SWD/track/issues/31

[RDF-CONCEPTS]
http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[RDF-SEMANTICS]
http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

[RFC-4646]
http://www.ietf.org/rfc/rfc4646.txt

[LABEL-SEMANTICS]
http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0170.html

[VATANT]
http://lists.w3.org/Archives/Public/public-esw-thes/2007Jun/0039.html

[OWL-REFERENCE]
http://www.w3.org/TR/2004/REC-owl-ref-20040210/

```
--- Change Log ---
$Log: labelling-properties.html,v $
```

```
Revision 1.2  2007/10/02 12:09:21  ajm65
Added reference links. Minor edit to discussion of languages and language tags.

Revision 1.1  2007/10/02 11:43:39  ajm65
First completed draft. TODO reference links.
```

ISSUE-44

BroaderNarrowerSemantics

State:
OPEN

Product:
SKOS

Raised by:
Sean Bechhofer

Opened on:
2007-06-15

Description:

```
What are the semantics of skos:broader and skos:narrower? There are several open
questions:

 * Are they transitive? Intransitive? Or can the application choose?
 * Are they reflexive? Irreflexive? Or can the application choose?
 * Should the transitive closure be irreflexive (i.e. a cycle is a
contradiction)? Or can the application choose?
```

Related emails:

1. ISSUE-44: BroaderNarrowerSemantics (from dean+cgi@w3.org on 2007-06-15)
2. Re: ISSUE-44: BroaderNarrowerSemantics (from dlrubin@stanford.edu on 2007-06-15)
3. Re: ISSUE-44: BroaderNarrowerSemantics (from diego.berrueta@fundacionctic.org on 2007-06-21)
4. [SKOS] central issues (from A.J.Miles@rl.ac.uk on 2007-06-26)
5. Re: [SKOS] central issues (from bernard.vatant@mondeca.com on 2007-06-26)
6. Re: [SKOS] central issues (from aisaac@few.vu.nl on 2007-06-27)
7. Re: [SKOS] central issues (from bernard.vatant@mondeca.com on 2007-06-27)
8. Re: [SKOS] central issues (from schreiber@cs.vu.nl on 2007-07-02)
9. Re: [SKOS] central issues (from jphipps@madcreek.com on 2007-07-02)
10. Re: [SKOS] central issues (from aisaac@few.vu.nl on 2007-07-03)
11. Re: [SKOS] central issues (from jphipps@madcreek.com on 2007-07-03)
12. Re: [SKOS] central issues (from schreiber@cs.vu.nl on 2007-07-04)
13. [ALL] SWD face-to-face - draft agenda (from baker@sub.uni-goettingen.de on 2007-08-

22)
14. RE: [ALL] SWD face-to-face - draft agenda (from A.J.Miles@rl.ac.uk on 2007-08-24)
15. Re: [ALL] SWD face-to-face - draft agenda (from aisaac@few.vu.nl on 2007-08-27)

Related notes:

2007-08-21: Assigned to Sean during 21 August WG telecon

This page provides some discussion guide on giving OWL semantics to skos:concept, and bridging SKOS modelling with OWL one, for Amsterdam F2F meeting. It relates to ISSUE-54: ConceptSemantics.

Contents

1. The Issue

Issue 54 ConceptSemantics [1] reads:

> *"What are the semantics of skos:Concept? skos:Concept is currently declared as { skos:Concept rdf:type rdfs:Class . } Is this ok? Is this enough? What about the OWL universe?"*

This raises a first problem:

> Problem 1: how to give a formal definition of the skos:Concept primitive using RDFS/OWL?

Also, because of discussions on the SWD and ESW-Thes mailing list, a second problem emerged, linked to the wish to use SKOS features at the same time as standard RDFS or OWL ontology features [2]:

> Problem 2: what are the possible articulations between objects that are defined in SKOS world (“concept schemes” gathering “concepts”) and objects that are defined in OWL world (“ontologies” gathering “classes”, “properties” and “individuals”)

2. Giving semantics to skos:Concept

Preliminaries

Existing (informal) semantics of SKOS concept:

- SKOS specification [3] gives the definition *"An abstract idea or notion; a unit of thought."* following [4], for which a concept is a *"unit of thought. The semantic content*

of a concept can be re-expressed by a combination of other and different concepts, which may vary from one language or culture to another. Concepts exist in the mind as abstract entities which are independent of the terms used to label them".

- SKOS guide gives a clarification [5]: *"SKOS Core allows you to model a set of concepts (essentially a set of meanings) as an RDF graph. Other RDF applications, such as FOAF, allow you to model things like people, organisations, places etc. as an RDF graph. Technically, SKOS Core introduces a layer of indirection into the modelling…So, for a resource of type skos:Concept, any properties of that resource (such as creator, date of modification, source etc.) should be interpreted as properties of a concept, and not as properties of some 'real world thing' that that resource may be a conceptualisation of."*
  There is an example, where an instance of foaf:Person (which is an instance of owl:Class, as defined in [6]) is the dc:creator of an instance of skos:Concept.

Semantics of RDFS/OWL Classes:

- In RDFS Semantics [7] a class is a *"resource which represents a set of things in the universe which all have that class as the value of their rdf:type property"*
- In OWL Overview [8] *"A class defines a group of individuals that belong together because they share some properties."* Classes have OWL individuals as members (in OWL-Full, these members can be classes as well).

Towards OWL semantics for skos:Concept

Can we say skos:Concept rdf:type rdfs:Class? According to RDFS semantics, yes: it actually naturally comes from any statement that links a specific SKOS concept to skos:Concept via rdf:type

> **Proposal 1**: `skos:Concept rdf:type rdfs:Class.` shall be added to the SKOS semantics

Can we say skos:Concept rdf:type owl:Class? According to OWL definition for a class, yes. The class skos:Concept is the group of individuals that belong together because they are units of thought forming the elements knowledge organization schemes as modelled in SKOS.

> **Proposal 2**: `skos:Concept rdf:type owl:Class` shall be added to the SKOS semantics. As RDFS class membership is trivial and every OWL class is an RDFS class (OWL Reference, [9]) this statement can replace the one of proposal 1

Can we compare the set of OWL classes with the set of SKOS concepts?

The SKOS guide example previously mentioned gives a first answer on how to link some individuals to OWL class. It avoids committing however to the special cases where one wants to "bridge" (eventually by identity) standard OWL modelling of ontologies and knowledge bases (especially OWL classes) with SKOS modelling of general knowledge organization systems.

Actually it brings additional ambiguity: according to SKOS guide [5], a resource of type skos:Concept is a "conceptualization" of a real thing, while, according to common ontology definitions, an ontology is a " specification of conceptualization". The notion of "concept" may sound quite close to the one of "class"…

OWL classes are strongly linked to an “extension” : the set of their instances. Formally, an instance of skos:Concept (ex:Car) is not defined to have such instances in SKOS: a SKOS concept is supposed to be an individual element. Yet it can be linked to two types of “extensions”:

- the documents that have this concept as subjects (as defined e.g. by the “literary warrant” in library science) by means of skos:subject in SKOS
- the “real things” that can constitute the reference of the concept, e.g. all cars constitute the reference of ex:Car.

But none of these “extensions” can define a SKOS concept the way an OWL class is defined: books are not instances of a subject in the sense of rdf:type, and a “reference” is not easy to determine for all SKOS concepts, e.g. abstract things like “freedom” - as opposed to tangible ones as “cars”.

So generally we cannot say that all SKOS concepts are OWL classes. In terms of formal semantics we would not have `skos:Concept rdfs:subClassOf owl:Class`. Now, do we have `skos:Concept owl:disjointWith owl:Class`, i.e. can we allow some OWL classes to be considered as SKOS concepts and vice versa, or not?

3. Bridging SKOS modelling with OWL modelling

Motivation

There is no formal requirement for it in SKOS use cases [10], but some mails on the SWD and SKOS list hint at potential requirements:

1. A general strategic requirement, as expressed by Daniel [11]. *"I thought the goal of SKOS is to "provide a standard way to represent knowledge organization systems." Ontologies certainly fall under that umbrella. Some of the SKOS model is certainly very relevant to people building ontologies."*
2. Daniel and the RadLex use case *"In the RadLex use case, for example, "terms" are entities in reality, such as blood vessels. These are linked together via relations such as "part-of" and "continuous-with"."* [12] *"To me, "term" is the name for something. "Concept" is something that is someone's head. "Entity" is something that exists in reality. There are certainly communities who need to describe things, names of things, and both (in the case of RadLex). Ideally, SKOS should be able to be useful to these communities."* [11]
3. Kjetil and myOpera/POWDER cases. *"My idea is to use SKOS for tags, but add a mapping to something else, like a foaf:Person."* [13] The discussion on POWDER use case is available at [14,15]; a similar case was raised by Richard Cyganiak on the SKOS list [16]
4. Informal discussions on interest for SKOS. *"I presented SKOS to a biomedical workshop this week, and it seems that some people wished to use SKOS, but on 'real' ontologies they were developing."* [Antoine, 2] Also: *"While OWL statements are always domain 'object' specific and not always about disambiguating natural language 'terms' .. when the object of a OWL declaration is ALSO a term in a Common Vocabulary .. SKOS helps explain how a set of specific national language terms /*

symbols are used in that community of practice." [Carl Mattocks, 17]

Practical needs

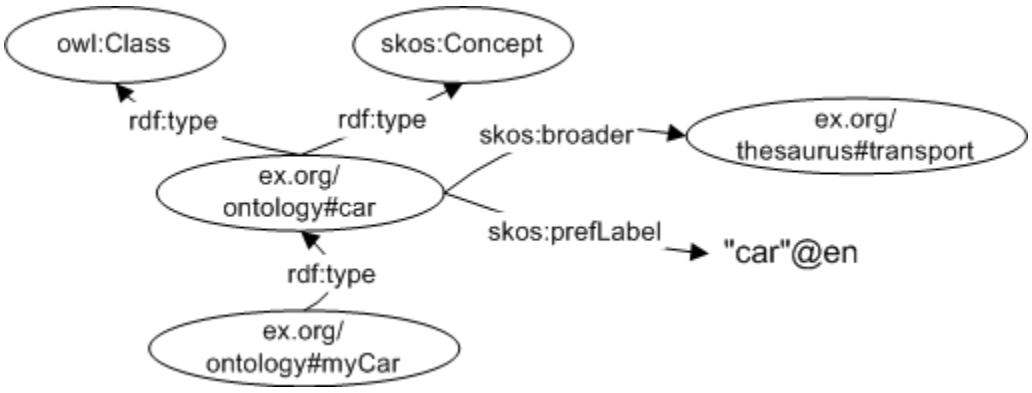
1. use of SKOS to define ontology classes (or instances of classes) by means of SKOS features.
   - to benefit from SKOS rich labelling properties, as mentioned in "OWL-DL Compatibility" wiki page [18] about the possibility to make skos:prefLabel a specialization of rdfs:label:
     `ex:Foo rdf:type owl:Class; skos:prefLabel "Foo".`
   - to insert existing owl:Class (e.g. exOnt:Planet) and instances (exOnt:Venus) into a thesaurus-like thematic structure
     (`exOntVenus skos:broader exOnt:Planet, exOnt:Planet skos:broa`
     which could be very useful to develop browsers more useful for humans [2]
2. link instances of skos:Concept with instances of owl:Class that seems to be "semantically related" (e.g. a skos:Concept henryVIIIConcept with a foaf:Person henryVIIIPerson)

In some of these cases, the direct application of SKOS primitives to instances of owl:Class could cause these instances of owl:Class to be also classified as instances of skos:Concept according to the semantics of SKOS [19].

Possible solutions

Solution 1: instances of owl:Class (or rdf:Property or owl:Individual) are allowed to be instances of skos:Concept. There is no disjointness between owl:Class and skos:Concept, and skos:prefLabel and all others SKOS properties can be attached to classes defined in OWL ontologies.

"if there is a modeling distinction, there is no [formal] exclusion. You can have skos:Concepts (my:Car rdf:type skos:Concept) that are also RDFS/OWL classes (ex:Car rdf:type rdfs:Class) so that you can create 'objects' which are classified under it (ex:danielsCar rdf:type ex:car)." [Antoine,20]



Pros:

- It is compatible with current semantics of SKOS concept
- Simplicity: the solution is the simplest to use for these who just want to use SKOS

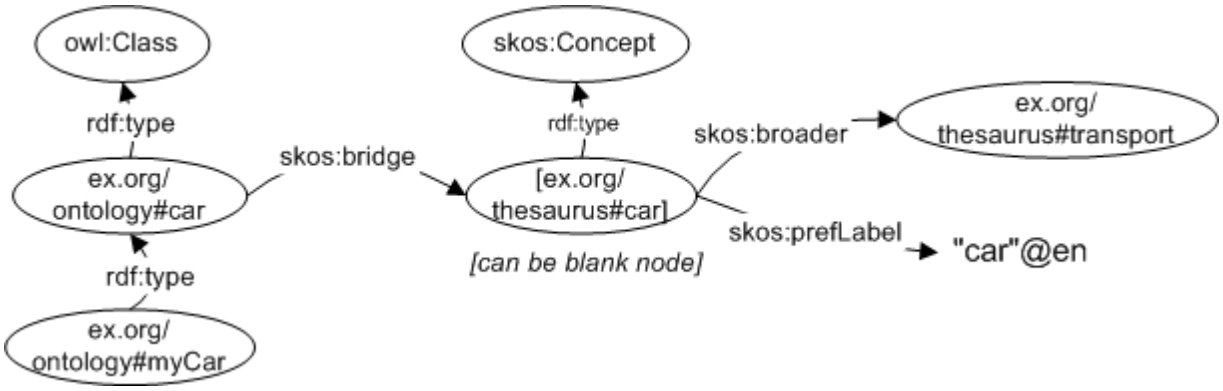
properties as extra labelling properties for instance.
- Conciseness: there is no redundancy between what is defined for the "standard" ontological aspects and what is defined for the SKOS-oriented ones. No new object (RDF resource) is created to represent the class and the related SKOS information
- Ontological commitment as standard: SKOS users can rely on work done by the SWD WG to frame such a mixing of OWL and SKOS

Cons:

- OWL-Full (not so important now that the issue is postponed, but still interesting to mention)
- Additional commitment on the agenda SWD WG, indeed messing OWL and SKOS missions: OWL for logic-oriented modelling, SKOS for more informal (incl. linguistic) aspects of the semantics [21,22].
- Re-usability and authority: there could be cases where the developers of the SKOS aspects and the OWL ones are not the same (typically when you want to link your work on a SKOS representation to an already existing OWL one). Mixing the different aspects eventually leads to disappearance of benefits of re-using the same objects.
- This solution is especially hard to buy for instances of owl:Individuals that represent "tangible" things. *"how can it be that a concept is also something tangible such as a car?"*
  [Daniel, 23] Cf. also "dc:creator-test" in SKOS guide: the dc:creator of a car is likely to be different from the dc:creator of a "corresponding" car SKOS concept.

Solution 2: instances of owl:Class/rdf:Property/owl:Individual and instances of skos:Concept are "bridged" via a dedicated property.

Proposed by Dan Brickley, [24]



Pros:

- Not mixing OWL and SKOS missions, only bridging. SWD WG make a smaller commitment, in the boundaries of the WG's scope.
- Can be OWL-DL (not so important now the issue is postponed, but still interesting to mention)
- Ease of modelling: Users can deal with separate object with simple interpretation, no unexpected logical consequences

Cons:

- WG is to find names and (still) proper semantic foundation (is "link between a SKOS

concept and its OWL reference" enough as a definition for the property??)
- Less conciseness: duplication of "corresponding concept" RDF resources, users/developers have to deal with longer path in RDF graph queries

Problem: what are the semantics and proper name for such a link? It corresponds to something like "reference": a link from a concept to a "real" thing (or set of "real" things) it denotes.

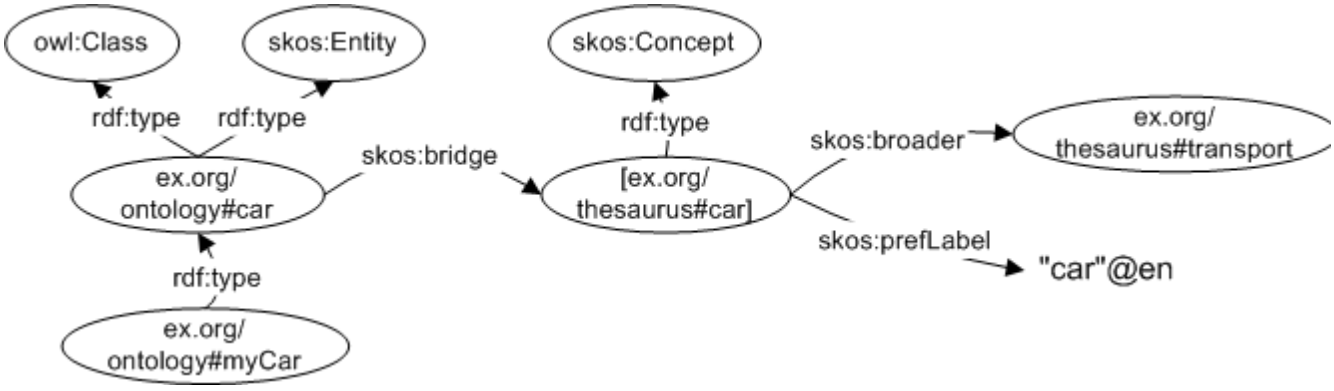
"one of the property names Alistair and I were throwing around for this was "skos:as" (inverse: skos:it). This would link between inviduals ("dan") and classes ("Person") and corresponding topics in a SKOS scheme:" [Dan, 24]

```
<owl:Class rdf:ID="Person">
  <skos:as skos:prefLabel="Person" skos:altLabel="Guy"/>
</owl:Class>
```

(SKOS list has previous discussions on the topic of skos:it/skos:as [25, 26]; skos:denotes, skos:represents and skos:intends were refused)

There are actually objections to a link with such "denotation" interpretation: *"So I think that the word "denotes" to connect a thesaurus-theoretic and a ontology-theoretic point of view is dangerous - as in logic and mathematics this usually signifies semantics - we are not (should not) be saying that the class provides a semantics for the concept."* [Brian Matthews, one mail in [25,26] threads]

Variant 1 for Solution 2: Same pattern, but introducing skos:Entity as a sibling of skos:Concept to type the "real things" that are denoted by SKOS concepts. SKOS properties (especially the labelling ones) can be applied to both concepts and entities. Proposed by Daniel [27,28]



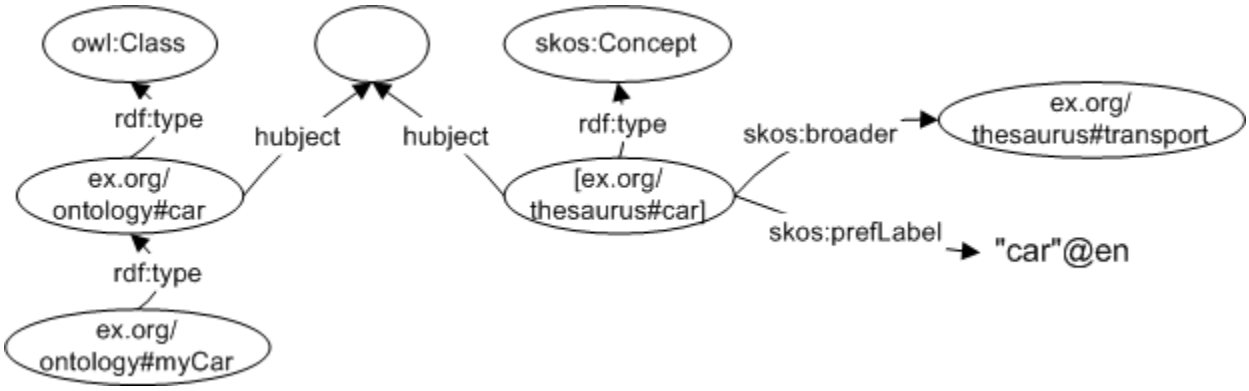
Pros:

- No properties linking SKOS objects to other ojects "outside of SKOS"
- User benefits from semantic clarification work done by WG on the link and this new skos:Entity class (e.g. it is disjoint from skos:Concept, etc.)

Cons:

- We have to do this additional semantic clarification work (only name and vague semantics are present in mails)
- We re-introduce messing SKOS and OWL missions: some SKOS entities will be OWL classes

Variant 2 for Solution 2: Different pattern: the skos:Concept and the owl:Class both refer to a same blank node that is interpreted as an abstract subject (treated as a blank node) they are both representation of. Introduced in many posts on the SKOS lists by Bernard Vatant, also at [29]



Pros:

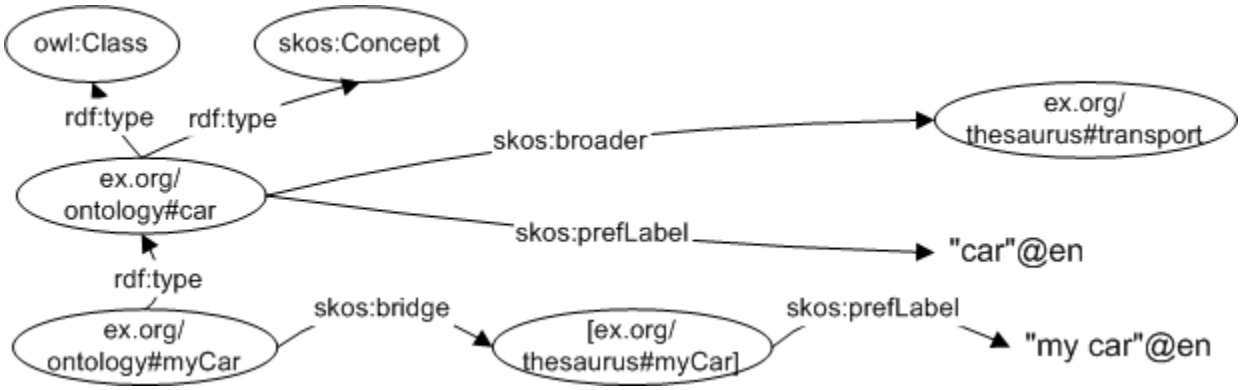
- No strong meta-level commitment: there is just "something out there" that is common the the skos:Concept and the owl:Class being defined
- Compatible with knowledge primitives other than skos:Concept and owl:Class

Cons:

- One additional step in the RDF graphs involved
- Complexity of modelling: use of blank node and resource without explicit semantic type is not really trivial. And hubjects are just a part of Bernard's proposal, which goes beyond

Solution 3, mixing 1 and 2
depending on the "nature" of the object considered in the OWL world. Only the OWL-defined resources that are of type owl:Class can be SKOS-defined according to solution 1, the others must be SKOS-defined according to solution 2.

It is easier to keep the line blurred when we have things defined as "concepts" and "sets". Cf. "dc:creator test": the dc:creator of an owl:Class can also be the dc:creator of a "corresponding" skos:Concept.



Pros:

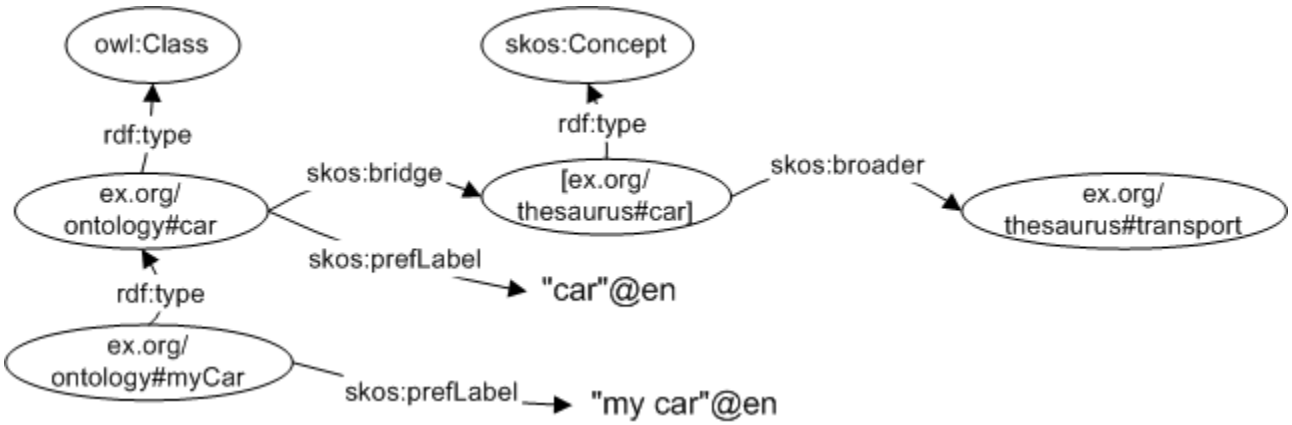
- Keeps simplicity for one very important part of the anticipated use cases (SKOS

definitions of OWL classes)

Cons:

- Forces user to make different modelling choices, depending on the resource he wants to define with both OWL and SKOS features

Solution 4, mixing 1 and 2
depending on the SKOS constructs that are used for the SKOS-definition of the OWL-object. Only the SKOS properties that do not have skos:Concept as rdfs:domain can be directly applied to OWL-defined objects. For the others, the pattern of solution 2 must be chosen.



Pros:

- Keeps simplicity for one very important part of the anticipated use cases (labelling properties)
- We can keep the original SKOS semantics for properties and introduce a disjointness axiom between owl:Class and skos:Concept

Cons:

- Forces user to make different modelling choices, depending on the modelling aspect he wants to use
- Could force users that wants to use a quite complete set of SKOS constructs for their OWL-defined objects to turn to different choices for a same application, unless we make both solution 1 and 2 usable for the case of SKOS properties that do not have skos:Concept as rdfs:domain

Conclusion

- Which pattern is recommended to link SKOS concept and OWL objects?
- Do we add the disjointness axiom skos:Concept owl:disjointWith owl:Class?

References

[1] http://www.w3.org/2006/07/SWD/track/issues/54

[2] http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0077.html

[3] http://www.w3.org/TR/swbp-skos-core-spec/#Concept

[4] http://www.willpowerinfo.co.uk/glossary.htm

[5] http://www.w3.org/TR/swbp-skos-core-guide/#secmodellingrdf

[6] http://xmlns.com/foaf/spec/index.rdf

[7] http://www.w3.org/TR/rdf-mt

[8] http://www.w3.org/TR/owl-features

[9] http://www.w3.org/TR/owl-ref

[10] http://www.w3.org/TR/skos-ucr

[11] http://lists.w3.org/Archives/Public/public-swd-wg/2007May/0075.html

[12] http://lists.w3.org/Archives/Public/public-swd-wg/2007May/0073.html

[13] http://lists.w3.org/Archives/Public/public-swd-wg/2007Mar/0049.html

[14] http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0143.html

[15] http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0125.html

[16] http://lists.w3.org/Archives/Public/public-esw-thes/2006Dec/0013.html

[17] http://lists.w3.org/Archives/Public/public-esw-thes/2006Nov/0015.html

[18] http://lists.w3.org/Archives/Public/public-swd-wg/2007Feb/0145.html

[19] http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0021.html

[20] http://lists.w3.org/Archives/Public/public-swd-wg/2007May/0076.html

[21] http://lists.w3.org/Archives/Public/public-swd-wg/2007Feb/0159.html

[22] http://lists.w3.org/Archives/Public/public-swd-wg/2007Feb/0161.html

[23] http://lists.w3.org/Archives/Public/public-swd-wg/2007May/0083.html

[24] http://lists.w3.org/Archives/Public/public-swd-wg/2007Feb/0160.html

[25] http://lists.w3.org/Archives/Public/public-esw-thes/2004Sep/0041.html

[26] http://lists.w3.org/Archives/Public/public-esw-thes/2005Jun/0002.html

[27] http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0017.html

[28] http://lists.w3.org/Archives/Public/public-swd-wg/2007Jun/0030.html

[29] http://www.mondeca.com/lab/bernard/hubjects.pdf

SkosDesign/ConceptSemantics (last edited 2007-09-25 17:03:59 by AntoineIsaac)

SWDWG Amsterdam F2F October 2007

Topic: SKOS Label Relations
Discussion of Option 2 (Guus' "Simple Extension" Proposal)

This document gives some discussion of Guus' "Simple Extension" proposal for stating label relations in SKOS, as input to the "Label Relations" topic at the October 2007 SWDWG F2F meeting in Amsterdam.

Latest Version: http://purl.org/net/skos/2007/10/f2f/label-relations.html

$Revision: 1.3 $ on $Date: 2007/10/03 11:04:48 $

Contents

Introduction

The basic requirement is to be able to assert relationships between "lexical labels", to indicate e.g. a relationship between an acronym and it's expansion, or an abbreviation and its expansion, or a literal translation between labels in different languages.

Note that the notion of a "lexical label" was invented for SKOS, and has never been precisely defined. However, originally, there was intended to be a close correspondance between "lexical labels" and RDF plain literals. I.e. a "lexical label" has at least has two components, it's lexical form (a UNICODE string) and it's language (denoted by e.g. a valid [RFC-4646] tag). See also [LABEL-SEMANTICS].

RDF does not allow plain literals to occur in the subject position of a triple, therefore there is no direct way to make statements about a plain literal in RDF.

One work-around for this restriction is to use the n-ary relations design pattern. This is the basis for the [MINIMAL-LABEL-RELATION] proposal.

Another work-around is to define a new class of resources to represent "lexical labels". This is the basis for the [SIMPLE-EXTENSION] proposal.

This document discusses the [SIMPLE-EXTENSION] proposal, and the practical implications of introducing a new class of resource in SKOS to denote a class of "lexical labels" or similar.

Alternative Semantics of skos:Label

The basic idea of [SIMPLE-EXTENSION] is that we introduce a new class ... skos:Label.

We can then state that some resource is an instance of skos:Label, for example:

```
ex:cow rdf:type skos:Label.
```

We then need some property to state a corresponding plain literal value of this instance. In [SIMPLE-EXTENSION], rdfs:label is used. However, this makes it harder to illustrate the different possible semantics for the skos:Label class. Therefore, for discussion here, I will invent skos:plainLiteralValue, and assume that the domain of skos:plainLiteralValue is the class skos:Label, and the range is the class of RDF plain literals. For example:

```
ex:cow rdf:type skos:Label; skos:plainLiteralValue "cow"@en.
```

We then need some property to state relationships between two instances of skos:Label. Following [SIMPLE-EXTENSION], let's use skos:labelRelation. For example:

```
ex:cow rdf:type skos:Label; skos:plainLiteralValue "cow"@en.
ex:vache rdf:type skos:Label; skos:plainLiteralValue "vache"@fr.
ex:cow skos:labelRelation ex:vache.
```

Of course, skos:labelRelation would not be used directly, but as an extension point for other properties representing more specific types of label relationship, such as acronym or literal translation relationships.

Now consider two basic questions.

Firstly, can an instance of skos:Label have more than one plain literal value?

For example, is the following graph allowed (i.e. consistent):

```
ex:foo rdf:type skos:Label;
  skos:plainLiteralValue "foetus"@en;
  skos:plainLiteralValue "fetus"@en.
```

...?

Secondly, can two different instances of skos:Label have the same plain literal value?

For example, is the following graph allowed (i.e. consistent):

```
ex1:cow rdf:type skos:Label; skos:plainLiteralValue "cow"@en.
ex2:cow rdf:type skos:Label; skos:plainLiteralValue "cow"@en.
ex1:cow owl:differentFrom ex2:cow.
```

...?

Given that each question can have a yes-or-no answer, this gives us four possible options.

Each of these four options gives a fundamentally different semantics, and each of these options leads to a very different "story" about what an instance of skos:Label actually "is".

[SIMPLE-EXTENSION] actually rules out two of these four options (it says "no" to the first question), but it does not specify which of the remaining two it chooses. That is why there are actually two variants of the [SIMPLE-EXTENSION] proposal, which I will attempt to illustrate below. For completeness, I will attempt to illustrate all four possible options.

A Note on Binary Relations, Functions, Inverse Functions and Bijections

What we are looking at here are the possible options for the relationship between the set of instances of skos:Label and the set of RDF plain literals.

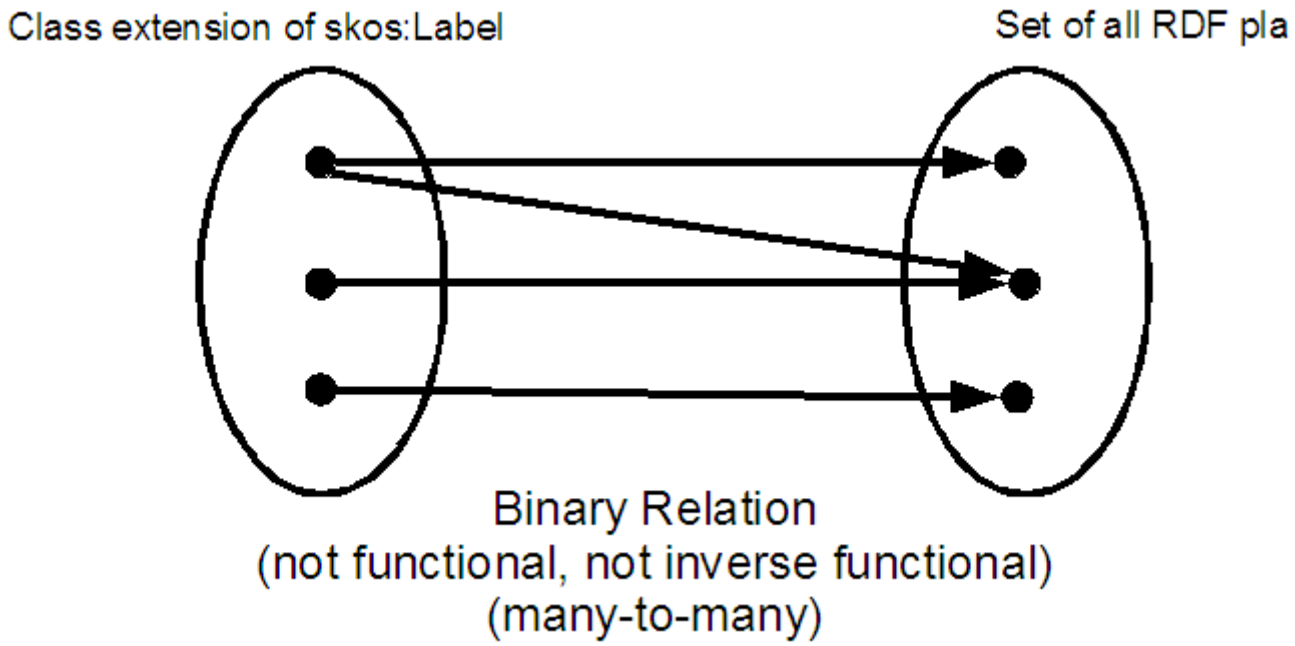
The essential difference between the four possible options for this relationship is captured in the difference between four mathematical concepts: a binary relation between two sets (many-to-many), a functional relation (many-to-one), an inverse functional relation (one-to-many), and a bijection (one-to-one).

N.B. the set of instances of skos:Label is also called the *Class Extension* of skos:Label, which is written in [RDF-SEMANTICS] as ICEXT(I(`skos:Label`)).

Binary Relation (Many-to-Many)

If we answer "yes" to both of the questions above, then there is a many-to-many relationship between the class extension of skos:Label and the set of RDF plain literals.

In the diagram below, the ellipse on the left depicts the class extension of skos:Label, and the ellipse on the right depicts the set of all RDF plain literals. The diagram as a whole depicts a binary relation between these two sets. The diagram shows a many-to-many relationship between the two sets.

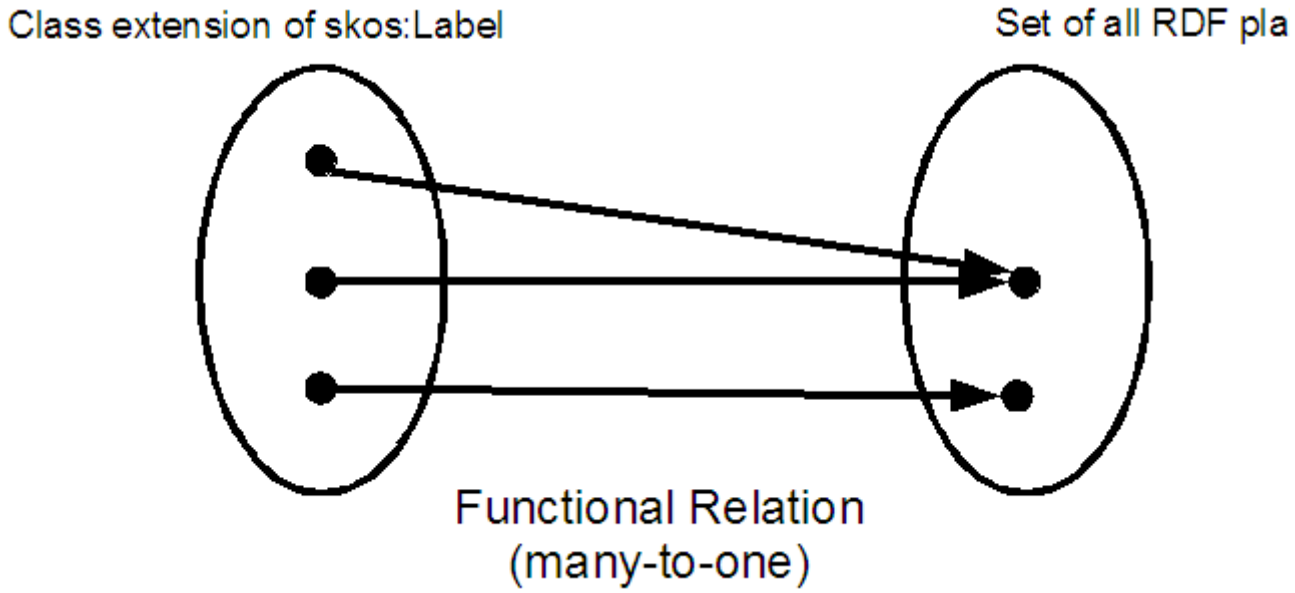


This diagram as a whole is also a depiction of the *Property Extension* of skos:plainLiteralValue.

Functional Relation (Many-to-One)

If we answer "no" to the first question, but "yes" to the second question, then there is a many-to-one relationship between the class extension of skos:Label and the set of RDF plain literals.

A many-to-one relationship between two sets is also called a functional relation (or simply function). The diagram below illustrates a functional relation.



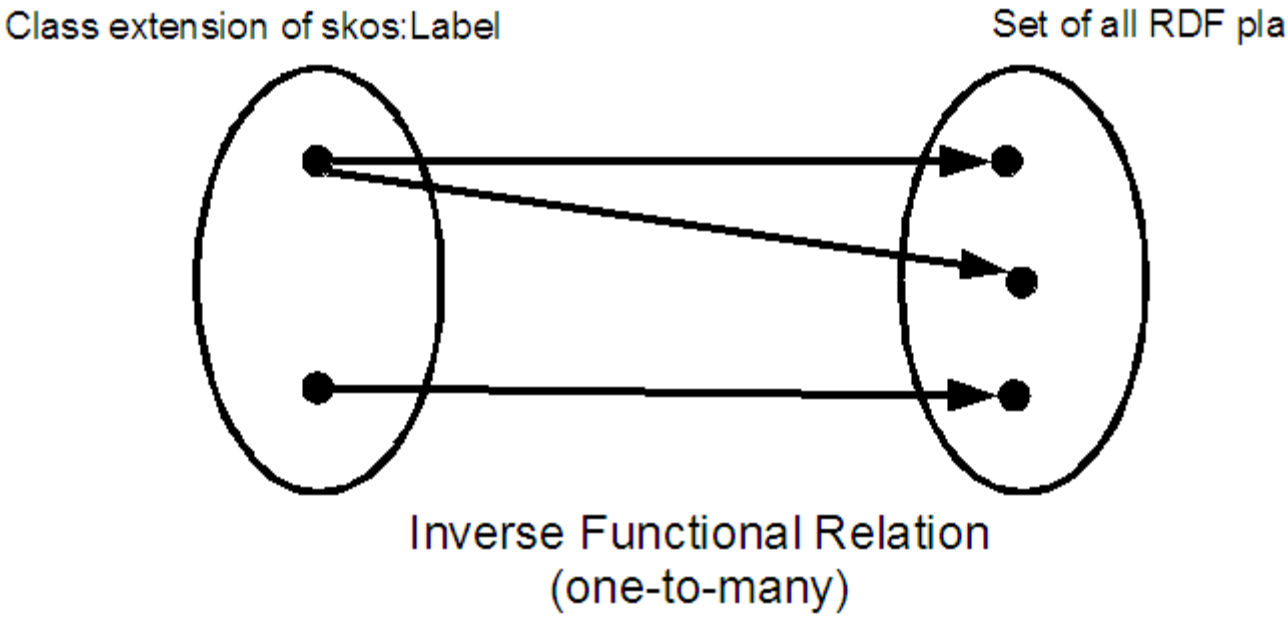
This semantics is captured in the following triples:

```
skos:plainLiteralValue rdf:type owl:FunctionalProperty.
```

Inverse Functional Relation (One-to-Many)

If we answer "yes" to the first question, but "no" to the second, then there is a one-to-many relationship between the class extension of skos:Label and the set of RDF plain literals.

A one-to-many relationship between the two sets is also called an inverse functional relation. The diagram below illustrates an inverse functional relation.



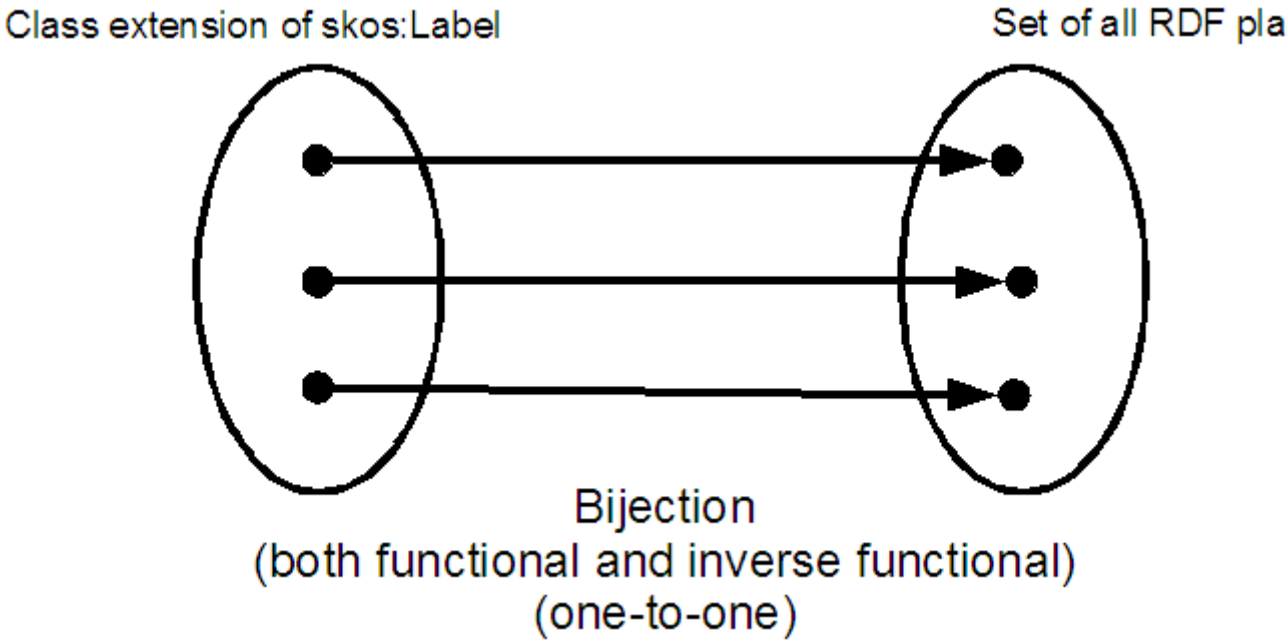
This semantics is captured in the following triples:

```
skos:plainLiteralValue rdf:type owl:InverseFunctionalProperty.
```

Bijection (One-to-One)

If we answer "no" to both questions, then there is a one-to-one relationship between the class extension of skos:Label and the set of RDF plain literals.

A one-to-one relationship between two sets is also called a bijection, which is illustrated below.



This semantics is captured in the following triples:

```
skos:plainLiteralValue rdf:type owl:FunctionalProperty, owl:InverseFunctionalProp
```

Consequences

As I mentioned above, [SIMPLE-EXTENSION] rules out two of these four options. It rules out a many-to-many relation, and it rules out a one-to-many relation.

However, for completeness, let us consider these two options briefly.

Many-to-Many / One-to-Many

If we allowed there to be one or more plain literal values for any instance of skos:Label, then an example like the one given above (repeated below) is allowed.

```
ex:foo rdf:type skos:Label;
  skos:plainLiteralValue "foetus"@en;
  skos:plainLiteralValue "fetus"@en.
```

The difficulty with these two options is then defining the circumstances under which it makes sense to allow more than one plain literal value for an instance of skos:Label. The same problem is encountered when we try to tell a "story" about what an instance of skos:Label actually "is". E.g. could we say that an instance of this class is a sequence of phonemes? We immediately get into the modeling of natural language, which is not something I am an expert on, and is not necessarily relevant to our use cases (which are mostly focused on information retrieval). If there is an existing model of language we could adopt, then these options *might* become feasible, but even then I'm not sure it's worth the effort.

One-to-One

If there is one and only one instance of skos:Label for every RDF plain literal, then neither of the examples given above are allowed.

Under these circumstances, the class skos:Label is effectively equivalent to the class of RDF plain literals.

There are some important consequences here.

Firstly, a number of inferences are licensed. For example, the graph below:

```
ex:foo rdf:type skos:Label;
  skos:plainLiteralValue "cow"@en;
  dcterms:modified "2007-09-09".

ex:bar rdf:type skos:Label;
  skos:plainLiteralValue "cow"@en.
```

... entails ...

```
ex:bar dcterms:modified "2007-09-09".
```

I.e. if two instances of skos:Label have the same plain literal value, then they are the same thing.

The danger here is that people coming from the thesaurus community might see skos:Label as being like a "term" in some thesaurus, and attach all sorts of extra information to an instance of skos:Label such as status, source, creation date, modification date etc., not realising that this would be very inappropriate -- because when the data was merged with other thesaurus data, if two "terms" happen to share the same plain literal value, all sorts of inappropriate inferences would be drawn.

I.e. we would have to be very careful to tell a "story" about skos:Label which clearly explains that instances of this class are **not** like thesaurus terms. This would be quite a hard sell I think.

On the other hand, from a logical point of view, this option is good. We have a semantics for skos:Label which essentially makes it equivalent to the class of RDF plain literals. We know exactly the conditions under which two instances of this class are identical -- when they share the same lexical form and the same language -- and the conditions under which they are not identical -- when they have a different lexical form and/or different language. I.e. we can piggyback on the formal definition of literal equality given in [RDF-CONCEPTS].

In fact, the correspondance between skos:Label and the class of RDF plain literals is so close, that for all practical purposes, we might as well state that skos:Label **is** the class of RDF plain literals.

Many-to-One

If we allow two different instances of skos:Label to share the same plain literal value, then an example like the following is allowed:

```
ex1:cow rdf:type skos:Label; skos:plainLiteralValue "cow"@en.
ex2:cow rdf:type skos:Label; skos:plainLiteralValue "cow"@en.
ex1:cow owl:differentFrom ex2:cow.
```

What we have here is a semantics for skos:Label which at least compatible with its use to represent a "term" in a thesaurus. I.e. extra information could be attached to instances of skos:Label, without danger of inappropriate inferences, e.g.:

```
ex1:cow rdf:type skos:Label;
  skos:plainLiteralValue "cow"@en;
  dcterms:created "2007-09-09".

ex2:cow rdf:type skos:Label;
  skos:plainLiteralValue "cow"@en;
  dcterms:created "1903-05-05".
```

... is OK.

The difficulty here comes when we want to tell a "story" about what what an instance of skos:Label actually "is". This difficulty is closely related to the problem of defining under what circumstances two instances of skos:Label **are** identical. What determines the identity of an instance of this class? Can instances have owners, for example? Can they be "in" a concept scheme? Must they be "in" a concept scheme? If two instances have the same plain literal value, and are both "in" the same concept scheme, are they then the same thing?

Remember that we are not only catering for thesauri, but for taxonomies, subject heading systems and classification schemes. Therefore we can't tell a story based entirely on the idea of a "thesaurus term". We have to have a story that is agnostic, and that bridges across these paradigms.

Summary

There are two possible variants of the [SIMPLE-EXTENSION] proposal. In one variant, there is a one-to-one relationship (bijection) between the class extension of skos:Label and the set of RDF plain literals. In the second variant, there is a many-to-one relationship (function) between the class extension of skos:Label and the set of RDF plain literals.

Each of these two variants has significantly different logical consequences, and requires a significantly different "story" to be told. Therefore, each of these two variants should be considered as a distinct proposal in its own right.

To be considered as a proposal, each of these two variants needs further work, especially on the "story" that goes with it.

References

[SKOS-GUIDE]
http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102
[SKOS-SPEC]
http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102
[ISSUE-31]
http://www.w3.org/2006/07/SWD/track/issues/31
[RDF-CONCEPTS]
http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
[RDF-SEMANTICS]
http://www.w3.org/TR/2004/REC-rdf-mt-20040210/
[RFC-4646]
http://www.ietf.org/rfc/rfc4646.txt
[LABEL-SEMANTICS]
http://purl.org/net/skos/2007/10/f2f/labelling-properties.html
[OWL-REFERENCE]
http://www.w3.org/TR/2004/REC-owl-ref-20040210/
[MINIMAL-LABEL-RELATION]
http://www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/Propos
[SIMPLE-EXTENSION]
http://www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/Propos
[REMOVE-RANGE]
http://lists.w3.org/Archives/Public/public-swd-wg/2007Sep/0013.html

---

```
--- Change Log ---
$Log: label-relations.html,v $
Revision 1.3  2007/10/03 11:04:48  ajm65
First completed draft.

Revision 1.2  2007/10/02 17:14:13  ajm65
Minor edita.

Revision 1.1  2007/10/02 17:09:19  ajm65
Initial check-in, part baked.
```

Capturing relationships between labels

This page provides some background and discussion relating to various types of links between labels found in some thesauri, and also in other types of concept scheme. This page relates to ISSUE-26: RelationshipsBetweenLabels.

SKOS allows to represent semantic relationships (broader, related) between concepts. It also allows to represent relationships between concepts and labels (prefLabel, altLabel). However, there is nothing proposed in SKOS to capture links between labels themselves, a configuration which sometimes happens in concept schemes.

Proposals:

- Proposal one "LabelRelation", mail by Guus http://lists.w3.org/Archives/Public/public-swd-wg/2007Feb/0181.html, modified in http://lists.w3.org/Archives/Public/public-swd-wg/2007Feb/0195.html
- Proposal two "LabelAnnotation", mail by Alistair http://lists.w3.org/Archives/Public/public-swd-wg/2007Mar/0092.html
- /ProposalThree "Simple Extension", by Guus, http://lists.w3.org/Archives/Public/public-swd-wg/2007May/0057.html
- /ProposalFour
  "Minimal Label Relation", by Alistair, similar to proposal one but generalised for label relations of any arity.

notice: these proposals, more recent that the wiki page below, do not necessarily propose the same solutions. Roughly proposals one and two are comparable to the *second solution* of this page, while proposal three is comparable to the third solution.

Motivation

The need for links between labels associated to concepts, whether preferred or alternative ones, occurs in the following situations:

- monolingual terminological and lexical knowledge:
  - synonym, e.g. `("bucket", synonym, "pail")` (from EucAimsDetailed)
  - antonym, e.g. `(domestic_flight-noun-1, antonym, international_flight-noun-1` (from Wordnet)
  - abbreviation, e.g. `("Corp.", abbreviation_of, "Corporation")` (from EucAimsDetailed)
  - acronym, e.g. `("Food and Agriculture Organization", acronym, "FAO")` (from EucAimsDetailed)

    - spelling variant, e.g. `("organisation", spelling_variant, "organization")` (from EucAimsDetailed)

  Motivating use cases: EucAimsDetailed; RucPersonalizedTV (using Wordnet); RucHilt, RucIntraLibrary, RucSeaLife and EucBiozen (using MeSH, which includes such links, as mentioned in a mail by Mark van Assem)

- multilingual terminological knowledge
    - translation at term level, e.g. `("vache" (FR), translation, "cow" (EN))` (from EucAimsDetailed).

      Notice: especially relevant when the concept has preferred label `"bovine"` and has as English alternative labels `"cow"`, `"bull"`, and `"buffalo"` which have indovidual translation links to `"vache"`, `"taureau"`, and `"buffle"`.

    - translation between languages from different levels, e.g. `("African violet", scientific_taxonomic_name, "Saintpaulia")`

      Notice: also interesting with `"bovine"` which can be translated into scientific `"bovinae"` while `"buffalo"` should be translated to something else.

  Motivating use cases: EucAimsDetailed, EucIconclassDetailed (though keywords *there are very specific kind of index labels). A mail by Paul Hermans refers to OECD Macrothesaurus and ILO thesaurus cases. another mail by Ron Davies mentions personal experience about these caracteristics.*

Possible modelling approaches

SKOS current version does not provide support for representing links between labels. Worse, **SKOS current version prohibits designing extensions to SKOS vocabulary that would cope with this problem**, e.g. by introducing a simple `abbreviation` RDF properties. Such extensions would indeed require labels to be modeled as RDF resource, which is contradictory to current SKOS specifying RDFS literal as the range of its labelling properties (e.g. `skos:prefLabel`).

First solution: Term-as-class

This solution is described e.g. in Mark van Assem's change proposal. Its goal is to provide a way to represent "terms" as instances of a class so that additional properties can be attached to them. It basically consists of:

- introducing a new class (`Term`, `Label` or any more convenient name) in the SKOS vocabulary
- renaming skos:prefLabel/skos:altLabel to skos:prefTerm/skos:altTerm
- changing the range of these properties into skos:Term

An example of extension to a SKOS vocabulary using this solution for a specific concept scheme could be:

```
ex_SKOSext:acronym rdf:type rdf:property;
  rdfs:domain skos:Term;
  rdfs:range skos:Term.

ex_voc:c rdf:type skos:Concept;
  skos:prefTerm ex_voc:t;
  skos:altTerm ex_voc:u.

ex_voc:t rdf:type skos:Term;
  rdfs:label "Food and Agriculture Organization";
  ex_SKOSext:acronym ex_voc:u.

ex_voc:u rdf:type skos:Term;
  rdfs:label "FAO".
```

Notice: this example uses `rdfs:label` as the property linking the term object to the literal which "embodies" it, but another property might be coined and choose by the SKOS vocabulary for this purpose. Whatever be the name, convenient semantics should be enforced: a `Term` shall not have more than one `rdfs:label` per language.

Second solution: link as annotation based on n-ary relation pattern

This is the solution proposed by Alistair's note on annotation patterns, adapted to the special problem of creating links between terms.

It basically consists of:

- introducing a new class (`skos:LabelRelation`, or any more convenient name) in the SKOS vocabulary, as a specialization of `skos:Annotation` class of SkosDesign/AnnotationPatterns
- introducing new properties `skos:relationSubject` and `skos:relationObject` as specialization of the `skos:annotatesLiteral` property of SkosDesign/AnnotationPatterns
- assigning the domain of these properties to `skos:LabelRelation` and their range as `rdfs:literal`

An example of extension to a SKOS vocabulary using this solution for a specific concept scheme could be:

```
ex_SKOSext:AcronymRelation rdfs:subClassOf skos:LabelRelation.

ex_voc:c rdf:type skos:Concept;
  skos:prefLabel "Food and Agriculture Organization";
  skos:altLabel "FAO".

ex_voc:a rdf:type ex_SKOSext:AcronymRelation;
  skos:relationSubject "Food and Agriculture Organization";
  skos:relationObject "FAO".
```

Third solution: keeping standard SKOS and Term-as-class solutions co-existing

@@ TODO: retrieve reference (solution was mentioned in a mail by Mark) @@

The goal of this alternative is to stay compatible with current simple labelling model, while allowing to represent links between labels. For that it is expected that current SKOS labelling pattern and Term-as-class one can co-exist, together with bridge rules allowing to convert data from one model to the other.

This can be done by:

- introducing `prefTerm` and `altTerm` parallel to keeping `prefLabel` and `altLabel`
- specifying "conversion" axioms such as: [warning, this shall be reformulated properly]

> The interpretation of `skos:prefLabel` shall coincide with the (mathematical) composition of the two relations implied by the interpretations of `skos:prefTerm` and `rdfs:label`

Comparison between two main solutions

http://www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/Comparison

References

Many discussion elements on representing annotation on labels are also relevant for the problem of representing links between labels:

- http://www.w3.org/2004/02/skos/core/proposals.html#thesaurusRepresentation-11 Thesaurus representation issues from the old SKOS Proposals and issues list, linking to different mail threads on representing labels as resources.
- http://www.w3.org/2006/07/SWD/wiki/SkosDesign/AnnotationPatterns on using N-ary relation pattern for representing annotations on labels.

SkosDesign/RelationshipsBetweenLabels (last edited 2007-06-15 08:25:15 by AlistairMiles)

SKOS ISSUE-26: "Minimal Label Relation" Proposal

This is a proposal for resolution of SKOS ISSUE-26 and for a solution to the candidate requirement R-RelationshipsBetweenLabels. See also IssuesProcess.

0. Summary

This proposal is intended to provide a minimal, extensible, mechanism for the representation of relationships of any arity between lexical labels, and for the association of such relationships with other resources (typically the concept which provides the context for the label relationship).

In common with the "LabelRelation" (proposal one) proposal for this issue, this proposal introduces a new class `skos:LabelRelation`.

In this proposal, however, any number of lexical labels may be involved in a label relationship whereas the "LabelRelation"
proposal allows only two labels to be involved. This proposal could therefore be seen as a moderate generalisation of the "LabelRelation" proposal.

1. Vocabulary

The following vocabulary is required by this proposal:

`skos:LabelRelation`	`skos:labelRelated`	`skos:seeLabelRelation`

No vocabulary is deprecated by this proposal.

2. Axiomatic Triples

The following triples are part of the normative semantics for this proposal:

```
skos:labelRelated rdfs:domain skos:LabelRelation.
skos:labelRelated rdfs:range rdfs:Literal.
skos:seeLabelRelation rdfs:range skos:LabelRelation.
skos:LabelRelation owl:disjointWith skos:Concept.
skos:LabelRelation owl:disjointWith skos:Collection.
skos:LabelRelation owl:disjointWith skos:ConceptScheme.
```

3. Semantic Conditions

There are no further semantic conditions on the vocabulary of this proposal.

4. Consistent Examples

Some simple examples:

```
[] skos:labelRelated "FAO"@en; skos:labelRelated "Food and Agriculture
Organization"@en.

[] skos:labelRelated "cow"@en; skos:labelRelated "vache"@fr.

ex:foo
  skos:seeLabelRelation [
    skos:labelRelated "foo"@en;
    skos:labelRelated "bar"@en;
  ].
```

N.B. the vocabulary of this proposal isn't intended to be used directly, but as an extension point, for example:

```
ex:AcronymRelation rdfs:subClassOf skos:LabelRelation.
ex:fullForm rdfs:subPropertyOf skos:labelRelated; rdfs:domain
ex:AcronymRelation.
ex:acronymForm rdfs:subPropertyOf skos:labelRelated; rdfs:domain
ex:AcronymRelation.

ex:FAO rdf:type skos:Concept;
  skos:prefLabel "Food and Agriculture Organization"@en;
  skos:altLabel "FAO"@en;
  skos:seeLabelRelation [
    ex:fullForm "Food and Agriculture Organization"@en;
    ex:acronymForm "FAO"@en;
  ].
```

5. Inconsistent Examples

It is not possible to make inconsistent statements using the vocabulary of this proposal, other than by contradicting the disjointness axioms for the class `skos:LabelRelation` stated in the axiomatic triples above.

See also note on entailment directly below.

6. Entailment Rules

There are no entailments other than those that follow from the RDF and OWL semantics and the axiomatic triples given above.

N.B. This means that, for example, there does not necessarily have to be any correspondance between the lexical labels of a resource, and the labels involved in a label relation, to which the resource is related via the `skos:seeLabelRelation` property.

7. Syntactic Constraints

The following syntactic conditions apply to the vocabulary of this proposal:

> An application MAY ignore any triple in an RDF graph where the predicate is `skos:labelRelated` and the object is not an RDF plain literal.

8. Discussion

This proposal is intended to satisfy requirement R-RelationshipsBetweenLabels in the simplest possible way, and to avoid any complicated or possibly contentious logical consequences.

- [RDF Concepts] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
- [RDF Semantics] http://www.w3.org/TR/2004/REC-rdf-mt-20040210/
- [RFC 2199] http://www.ietf.org/rfc/rfc2119.txt

CategoryTemplate

SkosDesign/RelationshipsBetweenLabels/ProposalFour (last edited 2007-06-15 09:02:48 by AlistairMiles)

SKOS ISSUE-26 Relations between Labels: "SimpleExtension" Proposal

This is a proposal for resolution of SKOS ISSUE-26. See also IssuesProcess.

Resolution of this issue should enable us to satisfy candidate requirement R-RelationshipsBetweenLabels (see the SKOS Use Cases and Requirements document).

0. Summary

The proposal extends SKOS with the possibility to define a lexical label of a SKOS concept as a resource, such that statements can be made about it. To this end we introduce the class skos:Label plus the corresponding properties (pref, alt, hidden) which link a concept to a label resource. The property skos:labelRelation can be used to express relations between labels. Applications will typically specialize this to define particular label relations. The proposal ensures OWL DL compatibility.

1. Vocabulary

The proposal introduces the following new vocabulary:

skos:Label
skos:prefLabelR
skos:altLabelR
skos:hiddenLabelR
skos:conceptLabelR
skos:labelRelation

The <x>LabelR properties are the counterparts of the <x>Label properties and point to a resource instead of a literal.

No vocabulary is deprecated by this proposal.

2. Axiomatic Triples

RDF statements:

```
skos:Label rdf:type rdfs:Class.
skos:conceptLabelR rdf:type rdf:Property.
skos:conceptLabelR rdfs:domain skos:Concept.
skos:conceptLabelR rdfs:range skos:Label.
skos:prefLabelR rdf:type rdf:Property.
```

```
skos:prefLabelR rdfs:subPropertyOf skos:conceptLabelR.
skos:altLabelR rdf:type rdf:Property.
skos:altLabelR rdfs:subPropertyOf skos:conceptLabelR.
skos:hiddenLabelR rdf:type rdf:Property.
skos:hiddenLabelR rdfs:subPropertyOf skos:conceptLabelR.
skos:labelRelation rdf:type rdf:Property.
skos:labelRelation rdfs:domain skos:Label.
skos:labelRelation rdfs:range skos:Label.
```

OWL statements:

```
skos:Label rdf:type owl:Class.
skos:conceptLabelR rdf:type owl:ObjectProperty.
skos:prefLabelR rdf:type owl:ObjectProperty.
skos:altLabelR rdf:type owl:ObjectProperty.
skos:hiddenLabelR rdf:type owl:ObjectProperty.
skos:labelRelation rdf:type owl:ObjectProperty.
```

@@todo Add OWL restriction on class skos:Label that for this class the cardinality of rdfs:label must be to precisely 1.

3. Semantic Conditions

- In case of multiple prefLabelR triples with the same subject (a skos:Concept) each object (a skos:Label) should have a unique language tag for its rdfs:label value.
- A skos:Concept must be the subject of at least one one skos:prefLabel or skos:prefLabelR triple. NOTE: there seems to be no reason to forbid mixing of label and class approach. So the "or" is not an exclusive or.

4. Consistent Examples

EXAMPLE 1

The example below defines three labels for a concept, where one label is defined as the acronym of another label.

```
ex:who rdf:type skos:Concept;
  skos:prefLabelR ex:who1;
  skos:prefLabelR ex:who2;
  skos:altLabelR ex:wh03.
ex:who1 rdf:type skos:Label;
  rdfs:label "World Health Organization"@en-us.
ex:who2 rdf:type skos:Label;
  rdfs:label "Wereldgezondheidsorganisatie"@nl.
ex:who3 rdf:type skos:Label;
  rdfs:label "WHO"@en.
ex:acronymOf rdf:type rdf:Property;
  rdf:subPropertyOf skos:labelRelation.
ex:who3 ex:acronymOf ex:who1.
```

5. Inconsistent Examples

EXAMPLE 2

```
ex:who rdf:type skos:Concept;
  skos:prefLabelR ex:who1;
  skos:prefLabelR ex:who2;
ex:who1 rdf:type skos:Label;
  rdfs:label "World Health Organization"@en.
ex:who2 rdf:type skos:Label;
  rdfs:label "World Health Organisation"@en.
```

EXAMPLE 3

```
ex:who rdf:type skos:Concept;
  skos:prefLabelR ex:who1;
  skos:prefLabelR ex:who2;
ex:who1 rdf:type skos:Label;
  rdfs:label "World Health Organization".
ex:who2 rdf:type skos:Label;
  rdfs:label "Wereldgezondheidsorganisatie"@nl.
```

6. Entailment Rules

DISCUSSION: should we have an entailment rule that says something like:

for every conceptLabelR statement (i.e. either prefLabelR, altLabelR or hiddenLabelR) tools are allowed to assert a <x>Label triple for each of the rdfs:label statements of the object of the conceptLabelR statement.

So from example 1, applications may derive the following additional triples:

```
ex:who
  skos:prefLabel "World Health Organization"@en-us;
  skos:prefLabel "Wereldgezondheidsorganisatie"@nl;
  skos:altLabel "WHO"@en.
```

DISCUSSION: should the inverse also be true? E.g should applications be allowed to derive the following:

```
ex:who
  skos:prefLabel "World Health Organization"@en-us;
  skos:prefLabel "Wereldgezondheidsorganisatie"@nl;
  skos:altLabel "WHO"@en.
```

entails

```
ex:who
 skos:prefLabelR [a skos:Label
   rdfs:label "World Health Organization"@en-us];
 skos:prefLabelR [a skos:Label
   rdfs:label "Wereldgezondheidsorganisatie"@nl];
 skos:altLabelR [a skos:Label
   rdfs:label "WHO"@en].
```

NOTE: the difference with example 1 is here that the instances of skos:Label are represented as blank nodes.

7. Syntactic Constraints

@@todo

8. Discussion

The objective of this proposal is to allow label relations, while preserving compatibility with the label-as-literal approach. The main alternative would be to drop the semantic constraint on the <x>Label properties to be an owl:DatatypeProperty. This would make the following example consistent:

```
ex:who
  rdf:type skos:Concept;
  skos:prefLabel ex:who1.
ex:who1
  rdf:type skos:Label;
  rdfs:label "World Health Organization"@en-us.
```

even in combination with:

```
ex:who
  skos:prefLabel "World Health Organization"@en-us;
```

One can view this as the MinimalFix proposal but has two disadvantages:

- ambiguity wrt the range of the skos:<x>Label properties
- incompatibility with OWL DL, as it is not possible to define the label properties as either an objector a datatype property.

The SimpleExtension proposal preserves OWL DL compatibility. The entailment rules mentioned under 6 are meant to preserve interoperability between vocabularies that use different approaches to represent lexical labels.

- [Discussion Thread] http://lists.w3.org/Archives/Public/public-swd-wg/2007May/0057.html

CategoryTemplate

SkosDesign/RelationshipsBetweenLabels/ProposalThree (last edited 2007-07-24 13:24:42 by GuusSchreiber)

[SKOS] Compatibility with OWL-DL - Issue-26

From: Thomas Baker <baker@sub.uni-goettingen.de>
Date: Wed, 5 Sep 2007 13:49:19 +0200
To: "Miles, AJ (Alistair)" <A.J.Miles@rl.ac.uk>
Cc: SWD Working Group <public-swd-wg@w3.org>
Message-ID: <20070905114919.GA3500@sub-tombaker>

```
Alistair,

On Tue, Sep 04, 2007 at 04:57:27PM +0100, Alistair Miles wrote:
> Therefore I...
>
> PROPOSE that we build the SKOS semantics on OWL Full, and postpone the
> issue of OWL DL compatibility [ISSUE-38].

+1

> If we accept this proposal, then we will have to work out exactly what
> it means for the design principles we use to construct the SKOS
> Semantics. It might mean, for example, that we can declare all SKOS
> properties as either datatype properties or object properties, and not
> use owl:AnnotationProperty at all.

Issue 26 -- "Relationships between labels" -- would be impacted
by such a decision [1,2].  Maybe we could go back to a suggestion
Guus made in March [3] as an alternative to Proposal Three
("SimpleExtension") [4]:

    I had an amendment of this third proposal in mind.
    Instead of having two properties skos:prefTerm and
    skos:prefLabel, I would suggest to have just the
    current one, skos:prefLabel, and removing the
    range restriction (rdfs:literal). So, this means
    that if one queries for the the skos:prefLabel of
    a concept, one either gets a literal or a resource
    with a label equal to this literal. This prevents
    the use of construction rules and keeps the SKOS
    vocabulary simple. The only extension to the
    current SKOS vocabulary would a a class skos:Term.

Tom

[1] http://www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/
[2] http://www.w3.org/2006/07/SWD/track/issues/26
[3] http://lists.w3.org/Archives/Public/public-swd-wg/2007Mar/0004.html
[4] http://www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBetweenLabels/ProposalThree

--
Tom Baker - tbaker@tbaker.de - baker@sub.uni-goettingen.de
```

Received on Wednesday, 5 September 2007 11:49:38 GMT

RE: [SKOS] Amsterdam topic: "Label relations"

From: Miles, AJ \(Alistair\) <A.J.Miles@rl.ac.uk>
Date: Wed, 3 Oct 2007 13:13:47 +0100
Message-ID: <677CE4DD24B12C4B9FA138534E29FB1D0363B72E@exchange11.fed.cclrc.ac.uk>
To: "Thomas Baker" <baker@sub.uni-goettingen.de>, "SWD Working Group" <public-swd-wg@w3.org>, "Guus Schreiber" <schreiber@cs.vu.nl>
Cc: <public-esw-thes@w3.org>

Hi all,

For the record, here is my position on the three options:

> > > Three options will be discussed in Amsterdam
> > > *
> [http://www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBe
> tweenLabels/ProposalFour
> > > Alistair's "Minimal Label Relation" proposal]

I think this proposal is mature enough for consideration. There aren't any difficult or complicated logical semantics or dependencies on other parts of the SKOS vocabulary, and "telling the story" would be quite straighforward. We might need to discuss further about the precise meaning or name of the different components e.g. skos:seeLabelRelation, but we could at least agree to adopt this proposal *as a pattern*, then work on the details after.

If we have to make a decision at the f2f, I would recommend we agree to adopt the pattern used in this proposal, then work on the finer details of choosing URIs etc.

> > > *
> [http://www.w3.org/2006/07/SWD/wiki/SkosDesign/RelationshipsBe
> tweenLabels/ProposalThree
> > > Guus "Simple Extension" proposal]

I think this proposal needs further work, even as a pattern, and is not ready for consideration.

The main reason is that there are two possible variants of this proposal, which are actually very different in nature, which could each be considered as a proposal in their own right (given a bit more work on the story). See <http://purl.org/net/skos/2007/10/f2f/label-relations.html> for more explanation of this point.

> > > *
> [http://lists.w3.org/Archives/Public/public-swd-wg/2007Sep/001
> 3.html Guus "remove range
> > > restriction for skos: prefLabel" proposal]

I would like to permanently rule out this proposal.

The main reason is that having a complex range for the properties skos:prefLabel, skos:altLabel and skos:hiddenLabel makes for a very complicated semantics, which I don't know how to construct. For example, specifying the disjointess of these properties, or the cardinality of skos:prefLabel, would become very difficult, if not impossible. It's already hard enough to do this with just a simple range as the class of RDF plain literals. See <http://purl.org/net/skos/2007/10/f2f/labelling-properties.html> for more explanation of this point.

-- Conclusion --

If we *have* to make a decision at the f2f, then I think the only proposal which can be considered is the "Minimal Label Relation" proposal. I would recommend we agree to adopt the pattern illustrated in this proposal, then work some more on the details.

If we don't have to make a decision at the f2f, then we could do some more work on elaborating the two variants of the "Simple Extension" proposal, then make a decision in a couple of months.

Being absolutely ruthless, I don't think label relations are mission-critical for SKOS. I.e. SKOS could be a successful specification and meet it's most important requirements without built-in support for label relations. Support for label relations could always be developed as extensions/add-ons to SKOS within the community of practice. Therefore, I think delaying the decision (or even postponing the requirement) would be acceptable.

On the other hand, I do think label relations would significant value to SKOS -- it would be nice if we could work out a solution.

Cheers,

Alistair.

--
Alistair Miles
Research Associate
Science and Technology Facilities Council
Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Didcot
Oxfordshire OX11 0QX
United Kingdom
Web: http://purl.org/net/aliman
Email: a.j.miles@rl.ac.uk
Tel: +44 (0)1235 445440

Received on Wednesday, 3 October 2007 12:23:26 GMT

Basic Principles for Managing an RDF Vocabulary

Based on starting point:
http://www.w3.org/2001/sw/BestPractices/VM/principles/20050705

Abstract

This document articulates some basic principles of good practice for managing an RDF vocabulary. Following these principles makes an RDF vocabulary "usable": new users learn quickly how to use the vocabulary, and a relationship of trust is built between the user community and the vocabulary developers/maintainers. This promotes growth of a user community, which generates more feedback for the developers/maintainers, leading to further improvements in quality and usability.

This document focuses on those principles of good practice where a clear recommendation can be made. A number of issues related to the management of RDF vocabularies have yet to be resolved, but these are outside the scope of this document. Further, there are a number of ways to address most, if not all, of the topics highlighted below. While this document does not attempt to provide an exhaustive survey of those methodologies/approaches, it is intended to provide pointers to approaches that have worked well for seasoned practitioners.

Introduction

An RDF vocabulary is a set of resources denoted by URIs. Informally, these resources are known as the "terms" of the vocabulary. The resources will usually (but not necessarily) be of type rdf:Property, rdfs:Class, owl:Class, or skos:Concept.

An RDF vocabulary is created and maintained for the use of a community of people (the 'user community') as a set of building blocks for creating RDF descriptions of things in their domain of interest. An RDF vocabulary usually implies a shared conceptualisation, and thus the notion of an 'RDF vocabulary' is almost identical to the notion of a 'web ontology' [ref???]. (efk: consider taking this last sentence out ...)

Many controlled vocabularies have been encoded in RDF, OWL, and other knowledge representation languages, and a growing number of these are available in the public domain. A fraction of these appear to have fostered significant reuse to date, however [ref. recent discussion thread on mapping the Semantic Web]. While there are many issues that can limit reuse opportunities, a significant contributor is the lack of well-specified policies for vocabulary management, metadata, and provenance specification, depending on the application. Several of the most prominent RDF vocabularies currently in use (e.g., OWL, FOAF, Dublin Core, SKOS Core) have emerged from a close collaboration between a relatively small community of developers and a larger community of users. The prominence

of these vocabularies may be attributed to their utility, but also to the commitment made by those responsible for developing/maintaining the vocabularies to forming, accomodating, serving, and working with, a community of users.

In addition to these individual vocabularies, a number of portals are emerging as 'collection points' for vocabularies designed to support users in specific domains, such as the BioPortal from NCOR (National Center for Biomedical Ontology - http://www.bioontology.org/) or specific communities, such as the OMG's Ontology Portal (coming soon to http://ontology.omg.org/). Such portals are useful for users searching for the vocabularies they serve, but also because of the significant metadata describing the ontologies that they provide. Increasingly, the metadata describing a particular vocabulary is becoming as important as the vocabulary itself for documentation and reuse purposes.

The goal of implementing the principles outlined in this document is to make an RDF vocabulary "usable". This could be restated as, managing an RDF vocabulary in such a way that it can easily be understood and deployed by users.

(paragraph on digital preservation, per email from Dan Brickley re: social responsibility, etc.)

.... [some other stuff ???]

Principles of Good Practice

1. Use URIs For Naming

An RDF vocabulary consists of a set of URIs. 'Naming' refers to the act of allocating URIs to resources [ref. RDF Semantics, http://www.w3.org/TR/rdf-mt/#urisandlit].

The developers/maintainers of an RDF vocabulary should inform the potential user of the following:

- The URI space from which resource names are drawn.
- The ownership of this URI space.
- Any commitments made by the owner(s) of the URI space to the persistence of URIs in that space.
- Policies for delegation of responsibility for allocating URIs within that space to vocabulary developers/maintainers by owners of that space.
- Rules used by the developers/maintainers for constructing URIs to be used as resource names.

These practices are among the most critical for ensuring that potential users can trust that a particular RDF vocabulary is stable, will persist for some length of time, and can be either referenced or used in applications they are building.

For example, in developing applications that use the OWL-S vocabularies (http://www.w3.org/Submission/OWL-S/), there have been times when certain vocabularies on which OWL-S depends, such as vocabularies representing the names of cities and states in the US, return 404 errors (i.e., they are unavailable, and thus the corresponding applications may fail unless they have a locally cached version). This appears to be a reflection of a temporary

outage, but is an issue for developers nonetheless. Additionally, the standards described by certain dependent ontologies are constantly evolving, for example, language and country codes, while the referenced ontologies themselves appear to be static and aging. The end result is that users are less confident of the availability or applicability of these dependent vocabularies and thus of OWL-S itself.

In cases where access to such "utility" vocabularies is critical for many applications, there are discussions underway with authoritative organizations regarding development and management of key vocabularies. For example, the Library of Congress is the registration authority for parts of ISO 639 (language codes) and ISO 3166 (country codes), and thus the obvious choice to manage and maintain the corresponding RDF vocabularies (http://www.loc.gov/standards/). Until such time as "all of the vocabularies we might need" become available from an authoritative source, or for any of us considering publishing vocabularies designed specifically for reuse, the minimal set of naming conventions identified above should be the starting point.

Guidelines for choosing URI namespaces, including considerations and examples to assist in the process are provided in [ref. Recipes]. In addition, a number of organizations are grappling with decisions regarding general URI schemes, such as the date-based scheme generally used by the W3C. Some communities, such as the OMG, have found that as the number of documents and communities within the broader organization grows, dates alone may not be sufficient. In order to assist potential users in finding various artifacts on the OMG site, recent proposals suggest including the higher level specification name, date-based version information, artifact type, and so forth as part of the subordinate URI scheme. Use of a simple RDF vocabulary to support this scheme and assist in navigation, once adopted, is also being discussed.

E.g.s ....

```
F2F-01-2007:
This should be a hard recommendation.
Naming conventions could be linked to here from the cookbook
Good place to mention the domain registration problem
```

Paragraph on current practice for Dublin Core (Tom?) Paragraph on current best practices identified through the BioPortal (Daniel / Natasha?)

2. Provide Readable Documentation

The developers/maintainers of an RDF vocabulary should provide natural-language (i.e. human-readable) documentation about the vocabulary and its proper use. The principle aim of this documentation is to help potential users *learn* how to apply the vocabulary, and therefore to promote *consistency* in the way that the vocabulary is applied. Inconsistent usage reduces the value of a vocabulary, because the meaning associated with the vocabulary becomes in practice ambiguous.

As a bare minimum, a list of the terms should be published, with text definitions. It is recommended to publish detailed prose describing proper usage patterns and scenarios, with examples. Metadata should include a description of the use-case(s) that were the basis for the original vocabulary development, intented audience and target domain, references and authoritative sources used, development and validation methodology, and so forth.

Egs.

```
F2F-01-2007:
Both human readable and usage of machine readable (rdf:seeAlso)
Could be linked to the cookbook
show examples -- dc-terms, skos vocabulary, owl wordnet
```

3. Articulate Maintenance Policies

An RDF vocabulary may be developed in private by a closed community, and then published with no possibility for future change. An RDF vocabulary may, on the other hand, be developed in public by an open community, with the content of the vocabulary being allowed to evolve indefinitely. In any case, a potential user needs to know under what circumstances the vocabulary (or parts of it) may change, and what kinds of change may be expected.

The key concept here is 'stability'. When a potential user chooses a vocabulary, they are making an investment of time/money/effort that depends to a certain extent upon the stability of that vocabulary. Therefore a potential user needs to know exactly how stable a vocabulary is, in order to judge how much to invest. If a vocabulary is less than perfectly stable, the user needs to know exactly what may change, how it may change, and of course to be informed of changes when they do occur.

Therefore, the developers/maintainers of an RDF vocabulary should publish a maintenance policy for that vocabulary. The maintenance policy should articulate whether or not change is allowed, and the way that change is managed.

Egs.

The developers/maintainers should also provide some facility whereby users can be informed of changes as and when they are made.

Egs.

```
F2F-01-2007:
Point to some examples of different types of vocabularies
and their maintenance and persistence policies

Paragraph discussion (Elisa / Evan / Jishnu / Pete) on metamodel
management per OMG experiences
Paragraph discussion (Alistair?) per SKOS experience

For instance...
 http://www.w3.org/1999/10/nsuri "URIs for W3C Namespaces"
 discusses namespace maintenance
```

4. Identify Versions

Where a vocabulary is allowed to change, users developing systems based on that vocabulary may prefer to work to a stationary, rather than moving, target. To support these users, the developers/maintainers of a vocabulary should:

- Publish versions of the vocabulary, where a 'version' is a 'snapshot' of the vocabulary at a particular point in time.
- Allocated URIs to vocabulary versions, so that they may be referred to.

Where the resources that are the members of a vocabulary may evolve independently, or be at differing levels of stability, the developers/maintainers may also which to allocate URIs to historical versions of a particular resource.

Egs.

```
F2F-01-2007:
Discuss and display various methods of versioning
and maintenance policies with reference implementations

Tim and Alistair have suggested micro-ontologies (Alistair:
<http://purl.org/net/d4>) that might be articulated
for publishing the relationship between versions using OWL.

For instance...
http://www.w3.org/2006/07/SWD/wiki/BestPracticeRecipesIssues/ServingSnapshots
http://dublincore.org/usage/terms/history/
Knowledge Web
http://ontology.buffalo.edu/bfo/Versioning.pdf
http://www3.lehigh.edu/images/userImages/jgs2/Page_3813/LU-CSE-06-026.pdf
```

Note that version management for RDF vocabularies and, in particular, for OWL ontologies, is an ongoing research problem, and there is no single approach that may be appropriate in all situations. (Provide background on configuration management for ontologies, e.g., Jeff Hefflin's work cited above, Protege approach, issues raised for DARPA/REAL, etc.. Also need examples to include ontology versioning with dependencies / broken reasoning to make the point. Vit/Siggi point out that reasoning with change management in mind may include additional metadata to be encoded in the vocabulary as well as documentation indicating how such metadata supports doing so -- a decent discussion of this with pointers to reference implementations might be useful.)

Might also provide an example using named graphs -- check with Jeremy for decent examples

From Vit/Siggi, we should also discuss issues related to "human-readable" configuration / version management, as for some vocabularies, human usability is as important as machine-readability.

5. Publish a Formal Schema

An RDF description of an RDF vocabulary should be published. Potential users should be clearly informed as to which is the 'authoritative' RDF description of an RDF vocabulary.

Where the resources that are the members of an RDF vocabulary are denoted by HTTP URIs, an HTTP GET request with the header field 'accept=application/rdf+xml' against that URI should return an RDF/XML serialisation of an RDF graph that includes a description of the denoted resource.

```
F2F-01-2007:
we should say that it is best practice to publish an RDF/OWL document at
the namespace URI.
This may be obvious to us but evidently it's not obvious to everyone.
Point back to Recipes document for "... and here's how"
```

- See also:
  - http://www.w3.org/TR/xmlschema-guide2versioning/ (Sep 26)
  - http://esw.w3.org/topic/ConfigurationManagement
  - http://www.w3.org/2001/sw/BestPractices/VM/
  - http://metadataregistry.org/wiki/index.php/SKOS_Concept_History_Management
  - http://jonphipps.wordpress.com/2006/12/04/skos-concept-history-management-metadata-re

VocabMgtDraft (last edited 2007-08-21 05:40:02 by ElisaKendall)

Consolidated Recipes Issues List for Amsterdam Face-to-Face

Last revision: 24 September 2007

ISSUE-16

Default behavior

State:
OPEN
Product:
Recipes
Raised by:
Jon Phipps
Opened on:
2007-02-13
Description:

```
http://www.w3.org/TR/swbp-vocab-pub/#default
Currently says: "Performing content negotiation based on the value of the
'User-agent:' header field is not generally considered good practice, @@TODO why."
```

Related emails:

1. [RECIPES] 'Issue 1.3' (from baker@sub.uni-goettingen.de on 2007-09-15)
2. [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-15)
3. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from qreul@csd.abdn.ac.uk on 2007-09-18)
4. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from aisaac@few.vu.nl on 2007-09-18)
5. [ALL] agenda 25 Sep telecon - 1500 UTC (from schreiber@cs.vu.nl on 2007-09-24)
6. RE: [ALL] agenda 25 Sep telecon - 1500 UTC (from michael.hausenblas@joanneum.at on 2007-09-24)
7. RE: [ALL] agenda 25 Sep telecon - 1500 UTC (from vit.novacek@deri.org on 2007-09-24)

Related notes:

2007-02-13: This was discussed at the F2F meeting in Boston and Ralph agreed to write a brief description of how User Agent-based negotiation limits/stifles the creation of new User Agents in the future, citing past negative experience.

2007-09-15: http://www.w3.org/2007/04/24-swd-minutes.html#action09
http://www.w3.org/2007/01/22-swd-minutes.html#action14

ISSUE-17

Recipe 6 is incomplete

State:
OPEN
Product:
Recipes
Raised by:
Jon Phipps
Opened on:
2007-02-13
Description:

http://www.w3.org/TR/swbp-vocab-pub/#recipe6

says: "@@TODO this recipe is included as a placeholder, as its best practice implementation currently requires further investigation, discussion and testing, but is anticipated to be an important part of this document, 'completing the set' of the most commonly needed configurations."

Related emails:

1. [Recipes] discussion on Recipe 6 (and regrets) (from diego.berrueta@fundacionctic.org on 2007-03-18)

Related notes:

2007-02-13: This was discussed at the F2F meeting in Boston in January 2007. Diego will perform "further investigation, discussion and testing" and write a first draft of the recipe.

2007-09-24: There is a new draft here: http://www.w3.org/2006/07/SWD/wiki/BestPracticeRecipe6

ISSUE-18

QA Review comments from Karl Dubost

This issue has intentionally not been included

ISSUE-19

Recipes should supply a general server configuration template

State:
OPEN
Product:
Recipes
Raised by:
Jon Phipps
Opened on:
2007-02-13
Description:

```
The Recipes are specific to an Apache server, but may be applicable in
non-Apache environments. It would be useful to provide a general-purpose
configuration template for use by people setting up non-Apache servers"

As part of his QA review KKarl Dubost said...
"...here there's a good opportunity to create a template and/or invite people to
submit bindings to the Mailing List, id est how people applied this recipe in
this particular server environment, Web Apps Framework, HTTP Servers only, etc.
That would help other people to find the information. With a proposed template,
it would help people to collect, gather the information."

http://www.w3.org/2006/07/SWD/wiki/BestPracticeRecipesIssues/QaReviewSummary#head-3d2
```

Related emails:

1. [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-15)
2. [RECIPES] Amsterdam topic 'Recipes' (from baker@sub.uni-goettingen.de on 2007-09-15)
3. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from qreul@csd.abdn.ac.uk on 2007-09-18)
4. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from aisaac@few.vu.nl on 2007-09-18)

Related notes:

No additional notes.

ISSUE-20

Online server testing

State:
OPEN
Product:
Recipes
Raised by:

Jon Phipps

Opened on:

2007-02-14

Description:

```
The recipes provide example URIs and example responses. Diego has written some
httpUnit tests. Since the tests already exist, it would be very useful to make
these available online to provide a server validation service.

See Diego's email:
http://lists.w3.org/Archives/Public/public-swd-wg/2006Dec/0048.html
```

Related emails:

1. [Recipes] towards an online validation tool for best practices (from diego.berrueta@fundacionctic.org on 2007-03-16)
2. [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-15)
3. [RECIPES] Amsterdam topic 'Recipes' (from baker@sub.uni-goettingen.de on 2007-09-15)
4. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from qreul@csd.abdn.ac.uk on 2007-09-18)
5. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from aisaac@few.vu.nl on 2007-09-18)

Related notes:

No additional notes.

ISSUE-21

Apache configuration should add that mod_rewrite must be loaded and enabled

This issue has intentionally not been included

ISSUE-22

Questioning reference to 'IE6 hack'

State:

OPEN

Product:

Recipes

Raised by:

Bernard Horan

Opened on:

2007-02-14

Description:

```
In http://www.w3.org/TR/swbp-vocab-pub/#negotiation there is reference
to a hack that's required for Internet Explorer browser clients. The
paragraph begins as follows:

"In recipes 3, 4, 5, and 6 below, RDF/XML is configured as the default
response. This is chosen to minimize the impact on deployed Semantic Web
applications that do not currently send appropriate 'Accept:' header
field values for RDF content. Note that, however, with RDF as the
default response, a 'hack' has to be included..."

The issue I'd like to raise is two fold:

1) wordsmithing:

a) Suggest that there's a new para/section titled something like
"Workaround for Internet Explorer". At the moment the details of the
hack merge in with the rest of the default behaviour.

b) the use of "hack" and "peculiar" is somewhat pejorative!

c) the layout of the itemised instructions is confusing to read, as
they're broken up by a yellow line of directive


2) ambiguity
I think we need to come down on one side of the fence on whether this
"hack" should be included. Either (a) we remove from the document any
suggestion that the reader should delete the directive and just insert
an explanation as to why it's needed; or (b) explain that IE clients
will need this directive and include it _commented out_ in the recipes
so that implementers may include it. I favour (a).
```

Related emails:

1. [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-15)
2. [RECIPES] Amsterdam topic 'Recipes' (from baker@sub.uni-goettingen.de on 2007-09-15)
3. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from qreul@csd.abdn.ac.uk on 2007-09-18)
4. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from aisaac@few.vu.nl on 2007-09-18)

Related notes:

2007-02-14: There was some discussion about this at the January F2F in Boston: http://www.w3.org/2007/01/22-swd-minutes.html#action12 (scroll up from there).

2007-02-14: Diego was asked to verify that the hack was still necessary for IE7 (http://www.w3.org/2007/01/22-swd-minutes.html#action13) and he reported that it did (http://lists.w3.org/Archives/Public/public-swd-wg/2007Jan/0076.html)

2007-02-14: Corrected links for above note. See: http://www.w3.org/2007/01/22-swd-minutes.html#action13 -- and he reported that it did. See: http://lists.w3.org/Archives/Public/public-swd-wg/2007Jan/0076.html

ISSUE-23

There should be some discussion of alternatives to .htaccess

State:
OPEN
Product:
Recipes
Raised by:
Jon Phipps
Opened on:
2007-02-14
Description:

```
Many organizations, for performance and security reasons, don't allow the use of
AllowOveride and this precludes the use of .htaccess files completely. We should
include a section that specifically addresses this:

1. Briefly discuss the relative performance and security disadvantages of
.htaccess files, or at least point to such a discussion.

2. Discuss the possibility of enabling AllowOveride on a directory-specific
rather than a global basis to address those issues

3. Point out that each of the recipes can be implemented directly in httpd.conf
files without enabling AllowOveride and show how to do that for at least one of them

4. Include a recipe that rewrites the URL to a very simple cgi script (examples
in perl, php) that would handle the actual redirect. This appears to be the way
the recipes are often implemented.
```

Related emails:

1. [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-15)
2. [RECIPES] Amsterdam topic 'Recipes' (from baker@sub.uni-goettingen.de on 2007-09-15)
3. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from qreul@csd.abdn.ac.uk on 2007-09-18)
4. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from aisaac@few.vu.nl on 2007-09-18)

Related notes:

No additional notes.

ISSUE-24

Additional text explaining redirect choices in the recipes

State:
OPEN
Product:
Recipes
Raised by:
Jon Phipps
Opened on:
2007-02-14
Description:

```
There have been a number of instances where Alistair et al have answered
questions about the reasoning behind the use of 'conditional rewrites' and
redirects in the recipes (I'm sure there are more):

http://lists.w3.org/Archives/Public/public-swbp-wg/2005Dec/0016.html
http://lists.w3.org/Archives/Public/public-swbp-wg/2006Feb/0076
http://dowhatimean.net/2006/11/content-negotiation-with-hash-uris-long#comment-15928

It would be a good idea to answer those questions directly in the document,
either at the top or in an Appendix.
```

Related emails:

1. [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-15)
2. [RECIPES] Amsterdam topic 'Recipes' (from baker@sub.uni-goettingen.de on 2007-09-15)
3. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from qreul@csd.abdn.ac.uk on 2007-09-18)
4. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from aisaac@few.vu.nl on 2007-09-18)

Related notes:

No additional notes.

ISSUE-30

Determine how and if RDDL relates to the Recipes

State:
RAISED
Product:
Recipes
Raised by:
Alistair Miles
Opened on:

2007-03-13

Description:

http://lists.w3.org/Archives/Public/public-swd-wg/2006Dec/0086.html

```
Figuring out if and how the cookbook and RDDL [1] fit together is probably
something we should look at ...

[1] http://en.wikipedia.org/wiki/RDDL

related:
http://norman.walsh.name/2006/12/18/rddl

Associating Resources with Namespaces
Draft TAG Finding 13 December 2005
http://www.w3.org/2001/tag/doc/nsDocuments/
```

Related emails:

1. ISSUE-30: Determine how and if RDDL relates to the Resipes (from dean+cgi@w3.org on 2007-03-13)
2. RE: ISSUE-30: Determine how and if RDDL relates to the Resipes (from A.J.Miles@rl.ac.uk on 2007-03-19)

Related notes:

No additional notes.

ISSUE-58

.htaccess 'accept header' ONLY responds to a header which EXACTLY matches

State:

OPEN

Product:

Recipes

Raised by:

Jon Phipps

Opened on:

2007-07-17

Description:

```
Per Tim BL

"The recipe for responding to an accept header only responds to a header which
EXACTLY matches "application/rdf+xml". However, a client may send (and often
will) a header with many  comma-separated values, and they may have quality
parameters (q=0.xx).
```

```
This is a serious problem as people are copying the recipe, and making sites
which do not work."

This applies to Recipes 3, 4, 5 and references this part of the .htaccess Apache
configuration:


"...

# Rewrite rule to serve RDF/XML content if requested
RewriteCond %{HTTP_ACCEPT} application/rdf\\+xml

..."

The rewrite condition regular expression: "application/rdf\\+xml"
needs to be rewritten
```

Related emails:

1. ISSUE-58: .htaccess \\'accept header\\' ONLY responds to a header which EXACTLY matches \\'application/rdf+xml\\' (from dean+cgi@w3.org on 2007-07-17)
2. Re: [Recipes] ISSUE-58: .htaccess 'accept header' ONLY responds to a header which EXACTLY matches \\'application/rdf+xml\\' (from diego.berrueta@fundacionctic.org on 2007-07-18)
3. RE: [Recipes] ISSUE-58: .htaccess 'accept header' ONLY responds to a header which EXACTLY matches \\'application/rdf+xml\\' (from A.J.Miles@rl.ac.uk on 2007-07-24)
4. Re: [Recipes] ISSUE-58: .htaccess 'accept header' ONLY responds to a header which EXACTLY matches \\'application/rdf+xml\\' (from diego.berrueta@fundacionctic.org on 2007-07-25)
5. RE: [Recipes] ISSUE-58: .htaccess 'accept header' ONLY responds to a header which EXACTLY matches \\'application/rdf+xml\\' (from A.J.Miles@rl.ac.uk on 2007-07-30)
6. RE: [Recipes] ISSUE-58: .htaccess 'accept header' ONLY responds to a header which EXACTLY matches \\'application/rdf+xml\\' (from diego.berrueta@fundacionctic.org on 2007-08-06)
7. Re: Agenda - Sep 04 2007 SWD telecon - 1500 UTC (from diego.berrueta@fundacionctic.org on 2007-09-04)
8. [ALL] Discussion of 'Recipes' in Amsterdam (from baker@sub.uni-goettingen.de on 2007-09-04)
9. [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from baker@sub.uni-goettingen.de on 2007-09-15)
10. [RECIPES] Amsterdam topic 'Recipes' (from baker@sub.uni-goettingen.de on 2007-09-15)
11. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from qreul@csd.abdn.ac.uk on 2007-09-18)
12. Re: [ALL] Agenda - Sep 18 2007 SWD telecon - 1500 UTC (from aisaac@few.vu.nl on 2007-09-18)

Related notes:

No additional notes.

The last email related to this...
From: Miles, AJ \(Alistair\) <A.J.Miles@rl.ac.uk>
Date: Mon, 30 Jul 2007 16:52:13 +0100
Message-ID: <677CE4DD24B12C4B9FA138534E29FB1D030E460A@exchange11.fed.cclrc.ac.uk>
To: "Diego Berrueta" <diego.berrueta@fundacionctic.org>, "SWD WG" <public-swd-wg@w3.org>

Hi Diego,

I remembered, there is a relatively simple configuration which allows q-values to be handl

The problem with this configuration is that the HTML and RDF files must reside in the same

Cheers,

Al.

--
Alistair Miles
Research Associate
Science and Technology Facilities Council
Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Didcot
Oxfordshire OX11 0QX
United Kingdom
Web: http://purl.org/net/aliman

Email: a.j.miles@rl.ac.uk
Tel: +44 (0)1235 445440

> -----Original Message-----
> From: public-swd-wg-request@w3.org
> [mailto:public-swd-wg-request@w3.org] On Behalf Of Diego Berrueta
> Sent: 25 July 2007 11:37
> To: SWD WG
> Subject: Re: [Recipes] ISSUE-58: .htaccess 'accept header'
> ONLY responds to a header which EXACTLY matches
> \"application/rdf+xml\"
>
>
> I've run some additional tests. Find results below:
>
> (Case e) [same as c&d, but without the q-values]
> Accept: application/rdf+xml,text/html
> Expected Content-Type: application/rdf+xml
> Actual Content-Type: text/html
> Result: FAILURE
>
> (Case f)
> Accept: text/html,application/rdf+xml
> Expected Content-Type: text/html
> Actual Content-Type: text/html
> Result: Success
>
> Some additional conclusions:
>
> * Like cases (c) and (d), swapping the order of the rules in
> .htaccess also inverts the results, i.e., (e) works fine, but
> (f) fails.
>
> * I suspect the regular expressions can be refitted in order
> to make cases (e) and (f) succeed at the same time. I have to
> work more on this and, hopefully, I'll propose the necessary
> changes in the R.E.
>
> * However, cases (c) and (d) are much more challenging due to
> the presence of the q-values, and I still can't imagine how
> to fix them using regular expressions.
>
> * There are also other challenges, like MIME type masks with
> asterisks.
> For instance, a request containg the following Accept header
> is valid and should return HTML (text/html), but of course,

> it isn't matched by our regular expressions:
>
> Accept: text/*
>
> Regards,
>
> El miÃ©, 18-07-2007 a las 16:16 +0200, Diego Berrueta escribiÃ³:
> > I made some testing on this issue using Vapour [1]. These are the
> > relevant results for Recipe 3 (Recipes 4 and 5 are similar). In all
> > the cases, a GET request was made for the
> [vocabulary|class|property]
> > URI, using a number of different values for the Accept header. I
> > compared the expected Content-Type against the actual Content-Type
> > returned by the
> > server:
> >
> > (Case a)
> > Accept: application/rdf+xml;q=0.5
> > Expected Content-Type: application/rdf+xml
> > Actual Content-Type: application/rdf+xml
> > Result: Success
> >
> > (Case b)
> > Accept: text/html;q=0.5
> > Expected Content-Type: text/html
> > Actual Content-Type: text/html
> > Result: Success
> >
> > (Case c)
> > Accept: application/rdf+xml;q=0.3,text/html;q=.5
> > Expected Content-Type: text/html (due to higher "q"-value)
> > Actual Content-Type: text/html
> > Result: Success
> >
> > (Case d)
> > Accept: application/rdf+xml;q=0.5,text/html;q=.3
> > Expected Content-Type: application/rdf+xml (due to higher
> "q"-value)
> > Actual Content-Type: text/html
> > Result: FAILURE

> >
> > My conclusions:
> >
> > * Test cases (a) and (b) work fine because the regular expressions
> > don't have begin-of-line and end-of-line delimiters (i.e.:
> symbols ^
> > and $ respectively). Therefore, the additional ";q=0.5"
> substring at
> > the end of the Accept header is silently skipped.
> >
> > * Test case (c) works fine because of the order of the RewriteRule
> > sentences in the .htaccess file. The first rule (the one
> that matches
> > any Accept header that contains 'text/html') has precedence, so the
> > appropiate redirection is returned.
> >
> > * Unfortunately, test case (d) fails for the same reason. The first
> > rule produces an undesired match and shadows the second one. The
> > values of the "q" parameters are opaque to the regular expression.
> >
> > * If we swap the order of the RewriteRules in the
> .htaccess, then (d)
> > will succeed, but (c) will fail.
> >
> > * As TimBL said, there is a serious problem, because content
> > negotiation is not working properly. We fail to deliver the

```
> preferred
> > content-type even if it is actually available! (see case (d)).
> >
> > * It seems to me that this issue cannot be easily solved with the
> > current regex approach, due to the inability of the regex
> to compare
> > numerical values (AFAIK). In practical terms, the list of
> MIME types
> > cannot be ordered by their "q"-value.
> >
> > I hope I'm not missing anything. Creative ideas on how to fix this
> > issue would be greatly appreciated!
> >
> > [1] http://vapour.sourceforge.net/

> >
>
> --
> Diego Berrueta
> R&D Department  -  CTIC Foundation
> E-mail: diego.berrueta@fundacionctic.org
> Phone: +34 984 29 12 12
> Parque CientÃfico TecnolÃ³gico GijÃ³n-Asturias-Spain
> www.fundacionctic.org
>
>
```

Received on Monday, 30 July 2007 15:52:35 GMT

ISSUE-60

Guidelines needed for proper construction of vocabulary scheme and 'term' URIs

State:
RAISED
Product:
Recipes
Raised by:
Jon Phipps
Opened on:
2007-09-18
Description:

```
The recipes don't spend very much time discussing best practices for URI
construction for vocabulary schemes and 'terms'.

Based on a request by the International Press Telecommunications Council:
http://lists.w3.org/Archives/Public/public-swd-wg/2007Sep/0023.html

They are seeking guidance wrt construction of URIs to identify taxonomy schemes
and "terms". They have two questions:

1.  Should we opt for "#" or "#_" or "/" or "?" or "?<foo>=" or
    some other string as the scheme URI terminator?

2.  What mechanism should we adopt for constructing code URIs?
```

(By "Code URIs" they mean a mapping of URI to existing term codes: "...we decided to continue using the large number of existing codes that are used in News and related industries today.")

Below is the full text of the email that raised the questions:

Introduction

Below is a draft statement of matters on which the International Press Telecommunications Council [1] seeks help from the W3C and from the broader Semantic Web community.

The statement hasn't yet been reviewed by the relevant IPTC groups, but to save time I'm sending it in draft form. We would very much like to have these matters resolved by the time of the next IPTC meeting on 15-17 October 2007, in Prague.

Background

The IPTC decided a few years ago that its new G2 family of News Exchange standards must be compatible with the Semantic Web. We decided that:

1. Terms from taxonomies used for News would be associated with individual URIs.

2. We would encourage the use of GRDDL to convert News marked up with metadata into forms understood by SemWeb tools.

At the same time we decided to continue using the large number of existing codes that are used in News and related industries today.

To reconcile these two requirements (SemWeb plus existing codes), we chose an approach somewhat similar to QNAMEs, though with several significant differences. The approach is:

- Codes exist within (coding) schemes. Familiar examples are:
  ISO 4217 alpha codes
  ISO 4217 numeric codes
  ISO 3166-1 two-letter alpha codes
  ISO 3166-1 three-letter alpha codes
  ISO 3166-1 numeric alpha codes
  IETF BCP 47 language tags

  Possibly less familar examples are:
  CUSIPs (eg "037833100", Apple Computer)
  ISBNs(eg "0-321-18578-1", The Unicode Standard, Version 4.0)
  ISSNs (eg "0261-3077", The Guardian)
  SEDOLs (eg "0263494", BAE Systems)
  Valorens (eg "1203203", UBS)

- Each coding scheme is associated with a URI. That URI *must* resolve to a resource (or resources) containing information about the scheme.

- Each scheme URI is locally mapped to a prefix.

- There are almost no constraints on the values of codes. For example, a code may start with a digit.

- A qualified code (QCODE) is expressed in the form:
  prefix:code

- We shall define rules for how scheme URIs should be terminated. These rules may take the form of guidelines.

- We shall define rules for the construction of a code URI from the corresponding scheme URI and the code. These rules may or may not specify simple concatenation.

- In the case of schemes controlled by the News industry, each code URI *must* resolve to a resource or (content negotiated) resources containing information about the code.

- In the case of schemes used but not not controlled by the News industry, each code URI *should* resolve to a resource or (content negotiated) resources containing information about the code.

Matters we need help with
---

1. Should we opt for "#" or "#_" or "/" or "?" or "?<foo>=" or some other string as the scheme URI terminator?

2. What mechanism should we adopt for constructing code URIs?

   Simple concatenation would work for (made up) URIs such as:
   www.iptc.org/taxonomies/subjects#_
   www.iptc.org/taxonomies/subjects/
   www.iptc.org/taxonomies/subjects?
   www.iptc.org/taxonomies/subjects?code=

   It would not work for:
   www.iptc.org/taxonomies/subjects#
   as the resulting URI would not be legal for HTML if the code started with a digit.

   The alternative is to inject some buffer string during the construction of the code URI. This would probably have to be a fixed string for all News taxonomies, as the alternative of retrieving (from the scheme URI?) per-scheme rules seems too burdensome for the recipient.

   Such a string could be, eg "_", so allowing a scheme URI such as:
   www.iptc.org/taxonomies/subjects#
   and a code URI such as:
   www.iptc.org/taxonomies/subjects#_12345678

   Alternatively, such a string could be, eg "#_", so allowing a scheme URI such as:
   www.iptc.org/taxonomies/subjects
   and a code URI such as:
   www.iptc.org/taxonomies/subjects#_12345678

   The disadvantage of both approaches is that such a rule would make it difficult for people to use scheme URIs such as:
   www.iptc.org/taxonomies/subjects/
   www.iptc.org/taxonomies/subjects?
   www.iptc.org/taxonomies/subjects?code=

3. We would very much appreciate help in developing a GRDDL script for our G2 standards. Nearly two years ago we developed a script to convert NewsML-G2 to RDF triples (N-Triples). We were not, however, able to figure out how to handle statements about statements. Note that for each piece of descriptive metadata we support attributes such as:
   creator

```
      date modified
      confidence
      relevance
      why present

   Thus one can, losely speaking, express:

      On 7 September 2007, Reuters stated that this News item
      has a subject of:
      -  George W. Bush (with 60% confidence)
      -  George H. W. Bush (with 40% confidence)

   We appreciate that the best way to handle statements about
   statements may still be unresolved within the SemWeb
   community.

4. We request that the W3C and the broader Semantic Web
   community take our requirements into consideration in the
   development of new specifications and tools, and in the
   enhancement of existing ones.  We are aware that some of
   these assume particular URI formats, eg the presence of a "#"
   as a separator or the absence of a digit after such a "#".

[1] http://www.iptc.org/

Thank you

Misha Wolf
News Standards Manager
Reuters
```

Related emails:

1. ISSUE-60: Guidelines needed for proper construction of vocabulary scheme and \\'term\\' URIs (from dean+cgi@w3.org on 2007-09-18)

Related notes:

No additional notes.

Recipe 6 implementation

Following ISSUE-17, this page is a discussion on the implementation of Recipe 6 of the BestPracticeRecipes for publishing RDF vocabularies.

NOTE:
This page is intended to merely support the production of the Recipe 6 and as such do not even constitute a draft document. While it can be cited, it is highly volatile and should not be considered final in any way.

Requirements

From the working draft:

1. Extended configuration (i.e.: both HTML and RDF versions, content negotiation)
2. Slash namespace, for instance:
    - http://example.com/example6/ (vocabulary)
    - http://example.com/example6/ClassA (class)
    - http://example.com/example6/propA (property)
3. Multiple hyperlinked HTML documents
4. RDF content being made available via some sort of query service such that clients can obtain a partial RDF description of the vocabulary as appropriate

Requirements 1-3 are already present in Recipe 5. Only requirement 4 is actually new. However, it seems an unlikely scenario to use dynamically-generated RDF fragments and static HTML at the same time. Therefore, the partial HTML documents should be also available via (the same?) query service.

Implementation alternatives

1. Script in the server-side. Common server-side script languages (PHP, Python, Perl, Ruby) have RDF APIs and bindings with RDF stores, and therefore are suitable to write a simple script that queries an RDF file or RDF store and returns the relevant portion).
    - Pros: ease of deployment (many web hosting servers have support for one of these languages)
    - Cons: webmasters are expected to write a (probably ad-hoc) script
    - Sample implementation (see below).
2. Java Servlet (or equivalent).
    - Pros: Java's fairly good support for RDF, SPARQL and RDF stores
    - Cons: heavyweight solution, difficult to deploy (requires a servlet container)
    - Sample implementation, valid for alternatives 1 and 2 (Apache rewrite rules, beware of line wraps). The DBPL server published by Free University of Berlin is used in the example as content provider:

```
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml
RewriteRule ^example6/(.+)
http://www4.wiwiss.fu-berlin.de/dblp/page/person/$1
[R=303]

RewriteCond %{HTTP_ACCEPT} application/rdf\+xml
RewriteRule ^example6/(.+)
http://www4.wiwiss.fu-berlin.de/dblp/data/person/$1
[R=303]

RewriteRule ^example6/(.+)
http://www4.wiwiss.fu-berlin.de/dblp/data/person/$1
[R=303]
```

The rules are straightforward. Requests for HTML data are forwarded to the URL of the HTML version exported by the D2R servlet; requests for RDF data (or without Acccept: header) are forwarded to the URL of the RDF data exported by the D2R servlet.

3. HTTP redirection to a SPARQL endpoint with HTTP bindings
   - Pros: lightweight, requires no programming
   - Cons: a SPARQL endpoint for the vocabulary must be available somewhere
   - Sample implementation (Apache rewrite rule, beware of line wraps):

```
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml
RewriteRule ^example6/(.+)
http://www4.wiwiss.fu-berlin.de/dblp/page/person/$1
[R=303]

RewriteCond %{HTTP_ACCEPT} application/rdf\+xml
RewriteRule ^example6/(.+)
http://www4.wiwiss.fu-berlin.de/dblp/sparql?query=DESCRIBE+<http://
[R=303]

RewriteRule ^example6/(.+)
http://www4.wiwiss.fu-berlin.de/dblp/sparql?query=DESCRIBE+<http://
[R=303]
```

This is similar to the previous one, but RDF requests are handled differently. A request such as http://example.com/example6/100007 is redirected to the result of executing a *DESCRIBE* < *http://www4.wiwiss.fu-berlin.de/dblp/data/person/100007* > sentence against the D2R SPARQL endpoint, which is an RDF description of !TimBL in DBLP.

   - Use case A: Joshua Tauberer announced he exposed a large RDF dataset from the US Census. There is an SPARQL endpoint available, and the URIs are dereferencable by means of URL rewriting (see, for instance, http://www.rdfabout.com/rdf/usgov/geo/us).
   - Use case B: D2R can publish large datasets and it uses redirects to SPARQL queries. More information in section 5 of 'Cool URIs for the Semantic Web'.

SKOS (Simple Knowledge Organisation System) Semantics

Abstract

This document specifies a precise formal semantics for the Simple Knowledge Organisation System (SKOS).

This document also specifies syntactic conditions which limit the use of the SKOS Vocabulary in RDF graphs.

See also SKOS/Primer.

Status of this Document

This is a wiki draft, and is a work in progress. The content of this document may change without warning at any time. This document has no official status whatsoever within any W3C process.

Table of Contents

Contents

Section: SKOS/Semantics/Introduction

This section last edited on 2007-10-04 12:16:22 is part of the SKOS/Semantics wiki draft.

Introduction

@@TODO general introduction

Basis for the Semantics

In this document, semantic conditions on the interpretation of the SKOS Vocabulary may be given in one of two forms:

- Wherever possible, semantic conditions are stated as a set of "axiomatic" RDF triples, using the RDF, RDFS and OWL Vocabularies.
- Where this isn't possible, semantic conditions are stated using the mathematical definitions and conventions established in section 5 of [OWL Semantics], where I = < R_I, P_I, EXT_I, S_I, L_I, LV_I, > is an an interpretation of some vocabulary V with respect to a datatype map D.

This formulation of the SKOS semantics has been designed to allow applications to choose whether to view SKOS as a vocabulary extension of either RDF, RDFS or OWL Full.

Note that some aspects of the semantics of SKOS, which are deemed to be useful under certain circumstances, cannot be stated without violating the syntactic constraints imposed by the OWL DL language. This has been raised as an issue for the SWDWG, which has yet to be resolved.

As a temporary solution, this document marks a number of axiomatic triples as *[OWL-Full]*. Applications MAY then choose to view SKOS as a vocabulary extension of OWL DL, by ignoring any triples marked as such.

Guidance on how to use the SKOS Vocabulary within the constraints of the OWL DL language will be given in the SKOS/Primer.

URI Abbreviations

Throughout this document, URIs are presented in an abbreviated form, using the following table of abbreviations:

URI Abbreviation Table	
`skos:`	`http://www.w3.org/2004/02/skos/core#`
@@TODO	...

Structure of this Document

The main body of this document is divided into a number of different "modules". This is entirely for the convenience of the reader, and no formal meaning is intended by the word "module".

The structure of each module is decribed below.

First, URIs from the SKOS Vocabulary are given, to which subsequent statements within the module apply. These are given as a list of abbreviated URIs, using abbreviations set out in section @@TODO. For example:

Vocabulary
`ex:foo` `ex:bar`

Second, semantic conditions on the interpretation of these elements are stated. As described in section @@TODO above, semantic conditions are either stated as a set of RDF triples using the same syntax as is used in [RDF Semantics] (i.e. N-Triples with URI abbreviations allowed), for example:

Axiomatic Triples

`ex:foo rdfs:range rdfs:Literal.`
`ex:bar rdfs:range rdfs:Literal.`
`ex:foo rdfs:subPropertyOf rdfs:label.` *[OWL-Full]*
`ex:bar rdfs:subPropertyOf rdfs:label.` *[OWL-Full]*

or as formal statements following the conventions and definitions established in section 5 of [OWL Semantics], for example:

Semantic Conditions
If <x,y> is in $EXT_I(S_I($`ex:foo`$))$ then there exists some z such that <x,z> is in $EXT_I(S_I($`ex:bar`$))$.

Third, informative examples may be given using the [Turtle] RDF syntax, which are consistent with the semantics, for example:

```
ex:a ex:foo "bar"@en.
```

Fourth, if applicable, informative examples may be given using the [Turtle] RDF syntax, which are **not** consistent with the semantics (i.e. give rise to a logical contradiction).

Fifth, any syntactic conditions which apply to the use of the vocabulary in RDF graphs are stated, using keywords MUST, SHOULD, MAY etc. as defined by [RFC 2199], for example:

Syntactic Conditions
An application MAY ignore any triple in an RDF graph where the predicate is either `ex:foo` or `ex:bar` and the object is NOT a plain literal.

Finally, any new entailment rules are stated, which follow from the semantic conditions given in the module. These rules are informative only, and are presented using the same conventions as used in section @@TODO of [RDF Semantics]. For example:

Entailment Rules (Informative)	
If E contains	**then add**

uuu `ex:foo` xxx .	uuu `ex:bar` _:nnn . where _:nnn identifies a new blank node in the graph.

Section: SKOS/Semantics/Labelling

This section last edited on 2007-10-04 12:16:22 is part of the SKOS/Semantics wiki draft. The following open issues are directly associated with this module: ISSUE-31. The following open issues may have a bearing on this module: ISSUE-26, ISSUE-27.

Module: Labelling

This module defines a semantics for the basic labelling properties available in SKOS.

Vocabulary
`skos:prefLabel` `skos:altLabel` `skos:hiddenLabel`

The triples below define semantic conditions on the interpretation of this vocabulary.

Axiomatic Triples

`skos:prefLabel rdf:type owl:AnnotationProperty.`
`skos:altLabel rdf:type owl:AnnotationProperty.`
`skos:hiddenLabel rdf:type owl:AnnotationProperty.`
`skos:prefLabel rdfs:range rdfs:Literal.` *[OWL-Full]*
`skos:altLabel rdfs:range rdfs:Literal.` *[OWL-Full]*
`skos:hiddenLabel rdfs:range rdfs:Literal.` *[OWL-Full]*
`skos:prefLabel rdfs:subPropertyOf rdfs:label.` *[OWL-Full]*
`skos:altLabel rdfs:subPropertyOf rdfs:label.` *[OWL-Full]*
`skos:hiddenLabel rdfs:subPropertyOf rdfs:label.` *[OWL-Full]*

There are no further semantic conditions on the interpretation of this vocabulary.

Intuitively, it doesn't make sense for a resource to have more than one "preferred" label in any given language, or for a label to be both "preferred" and "alternative", or for a label to be both "alternative" and "hidden", or for a label to be both "preferred" and "hidden". Note however that these intuitions are not handled within the semantics, but are handled instead at a syntactic level (see the syntactic conditions given below), which is perhaps easier to implement and allows error-tolerant strategies to be defined.

This means that, under the semantics given above, the examples below are logically consistent, even though they are intuitively inconsistent.

```
ex:a skos:prefLabel "foo"@en.
ex:a skos:prefLabel "bar"@en.
```

```
ex:a skos:prefLabel "foo"@en.
```

```
ex:a skos:altLabel "foo"@en.
```

```
ex:a skos:altLabel "foo"@en.
ex:a skos:hiddenLabel "foo"@en.
```

```
ex:a skos:prefLabel "foo"@en.
ex:a skos:hiddenLabel "foo"@en.
```

Syntactic conditions on the use of this vocabulary are given below.

Syntactic Conditions
An application MAY ignore any triple in an RDF graph where the predicate is either `skos:prefLabel`, `skos:altLabel` or `skos:hiddenLabel` and the object is NOT a plain literal.
Where an RDF graph contains more than one triple where the subject is the same, the predicate is `skos:prefLabel`, and the object is a plain literal with the same language tag, an application SHOULD ignore all but one of these triples.
An application SHOULD ignore a triple in an RDF graph where the predicate is `skos:hiddenLabel` and the graph contains another triple with the same subject and object, but with `skos:altLabel` or `skos:prefLabel` as predicate.
An application SHOULD ignore a triple in an RDF graph where the predicate is `skos:altLabel` and the graph contains another triple with the same subject and object, but with `skos:prefLabel` as predicate.

The first syntactic condition means that applications are only obliged to handle plain literals in the object position where these predicates are used. For example, an application MAY ignore the following triples:

```
ex:a skos:prefLabel ex:b.
ex:c skos:altLabel _:xxx.
```

The second syntactic condition specifies the recommended way for an application to handle more than one preferred label in any given language. For example, an application SHOULD ignore all but one of the following triples:

```
ex:a skos:prefLabel "foo"@en.
ex:a skos:prefLabel "bar"@en.
ex:a skos:prefLabel "baz"@en.
```

The third and fourth syntactic conditions specify the recommended way for an application to handle "clashes" between `skos:prefLabel`, `skos:altLabel` and `skos:hiddenLabel`. For example, an application SHOULD ignore the second triple in each of the three graphs below.

```
ex:a skos:prefLabel "foo"@en.
ex:a skos:altLabel "foo"@en.
```

```
ex:a skos:prefLabel "foo"@en.
ex:a skos:hiddenLabel "foo"@en.
```

```
ex:a skos:altLabel "foo"@en.
ex:a skos:hiddenLabel "foo"@en.
```

These syntactic conditions are intended to provide applications with an error-tolerant way of handling intuitively inconsistent data, which may easily arise where graphs are being merged from multiple sources. Nevertheless, producers of SKOS data are strongly advised to ensure that the graphs they produce do not contain any triples which trigger these syntactic conditions.

Section: SKOS/Semantics/Grouping

This section last edited on 2007-10-04 12:16:22 is part of the SKOS/Semantics wiki draft.

Module: Grouping Constructs

This module defines the grouping constructs available in SKOS, also known as "collections".

See also SKOS/Primer/Grouping.

Vocabulary
`skos:Collection skos:OrderedCollection skos:member skos:memberList`

The triples below define semantic conditions on the interpretation of this vocabulary.

Axiomatic Triples

`skos:OrderedCollection rdfs:subClassOf skos:Collection.`
`skos:member rdf:type owl:ObjectProperty.`
`skos:member rdfs:domain skos:Collection.`
`skos:memberList rdf:type owl:FunctionalProperty.`
`skos:memberList rdfs:domain skos:OrderedCollection.`
`skos:memberList rdfs:range rdf:List.`
`skos:Collection owl:disjointWith skos:Concept.`
`skos:Collection owl:disjointWith skos:ConceptScheme.`

There is also a logical relationship between the `skos:memberList` and `skos:member` properties, which is stated below (where "sequence" is defined as in section 5 of [OWL Semantics]).

Semantic Conditions
If <x,y> is in $EXT_I(S_I($`skos:memberList`$))$ and y is a sequence of $y_1,...y_n$ then <x,y_1>...<x,y_n> is in $EXT_I(S_I($`skos:member`$))$.

I.e. informally, for ordered collections, every member given by the `skos:memberList` property can also be given by the `skos:member` property.

See SKOS/Primer/Grouping
for examples of consistent and inconsistent usage of this vocabulary.

Note in particular that the use of a `skos:Collection` in either the subject or object position of a triple involving `skos:broader`, `skos:narrower` or `skos:related` as predicate will result in an inconsistency. This is because `skos:Collection` is disjoint with `skos:Concept`, and because `skos:broader`, `skos:narrower` and `skos:related` are sub-properties of `skos:semanticRelation`, which has both domain and range `skos:Concept` (see also SKOS/Semantics/ParadigmaticRelations).

See SKOS/Primer/Grouping
for guidance on how to construct a systematic hierarchical display involving grouping constructs.

There are no syntactic conditions on the use of this vocabulary.

The following entailment rules follow from the logical relationship between the `skos:memberList` and `skos:member` properties stated above, where u v and x stand for any URI or blank node ID.

Entailment Rules (Informative)	
If E contains	**then add**
u `skos:memberList` v . v `rdf:first` x .	u `skos:member` x .
u `skos:memberList` v_1 . v_1 `rdf:rest` v_2 . ... v_{i-1} `rdf:rest` v_i . v_i `rdf:first` x .	u `skos:member` x .

SKOS/Semantics (last edited 2007-06-26 17:24:35 by AlistairMiles)

SKOS Use Cases and Requirements

W3C Working Draft 16 May 2007

This version:
http://www.w3.org/TR/2007/WD-skos-ucr-20070516/
Latest version:
http://www.w3.org/TR/skos-ucr
Previous version:
This is the first public Working Draft
Editors:
Antoine Isaac, Vrije Universiteit Amsterdam, aisaac@few.vu.nl
Jon Phipps, Cornell University, jphipps@madcreek.com
Daniel Rubin, Stanford Medical Informatics, dlrubin@stanford.edu

---

Abstract

Knowledge organisation systems, such as taxonomies, thesauri or subject heading lists, play a fundamental role in information structuring and access. The Semantic Web Deployment Working Group aims at providing a model for representing such vocabularies on the Semantic Web: SKOS (Simple Knowledge Organisation System).

This document presents the preparatory work for a future version of SKOS. It lists representative use cases, which were obtained after a dedicated questionnaire was sent to a wide audience. It also features a set of fundamental or secondary requirements derived from these use cases, that will be used to guide the design of SKOS.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document is the first public Working Draft of the "SKOS (Simple Knowledge Organization System) Use Cases and Requirements", developed by the W3C Semantic

Web Deployment Working Group [SWD]. The SWD Working Group is chartered to advance the November 2005 SKOS Core Vocabulary Specification Working Draft and the SKOS Core Guide Working Draft to W3C Recommendation.

The Use Cases
detailed in this document have been selected as representative of the use cases submitted in response to a "Call for Use Cases" published in December 2006. These use cases as well as Issues identified by the working group have resulted in draft Requirements
that will guide the design of the future SKOS Recommendaton. Early feedback is therefore most useful. Feedback on use cases that can help to resolve open issues is especially important. Note also that any feature listed under Candidate Requirements should be considered as "at risk" without further feedback.

Comments on this Working Draft are encouraged and may be sent to public-swd-wg@w3.org; please include the text "[SKOS] UCR comment" in the subject line. All messages received at this address are viewable in a public archive. Commentors may wish to review the list of open issues before generating a new comment.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

---

1 Introduction

Knowledge organisation systems play a fundamental role in information structuring and access, e.g. for asset description or web site organisation. Such vocabularies, coming in the form of thesauri, classification schemes, subject heading lists, taxonomies or even folksonomies, are developed and used worldwide, by institutions as well as individuals. However these very important knowledge resources are still mostly isolated from the outside world, and not widely used in implementing systems.

The development of new information technologies and infrastructures, such as the World Wide Web, calls for new ways to create, manage, publish and use these knowledge organisation systems. It is especially expected that conceptual schemes will benefit from greater shareability, e.g. by being published via web services. In the meantime, the documentary systems which use them will turn to advanced information retrieval techniques to construct most of their semantic structure and lexical content.

SKOS (Simple Knowledge Organisation System) *[SWBP-SKOS-CORE-GUIDE]* provides a model to represent and use vocabularies and ontologies in the framework of the Semantic Web. A first version has been produced by the Semantic Web Best Practices and Deployment working group *[SWBPD]*, and is already used in some research projects. The Semantic Web Deployment Working Group *[SWD]* has been chartered to continue this work, and to "produce guidelines and an RDF vocabulary (SKOS) for transforming an existing vocabulary representation into an RDF/OWL representation" *[SWD-Charter]*.

In order to delimit the scope and elicit the required features for SKOS, the SWD working group has issued a call for use cases, asking for descriptions of existing or planned SKOS applications, according to a specific questionnaire. Following the gathering of these use cases, the Working Group has elicited a number requirements for SKOS which are motivated by the previous work on SKOS, or by contributions received after the call for use cases.

This document gives an account of this process. First, section 2 presents summaries of selected contributions, and pointers to the complete set of cases which were sent to the Working Group. Second, section 3 lists the requirements the Working Group has elicited so far.

2 Use Cases

2.1 Use Case #1 — An integrated view to medieval illuminated manuscripts

(Contributed by Antoine Isaac.
Complete description available at http://www.w3.org/2006/07/SWD/wiki/EucManuscriptsDetailed and at http://www.w3.org/2006/07/SWD/wiki/EucIconclassDetailed)

The purpose of this application is to provide the user with access to two collections of illuminated manuscripts from the Dutch and French national libraries, *Medieval Illuminated Manuscripts* and *Mandragore* (accessible online at http://www.kb.nl/manuscripts and http://mandragore.bnf.fr). The descriptions of images from these two collections follow different metadata schemes, and contain values from different controlled vocabularies for subject indexing. The user should however be able to search for items from the two collections using his preferred point of view, either using vocabulary from collection 1 or vocabulary from collection 2.

The main feature of the application is collection browsing, which uses hierarchical links in vocabularies: if a concept matching a query has subconcepts, the documents indexed against these subconcepts should be returned. The application also uses mapping links between concepts from the two vocabularies. For example, if an equivalence link is found between a query concept from one vocabulary and another concept from the second one, documents indexed by this other concept shall also be included in the query results.

Requires: R-ConceptualRelations, R-IndexingRelationship

Additionally, the application enables search based on free text queries over the collection metadata: documents can be retrieved based on free-text querying of the different fields used to describe the documents (creator, place, subject, etc.). For subject indexing, if a text query matches the label of a controlled vocabulary concept, the documents indexed against this concept will be returned.

The two collections use respectively the Iconclass and Mandragore analysis vocabularies.

Iconclass (http://www.iconclass.nl) contains 28000 items used to describe the subjects of an image (persons, event, abstract ideas). Complete versions are available for English, German, French, Italian, and partial translations for Finnish and Norwegian.

Requires: R-MultilingualLexicalInformation

The main building blocks of Iconclass are *subjects*, used to describe the subjects of images. An Iconclass subject consists of a *notation* (an alphanumeric identifier used for annotation) and a *textual correlate*
(e.g. “25F9 mis-shapen animals; monsters”). Subjects are organized in hierarchical trees, as in the following extract:

```
2 Nature
    25 earth, world as celestial body

        25F animals

            25F(+) KEY

            25F1 groups of animals

            …
```

25F9 mis-shapen animals; monsters

25FF fabulous animals (sometimes wrongly called 'grotesques'); 'Mostri' (Ripa)

Subjects can have associative cross-reference links between them (*systematic references*) and are linked to *keywords* that are used to search for them in Iconclass tools. Keywords form a network of their own, featuring *see* links (from one non-preferred keyword, not attached to any subject, to a preferred one), *see also* links (between keywords that are semantically or iconographically related) and *translation* links (between keywords in different languages).

Requires: R-LabelRepresentation, R-RelationshipsBetweenLabels

Iconclass additionally provides *auxiliary* mechanisms for subject specialization at indexing time. These actually allow for collection-specific vocabulary extension:

- by specializing a conceptual "placeholder" into a named individual (*bracketed text*) : `11H(…) saints` can be specialized into `11H(VALENTINE)`, which does not exist in the standard Iconclass,
- or by combining an existing subject with special auxiliaries (e.g. *keys* and *structural digits*): `25F2 mammals` can be combined with `(+33) head of an animal`, resulting in `25F(+33)` which will index an image of a mammal's head. Or `11H(VALENTINE)2` can be synthesized from `11H(VALENTINE)` and `11H(...)2 early life of male saint` to index an image which specifically denotes the early days of St. Valentine.

Requires: R-ConceptSchemeExtension, R-SkosSpecialization, R-IndexingAndNonIndexingConcepts, R-ConceptCoordination

Maintenance of the vocabulary is done via manual editing of semi-structured source files. As a general rule, the standard version will only be changed in a conservative way, not modifying the existing subjects.

Mandragore contains 16000 subjects. 15800 are *descriptors*, which are used to describe the illuminations and form a flat list. Additional structure is given by 200 *abstract topic classes*
which form a hierarchy organizing the descriptors according to general domains, but cannot themselves be used to describe documents:

ZOOLOGIE

.zoologie (généralités)

.mollusques

.mammifères

cochon [mammifère ongulé]

girafe [mammifère ongulé]

A descriptor is specified by a French label (“cochon”, for pig), optional rejected forms (“porc”), an optional definition (“mamifère ongulé”, hoofed mammal) and a reference to one or more topic classes (“.mammifères”, mammals). A note can sometimes be found as a complementary definition.

To enable integrated browsing, elements from Mandragore and Iconclass vocabularies must be linked together using equivalence or specialization links as in the following:

`25F72 molluscs` (Iconclass) is equivalent to `mollusques` (Mandragore)
`25F711 insects` (Iconclass) is more specific than `autres invertébrés (vers,arachnides,insectes...)` ("other invertebrates (worms, arachnida, insects", Mandragore)
`11U4 Mary and John the Baptist together with (e.g. kneeling before) the judging Christ, 'Deesis' ~ Last Judgement` (Iconclass) is equivalent to the *combination* of subjects `s.marie`, `s.jean.baptiste`, `christ` and `jugement.dernier` (Mandragore)
`25F(+441) herd, group of animals` (Iconclass) is equivalent to `troupeau` (Mandragore)

Requires: R-ConceptualMappingLinks

2.2 Use Case #2 — Bio-zen ontology framework for representing scientific discourse in life science

(Contributed by Matthias Samwald, Medizinische Universität Wien.
Complete description available at http://www.w3.org/2006/07/SWD/wiki/EucBiozenDetailed)

Bio-zen (http://neuroscientific.net/index.php?id=43) allows the description of biological systems and the representation of scientific discourse on the web in a highly distributed manner. It is intended to be used by researchers and developers in the life sciences.

SKOS is used in bio-zen for the representation of many existing life sciences vocabularies, taxonomies and ontologies coming from the "Open Biomedical Ontologies" (OBO) collection (http://www.fruitfly.org/~cjm/obo-download/). The size of all converted taxonomies taken together is on the order of millions of concepts. Typical examples are the Gene Ontology or Medical Subject Headings (MeSH), an entry of which is displayed here:

id	MESH:A.01.047.025
name	abdominal_cavity

def	"The region in the abdomen extending from the thoracic DIAPHRAGM to the plane of the superior pelvic aperture (pelvic inlet). The abdominal cavity contains the PERITONEUM and abdominal VISCERA\, as well as the extraperitoneal space which includes the RETROPERITONEAL SPACE." [MESH:A.01.047.025]
synonym	abdominal_cavity
synonym	cavitas_abdominis
is_a	MESH:A.01.047 ! abdomen

To represent such vocabulary elements as well as other types of information, the existing SKOS model has been integrated into a single OWL ontology, together with the DOLCE foundational ontology and the Dublin Core metadata model. In the process, the SKOS model has been extended with special types of concepts, e.g. biozen:sequence-concept. To enable efficient reasoning with the available dataset, it is important to note that existing constructs have been made compatible with the OWL-DL language.

Requires: R-CompatibilityWithOWL-DL

The bio-zen framework will consist of several applications, especially Semantic Wikis. A Bio-zen ontology incorporates constructs to make statements about digital information resources, that is creating "concept tags". This concept-tagging is an important feature of bio-zen, because it eases the integration of information from different sources.

Requires: R-IndexingRelationship

2.3 Use Case #3 — Semantic search service across mapped multilingual thesauri in the agriculture domain

(Contributed by Margherita Sini and Johannes Keizer, Food and Agriculture Organization. Complete description available at http://www.w3.org/2006/07/SWD/wiki/EucAimsDetailed)

This application coming from the AIMS project (http://www.fao.org/aims) is a semantic search service that makes use of mapped agriculture thesauri. It allows users to search any available terminology in any of the languages in which the thesauri are provided and retrieve information from resources which may have been indexed by one of the mapped vocabularies. Typical functions are navigating resources, helping to build boolean searches via concept identification, or expanding given searches by extra languages or synonyms.

Requires: R-IndexingRelationship

The service builds on several agriculture vocabularies: the Agrovoc Thesaurus (http://www.fao.org/aims/ag_intro.htm), the Agris/Caris Classification Scheme (ASC), the FAO Technical Knowledge Classification Scheme (TKCS), the subjects from the FAOTERM vocabulary, etc.

Agrovoc contains 35000 *terms*
in 12 languages (not all of the languages feature the same translated terms, however), while ASC, TCKS and FAOTERM range between 100 and 200 categories available in the

5 official FAO languages. Agrovoc terms consist of one or more words and always represent a single concept. Terms are divided into *Descriptors* and *non-descriptors*, the first currently only used for indexing. For each descriptor, a word block is displayed showing the relation to other terms: BT (broader term), NT (narrower term), RT (related term), UF (non-descriptor). There are also scope notes, used to clarify the meaning of both descriptors and non-descriptors.

Term code	1939
Term label	EN : Cows, FR : Vache, ES : Vaca, AR : بقرات , ZH : ?? , PT : Vaca, CS : krávy, JA : ?? , TH : ?ม่โค , SK : kravy, DE : KUH
BT	Cattle (code 1391)
NT	Suckler cows, Dairy cows (26767, 36875)
RT	Heifers, Cow milk, Milk yielding animals, Females (3535, 4833, 15969, 16080)
SNR	Females (15969)
Scope Note	Use only for cattle and zebu cattle; for other species use "Females" (15969) plus the descriptor for the species

Requires: R-ConceptualRelations, R-LabelRepresentation, R-TextualDescriptionsForConcepts, R-MultilingualLexicalInformation

Actually, the AIMS project includes some more specific links, presented in http://www.fao.org/aims/cs_relationships.htm: Concept-to-Concept relationships (subclass of; caused by; member of; part of), Term-to-Term relationships (related term; synonym; translation) and String-to-String relationships (spelling variant; acronym).

Examples of such links are:

synonym	bucket	pail
abbreviation_of	Corp.	Corporation
acronym	Food and Agriculture Organization	FAO
spelling_variant	organisation	organization
translation	vache	cow
scientific_taxonomic_name	African violet	Saintpaulia

Requires: R-SkosSpecialization, R-RelationshipsBetweenLabels

Currently the Agrovoc management system lacks distributed maintenance, but it is expected that a new system will soon solve this problem, which is crucial since changes are made by experts from all over the world.

For AIMS, Agrovoc has been converted into SKOS (ftp://ftp.fao.org/gi/gil/gilws/aims/kos/agrovoc_formats/skos/2006) and is being mapped to two other vocabularies: the Chinese Agricultural Thesaurus (CAT) and the National

Agricultural Library thesaurus (NAL). This mapping uses links inspired by the SKOS mapping vocabulary *[SWBP-SKOS-MAPPING]*, as below:

CAT-ID	CAT-EN	Map	AG-ID	AG-EN	AG-ID	AG-EN
30854	Senta flammea	Exact	9748	Cheena		
50008	Mayetola destructor	Exact-OR	24260	Triticale (gramineae)	7949	Triticales (product)
1160	Two-shear sheep	NT1	3662	Hordeum vulgare		

Requires: R-ConceptualMappingLinks

2.4 Use Case #4 — Supporting product life cycle

(Contributed by Sean Barker, BAE Systems.
Complete description available at
http://www.w3.org/2006/07/SWD/wiki/EucProductLifeCycleSupportDetailed)

The problem of the Product Life Cycle Support (PLCS) application is to integrate a network of interconnected supply chains, with multiple, large customers buying a wide range of products (from shoes to aircraft) each dictating their own standards, and with every supplier being part of multiple supply chains. Each customer wants to maintain a common approach over all its supply chains. And each supplier wants to maintain the same system for each of the supply chains it works in.

The aim of this application is to propose a data exchange mechanism for managing the life support of complex products (http://www.oasis-open.org), including configuration definition, maintenance definition, maintenance planning and scheduling, and maintenance and usage recording (including configuration change).

For that, an upper ontology of several hundred items for the description of the product life cycle will be defined. There is no chance of the entire supply system (10,000's of businesses) developing a single detailed model. However, given the upper ontology, they will be free to specialize individual ontology terms (playing the role of place holders for local extension) to meet their precise needs.

PLCS is conceptually a co-operatively developed web in XML, with the live version being a set of runtime views assembled from files submitted by a dozen or so contributors. It may be useful, where ontologies diverge, to map terms between the diverging branches, either to indicate where terms can be harmonized to their equivalent, or to identify that there is a similarity link that is not exact equivalence.

Requires: R-ConceptualRelations, R-ConceptSchemeExtension, R-ConceptualMappingLinks

The PLCS vocabulary addresses hundreds of separate functions, including classification of items, classification of information usages (e.g. types of part identifier), classification of entity roles (e.g. date as start date) or classification of relationships (e.g. supersedes).

Typical examples of terms are:

Identification_code	An Identification_code is an identifier_type which is encoded according to some convention. Typically but not necessarily concatenated from parts each with a meaning. E.g. tag number, serial number, package number and document number.
Part_identification_code	A Part_indentfication_code is an Identification_code that identifies the types of parts. For example, a part number. CONSTRAINT: An Identification_assignment classified as a Part_identification_code can only be assigned to Part Organization_name
Owner_of	An Owner_of is an Organization_or_person_in_organization_assignment that is assigning a person or organization to something in the role of owner. For example, the owner of the car.

The vocabulary has been encoded using OWL, and is managed via the Protege OWL editor.

Requires: R-TextualDescriptionsForConcepts

2.5 Use Case #5 — CHOICE@CATCH ranking of candidate terms for description of radio and TV programs

(Contributed by Véronique Malaisé and Hennie Brugman, Vrije Universiteit Amsterdam and Max Planck Institute for Psycholinguistics.
Complete description available at http://www.w3.org/2006/07/SWD/wiki/EucRankingForDescriptionDetailed and at http://www.w3.org/2006/07/SWD/wiki/EucGtaaBrowser)

Radio and television programs at the Dutch national broadcasting archive (Sound and Vision) are typically associated with contextual text descriptions: web site texts, subtitles, program guide texts, texts from the production process, etc. These context documents are used by documentalists at Sound and Vision who manually describe programs using concepts from the GTAA thesaurus (Gemeenschappelijke Thesaurus Audiovisuele Archieven - Common Thesaurus for Audiovisual Archives).

The CHOICE project (part of the Dutch CATCH research program) uses natural language processing techniques to automatically extract candidate GTAA terms from the context documents. The application focused on in this section takes these candidate terms as input, and ranks them on the basis of the structure of the GTAA thesaurus. For example, the fact that "Voting" and "Democratization" are related in GTAA by a two-step path (via the "Election" term and two "related-to" links) will positively influence the ranking of these

terms. Ranked terms will be presented to documentalists to speed up their description work.

The GTAA vocabulary covers a wide range of topics, as it is meant to describe anything that can be broadcast on TV or radio. It contains approximately 160,000 terms, divided into 6 disjoint facets: Keywords, Locations, Person Names, Organization-Group-Other Names, Maker Names, and Genres.

The thesaurus mainly uses constructs from the ISO 2788 standard, like Broader Term, Narrower Term, Related Term and Scope Notes. Terms from all facets of the GTAA may have Related Terms, Use/Use For and Scope Notes, but only Keywords and Genres can also have Broader Term/Narrower Term relations, organizing them into a set of hierarchies. In addition to these standard features, Keywords terms are thematically classified in 88 subcategories of 16 top Categories.

Preferred Term	ambachten *(crafts)*
Related Terms	ondernemingen *(ventures)* , beroepen *(professions)*, artistieke beroepen *(artistic professions)*
Broader Term	beroepen *(professions)*
Narrower Terms	boekbinders *(bookbinders)*, bouwvakkers *(building workers)*, glasblazers *(glassblowers)*
Scope Note	niet voor afzonderlijke ambachten maar alleen als verzamelbegrip, bijv. voor (markten van) oude ambachten *(not for specific crafts, only in general meaning, e.g. (markets of) old crafts)*
Categories	05 economie (economy), 09 techniek (technique)

Requires: R-ConceptualRelations, R-LabelRepresentation, R-SkosSpecialization

The application, envisioned as a SOAP web service, uses a Sesame RDF web repository containing the SKOS version of the GTAA thesaurus to retrieve the 'term contexts' of the terms in the input list, which is stored in a local RDF repository.

This term context includes, for one given term, all terms that are directly connected to it by Broader Term, Narrower Term or Related Term relations. This includes pre-computed inter-facet links that are not part of the ISO standard, though allowed by the GTAA data model. For example, one can link a "King" in the Person facet to the general subject "Kings" and the country which this King rules.

For the ranking, it is now assumed that candidate terms that are mutually connected by thesaurus relations (directly or indirectly) are more likely to be good descriptions than isolated candidate terms. Later on, it might be interesting to differentiate between types of thesaurus relations, or to use more complex patterns of these relations.

The thesaurus-based recommendation system can also be integrated with a recommendation system that is based on co-occurences between terms that are used in previously existing descriptions of programs.

2.6 Use Case #6 — BIRNLex: a lexicon for neurosciences

(Contributed by William Bug, Drexel University College of Medicine.
Complete description available at http://www.w3.org/2006/07/SWD/wiki/EucBirnLexDetailed)

BIRNLex is an integrated ontology+lexicon used for various purposes — some end-user/interactive, others back-end/infrastructure — within the BIRN Project to support semantically-formal data annotation, semantic data integration, and semantically-driven, federated query resolution.

Requires: R-ConceptualMappingLinks, R-IndexingRelationship, R-LexicalMappingLinks

Below are examples of BIRNLex class definitions that illustrate the need for lexical support and links to external knowledge sources. The general design goals have been to use both the Dublin Core metadata elements and SKOS where ever possible. The goal is to use SKOS for all lexical qualities. There are certain annotation properties that should be shared across all biomedical knowledge resources. There are other required elements specific to the specific needs in BIRN (the group producing BIRNLex).

Class	Anterior_ascending_limb_of_lateral_sulcus
birn_annot:birnlexCurator	Bill Bug
birn_annot:birnlexExternalSource	NeuroNames
birn_annot:bonfireID	C0262186
birn_annot:curationStatus	raw import
birn_annot:neuronames	ID 49
birn_annot:UmlsCui	C0262186
obo_annot:createdDate	"2006-10-08"^^http://www.w3.org/2001/XMLSchema#dat
obo_annot:modifiedDate	"2006-10-08"^^http://www.w3.org/2001/XMLSchema#dat
skos:prefLabel	Anterior_ascending_limb_of_lateral_sulcus
skos:scopeNote	human-only

Class	Medium_spiny_neuron
birn_annot:birnlexCurator	Maryann Martone
birn_annot:birnlexDefinition	The main projection neuron found in caudate nucleus, putamen and nucleus accumbens...
birn_annot:bonfireID	BF_C000100
birn_annot:curationStatus	pending final vetting
dc:source	Maryann Martone
obo_annot:createdDate	"2006-07-15"^^http://www.w3.org/2001/XMLSchema#date
obo_annot:modifiedDate	"2006-09-28"^^http://www.w3.org/2001/XMLSchema#date

skos:prefLabel	Medium_spiny_neuron

Requires: R-CompatibilityWithDC, R-CompatibilityWithOWL-DL, R-ConceptualRelations, R-LabelRepresentation, R-ConceptSchemeExtension

The following is a subset of BIRNLex applications, either extant or in the offing:

- online curation: tools to enable curating the BIRNLex ontology+lexicon via the web;
- annotation: applications designed to support domain experts annotating neuroimaging data;
- query/mediation: a mediator program federates more than 60 data repositories. Site databases register with the mediator by mapping the relevant elements from their resident data model into the mediator's global model.

In all of these applications, it is critical to have a clear, distinct, and shared representation for the associated lexicon. For instance, when integrating BIRN segmented brain images with those from other projects across the net, use of lexical variants from a variety of public terminologies and thesauri such as SNOMED and MeSH can provide a powerful means to largely automate semantic integration of like entities - e.g., corresponding brain region, equivalent behavioral assays described using different preferred labels/names. In providing a community shared formalism for representing the associated lexicon, SKOS can greatly simplify this task. If, for instance, the lexical repository (collection of Lexical Unique Identifier, each lexical variant of a term getting one LUI) contained in UMLS were represented according to SKOS, this would provide an extremely valuable resource to the community of semantically-oriented bioinformatics researchers, as well as a powerful tool to support latent semantic analysis or natural language processing when linking to unstructured text.

The following are the collection of terminologies and ontologies being linked into BIRNLex: Neuronames, Brainmap.org classification schemes, RadLex, Gene Ontology, Reactome, OBI, PATO, Subcellular Anatomy Ontology (CCDB - http://ccdb.ucsd.edu/), MeSH.

Neuronames concerns brain anatomy and is about 750 classes and thousands of associated lexical variants. Brainmap.org classification includes hierarchies to describe neuroanatomy, subject variables, stimulus conditions, and experimental paradigms associated with functional MRI of the nervous system The Subcellular Anatomy Ontology is designed to describe the subcellular entities associated with ultrastructural and histological imaging of neural tissue. Currently the application is only dealing with English lexical entries.

BIRNLex curators are working with the National Center for Biomedical Ontology (NCBO) to adopt the OBO Foundry recommendations in the construction of BIRNLex. Use of SKOS elements can be useful, so that, for instance, software applications can draw on "skos:prefLabel", "obo_annot:synonym", "obo_annot:definition", etc.

The management of BIRNLex is currently done manually in Protege-OWL.

Requires: R-CompatibilityWithOWL-DL

However, the ultimate goal is to adopt a client-server infrastructure that will created an

RDF-based backend store and support both curation of the ontology and annotation using the ontology via Java Portlet-based applications. BIRN has a core infrastructure staff dedicated to use of the GridSphere Java Portlet implementation framework (www.gridsphere.org).

2.7 Use Case #7 — Radlex: a lexicon for radiology

(contributed by Curt Langlotz.
Complete description available at http://www.w3.org/2006/07/SWD/wiki/EucRadlexDetailed)

RadLex provides a structured vocabulary of terms used in the field of radiology. Currently completed are listings of anatomic terms and "findings", which includes things that can be seen on or inferred from images produced by radiologists. These two sets include a total of about 7500 terms. A list of the terms used to describe the creation of such images, including information about the equipment used and the various imaging sequences performed, will be complete by the end of 2007.

An example application demonstrating functionality is an image annotation program that reads in RadLex and provides users the ability to search for and use particular RadLex terms to associate with images, post-coordinating them if necessary. Users would want to be able to retrieve RadLex terms by name or synonym.

Requires: R-ConceptualRelations, R-LabelRepresentation, R-TextualDescriptionsForConcepts, R-ConceptCoordination

RadLex, which can be searched and browsed online at www.radlex.org, is a taxonomy currently built predominantly using is-a relations. But there are also part-of and other relations (especially for anatomy), and new relations will be added as RadLex expands. Each term has a rich set of metadata fields to include provenance information and terminological data such as synonyms, definition, and related terms from other vocabularies.

The practical fields include:

- Term name
- ID number (with no inherent semantics)
- parents, and their relation to the term
- children, and their relation to the term

and optionally, any

- mappings to other vocabularies
- definition
- synonyms
- source (a reference publication which includes this term)
- other comments (such as derivation of the term, special or preferred uses of it, etc.)

Requires: R-ConceptualRelations, R-AnnotationOnLabel, R-RelationshipsBetweenLabels, R-LexicalMappingLinks

The relationships used among terms include:

- Continuous with [DEF: Two structures are “continuous with” one another if they are immediately adjacent and physically connected to one another. This relationship is often used for cavitary and tubular structures, such as the continuity between the left ventricle and the aorta. This is a reflexive relation.]
- Branch of [DEF: a smaller conduit is a “branch of” a larger conduit if it is one of a group of two or more conduits that continue in the direction of flow where only one conduits had existed before. It is permissible for one of the continuing conduits to maintain the same name of original conduits. This relation is often used with arteries.]
- Branch [DEF: the converse relation to “branch of”]
- Tributary of [DEF: a smaller conduit is a “tributary of” a larger conduit if it is one of a group of two or more vessels that join to form a single conduit in the direction of flow. This relation is often used with veins.]
- Tributary [DEF: the converse relation to “tributary of”]
- Part of [DEF: one object is “part of” another object if it comprises less than all of the other object. This relation is often used with solid body parts and organs.]
- Segment of [DEF: one tubular structure is a “segment of” another if it defines a part of that structure divided perpendicular to the axis of the tube. This relation is often used to define the subparts of tubular structures, such as arteries, veins, and intestines.
- Part [DEF: the converse relation to “part of”]
- Segment [DEF: the converse relation to “segment of”]
- Contained_in [DEF: One structure is “contained in” another if the first structure is inside the other. For example the liver is contained in the abdominal cavity.]
- Contains [DEF: the converse relation to “contained in”]
- Is_a; Type of
- Member of [DEF: One structure is a “member of” a set of structures. For example, the “liver” is a member of the “set of viscera of abdomen”]
- Member [DEF: the converse relation to “member of”]
- Synonym [DEF: the term is a less-preferred synonym of a preferred term]

For instance, “nervous system” has a part called “brain”, and “nervous system” contains “nervous system spaces”. The view of the hierarchy itself does not reveal the relationships among the terms; this information is found within the term features, shown in this format on the right-hand side. In this framework, the hierarchy is generated from the different relationships among terms, using either SPARQL or a custom interface to an application that consumes the terminology.

There are 9 separate hierarchies in the vocabulary: Treatment; Image acquisition, Processing and Display; Modifier; Finding; Anatomic Location; Uncertainty (to be renamed Certainty); Teaching Attribute; Relationship; and Image Quality (as seen in the screenshots above). There are currently no relations holding between terms in different hierarchies, though this could be developed in future (e.g. linking of particular Findings to

potential Anatomic Locations).

The Radlex vocabulary is provided in English, with plans to include other languages (e.g., German).

Requires: R-MultilingualLexicalInformation

Protégé has been used to create a machine-readable version of the vocabulary, which is available at http://www.radlex.org/radlex/docs/downloads.html. RadLex will be available in OWL-DL in the future.

Requires: R-CompatibilityWithOWL-DL

During the design of the vocabulary, basic guidelines from Cimino and Chute were used, such as ensuring that a term only corresponds to one concept. As the terminology is being developed into a more structured form, with more types of relationships, different parents are being allowed as long as the relationship type is different. E.g. one IS-A parent, one PART-OF parent, etc.

Potential changes in the vocabulary are submitted to the chair of the RadLex Steering Committee of the Radiological Society of North America, who consults with the relevant lexicon development committee. Accepted changes are periodically incorporated into the vocabulary. The first release was made public in November 2006.

Currently, a mapping is being developed between RadLex and the corresponding terms/codes in SNOMED (Systematized Nomenclature of Medicine) and the ACR (American College of Radiology) Index, the vocabularies that were used as a starting point for terminology development.

From a representational point of view, this mapping shall consist of equivalence and specialization links. Later, we expect people to compose atomic terms (post-coordination) to describe composite entities.

Requires: R-ConceptCoordination

2.8 Use Case #8 — NSDL Metadata Registry

(Contributed by Jon Phipps, Cornell University.
Complete description available at http://www.w3.org/2006/07/SWD/wiki/RucMetadataRegistryExtended)

The NSDL Registry is intended to provide a complete vocabulary development and management environment for development of controlled vocabularies. Services are primarily directed at vocabulary owners and include provisions for:

- managing access and editing rights for groups of vocabulary maintainers maintaining individual vocabularies
- import and management of existing vocabularies, with and without existing URIs
- namespace management and maintenance services providing permanent URIs
- registered users to receive notifications of changes to vocabularies to which they

have subscribed

- content negotiation for retrieval of registered vocabularies in various formats, currently RDF/XML (rdf), XHTML (html), and XML Schema (xsd)
- content negotiation and resolution services for registered vocabularies in non-registry namespaces (in alpha)
- controlled concept editing and maintenance using SKOS properties
- controlled editing of reciprocal relationships between concepts (Requires: SKOS:status in order to manage reciprocal endorsement)
- controlled mapping of relationships between concepts in different vocabularies. See http://www.w3.org/2006/07/SWD/wiki/RucMetadataRegistryExtended. (Requires: R-ConceptualMappingLinks)
- concept-level change history management (Requires: URI content negotiation and http 301/302 redirection)
- vocabulary- and concept-level version management (in alpha)
- multilingual vocabulary maintenance (Requires: the ability to manage concept equivalence between vocabularies in different languages and intra-concept language-related equivalence between concept properties in multiple languages)
- skos validation by user input constraint and validation of imported vocabularies (Requires: R-ConsistencyChecking)
- search and browse for concepts by label

The registry currently has a number of vocabularies registered. A sample entry of a vocabulary/scheme and a single concept is below (taken from http://metadataregistry.org/uri/NSDLEdLvl.html).

Scheme	NSDLEdLvl
Name	NSDL Education Level Vocabulary
Owner	National Science Digital Library
Community	Science, Mathematics, Engineering, Technology
URL	http://metamanagement.comm.nsdl.org/cgi-bin/wiki.pl?VocabDevel

Concept	NSDLEdLvl/1023
Label	Middle School
Top Concept	No
Status	published
history note	Term source: http://www.ed.gov
has narrower	Grade 6
has narrower	Grade 7
has broader	Grades Pre-K to 12
alternative label	Junior High School

2.9 Other use cases

The SWD Working Group maintains on its wiki site the complete list of descriptions that were sent following its call for use cases:

- Geographical web service for hierarchical browsing (contributed by Walter Koch). Summed up description available at http://www.w3.org/2006/07/SWD/wiki/EucTgn.
- Representation of Tactical Situation Objects (contributed by Sean Barker). Summed up description available at http://www.w3.org/2006/07/SWD/wiki/EucTacticalSituationObject.
- GTAA Web Browser (contributed by Véronique Malaisé and Hennie Brugman). Summed up description available at http://www.w3.org/2006/07/SWD/wiki/EucGtaaBrowser.
- Ontology for Biomedical Investigation (OBI) for for describing methods in biomedical research (contributed by Trish Whetzel). Detailed description available at http://www.w3.org/2006/07/SWD/wiki/EucObiDetailed.
- The STAR:dust conceptual model (contributed by Irene Celino, Emanuele Della Valle and Francesco Corcoglioniti). Summed up description available at http://www.w3.org/2006/07/SWD/wiki/EucStarDust.
- UDC: the Universal Decimal Classification (contributed by Antoine Isaac and Aida Slavic). Description for motivating coordination requirement available at http://www.w3.org/2006/07/SWD/wiki/EucUDC.
- the RAMEAU subject heading language (contributed by Antoine Isaac). Description for motivating coordination issue available at http://www.w3.org/2006/07/SWD/wiki/EucRameau.
- XMDR: Extended Metadata Registry Prototype (contributed by John McCarthy). Detailed description available at http://www.w3.org/2006/07/SWD/wiki/EucXmdrDetailed.
- High-level Thesaurus (HILT) Project (contributed by George Macgregor). Detailed description available at http://www.w3.org/2006/07/SWD/wiki/EucHiltDetailed.
- A Semantic Grid Browser for the Life Sciences Applied to the study of Infectious Diseases (contributed by Simon Jupp). Detailed description available at http://www.w3.org/2006/07/SWD/wiki/EucSeaLifeDetailed.
- Semantic-based Framework for Personalized TV Content Management in a cross-media environment (contributed by Pieter Bellekens). Detailed description available at http://www.w3.org/2006/07/SWD/wiki/EucPersonalizedTvDetailed
- intraLibrary: Support for web-based Repository of Learning Objects (contributed by Sarah Currier).

Detailed description available at
http://www.w3.org/2006/07/SWD/wiki/EucIntraLibraryDetailed.

- Hybrid and Network-Assisted Vocabulary Interface (HANAVI) and Japan National Diet Library List of Subject Headings (NDLSH) (contributed by Mitsuharu Nagamori).
  Detailed description available at
  http://www.w3.org/2006/07/SWD/wiki/EucHanaviDetailed.
- Squiggle: an application framework for model-driven development of real-world Semantic Search Engines (contributed by Irene Celino).
  Detailed description available at
  http://www.w3.org/2006/07/SWD/wiki/EucSquiggleDetailed.
- Conceptual Open Hypermedia Service (COHSE) (contributed by Sean Bechhofer).

  Detailed description available at
  http://www.w3.org/2006/07/SWD/wiki/EucCohseDetailed.
- The Semantic Web Environmental Directory (SWED) (contributed by Alistair Miles).
  Detailed description available at
  http://www.w3.org/2006/07/SWD/wiki/EucSwedDetailed.

3 Requirements

The use cases presented in the previous section motivate a number of requirements that the SKOS specification must or should meet in order to fulfill its aim as a standard model for porting simple concept schemes on the semantic web. Depending on the level of consensus reached in the Working Group, these requirements are categorized into *accepted* and *candidate* requirements.

Note: in the following, to avoid ambiguities, *vocabulary* will be used to refer to the *SKOS vocabulary*, that is, the set of constructs (classes, properties) introduced in the SKOS model. *Concept Scheme*
will be used to refer to the objects built with SKOS, i.e. the application-specific collections of concepts that are mentioned in SKOS use cases.

@@ Some requirements are linked to issues that are still being examined by the Working Group, as found on the wiki site
http://www.w3.org/2006/07/SWD/wiki/SkosIssuesSandbox. @@

3.1 Accepted requirements

R-ConceptualRelations

Representation of relationships between concepts

The SKOS model shall provide semantic relationships between concepts, for display or search purposes. Typical examples are the hierarchical relations *broader than* (BT), *narrower than* (NT) and the non-hierarchical associative relation *related to* (RT).

Motivation: Tgn, Manuscripts, Aims, ProductLifeCycleSupport, RankingForDescription, etc.

R-ConceptSchemeExtension

Extension of concept schemes

A concept scheme might be locally extended with new concepts referring to existing ones, e.g. as specializations of these.

Motivation: Manuscripts, BirnLex, ProductLifeCycleSupport

R-ConceptualMappingLinks

Correspondence/Mapping links between concepts from different concept schemes

In order to build links between concepts coming from different concept schemes, SKOS should provide proper semantic relationships. Possible links, similarly to the ones found existing SKOS and SKOS mapping *[SWBP-SKOS-MAPPING]* vocabularies, include concept equivalence and specialization/generalization relations.

Motivation: Manuscripts, Aims, ProductLifeCycleSupport, BirnLex, MetadataRegistry

R-LabelRepresentation

Representation of basic lexical values (labels) associated to concepts

The SKOS model shall provide means to represent the labels (preferred or not) of a concept, for display or search purposes.

Motivation: Tgn, Manuscripts, Aims, RankingForDescription, etc.

R-MultilingualLexicalInformation

Representation of lexical information in multiple natural languages

The lexical information specified in concept schemes (labels, but also definitions and notes) could come in different natural languages. A typical example is the case of a multilingual concept scheme with concepts having labels translated in several languages.

Motivation: Manuscripts , Aims, RadLex

R-SkosSpecialization

Local specialization of SKOS vocabulary

For particular situations, the designer of a SKOS concept scheme should be able to introduce new model-level classes and properties, and link them to existing SKOS constructs. Possible cases include the creation of specific kinds of textual definitions or notes for concepts, or the specification of new types of concepts.

Motivation: Manuscripts, Tgn, Aims, Biozen, RankingForDescription

@@ Linked to SKOS-I-extension-6, SKOS-I-SpecializationOfRelationships @@

R-TextualDescriptionsForConcepts

Representation of textual descriptions attached to concepts

The SKOS model shall provide means to represent descriptive notes that could help understanding the elements of concept schemes, e.g. scope notes explaining the way concepts are used to describe documents.

Motivation: Aims, ProductLifeCycleSupport, TacticalSituationObject, BirnLexDetailed, etc.

3.2 Candidate requirements

R-AnnotationOnLabel

Ability to represent annotations on lexical items

Labels, which are currently modeled as literals in SKOS, as well as possibly other literals, are valid subjects of discourse when modeling concept schemes, e.g. when recording the dates during which a particular label was in common use. However, in RDF only resources may be subjects of statements, and literals may only be objects of statements. The question then arises, how are we to annotate labels and other literals, that is to relate them as subjects, to other entities.

Motivation: RadLex

@@ Linked to SKOS-I-AnnotationOnLabel @@

R-CompatibilityWithDC

Compatibility between SKOS and Dublin Core Abstract Model

Using SKOS model shall be compatible with using Dublin Core Abstract Model *[DCAM]*. When there are links between SKOS features and Dublin Core ones, these shall be specified.

Motivation: BirnLex

@@ Linked to SKOS-I-CompatibilityWithDC @@

R-CompatibilityWithISO11179

Compatibility between SKOS and ISO11179[Part 3]

SKOS model shall be compatible with part 3 of ISO 11179 specifications *[ISO11179-3]*.

@@ Linked to SKOS-I-CompatibilityWithISO11179 @@

R-CompatibilityWithISO2788

Compatibility between SKOS and ISO2788

SKOS model shall be compatible with ISO 2788 specifications *[ISO2788]*.

@@ Linked to SKOS-I-CompatibilityWithISO2788 @@

R-CompatibilityWithISO5964

Compatibility between SKOS and ISO5964

SKOS model shall be compatible with ISO 5964 specifications *[ISO5964]*.

@@ Linked to SKOS-I-CompatibilityWithISO5964 @@

R-CompatibilityWithOWL-DL

OWL-DL compatibility

SKOS should provide a legal OWL-DL ontology, to be compatible with most common editors and reasoners.

Motivation: Biozen, BirnLex, RadLex

@@ Linked to SKOS-I-owlImport-7, SKOS-I-Semantics-10 @@

R-ConceptCoordination

Coordination of concepts

SKOS should provide the ability to create new concepts from existing ones, e.g. by using special qualifiers that add a shade of meaning to a normal concept.

Motivation: Manuscripts, RadLex, UDC, Rameau

R-ConceptSchemeContainment

Ability to explicitly represent the containment of any SKOS individual or statement within a concept scheme

It shall be possible to explicitly represent the containment of any individual which is an instance of a SKOS class (e.g. skos:Concept) or statement that uses SKOS property as predicate (e.g. skos:broader) within a concept scheme.

@@ Linked to SKOS-I-ConceptSchemeContainment @@

R-ConsistencyChecking

Checking the consistency of a concept scheme

Some SKOS applications might require testing the integrity of their concept scheme data. For example, conceptual relationships should only apply to individuals of type skos:Concept, and not for example between the (non-preferred) labels of concepts.

Motivation: GtaaBrowser, MetadataRegistry

@@ Linked issue: SKOS-I-Semantics-10 @@

R-GroupingInConceptHierarchies

Ability to include grouping constructs in concept hierarchies in thesauri

Concept schemes can contain elements (*arrays*, *guide terms*, etc.) used to group normal concepts together, e.g. based on a shared semantic property. While these special elements cannot be used for description purposes, they can be introduced in a concept scheme's hierarchy by means of generalization and specialization links.

@@ Linked to SKOS-I-GroupingInConceptHierarchies, SKOS-I-collections-5 @@

R-IndexingAndNonIndexingConcepts

Ability to distinguish between concepts to be used for indexing and for non-indexing

SKOS should provide different classes for conceptual entities that can be used for indexing resources and for those that cannot be used for such a purpose (e.g. specific qualifiers that can only be used to narrow down the meaning of an existing concept).
Motivation: Manuscripts, UDC, Rameau

@@ Linked to SKOS-I-IndexingAndNonIndexingConcepts, SKOS-I-coordination-8 @@

R-IndexingRelationship

Ability to represent the indexing relationship between a resource and a concept that indexes it

The SKOS model should contain mechanisms to attach a given resource (e.g. corresponding to a document) to a concept the resource is about, e.g. to query for the resources described by a given concept.

Motivation: Manuscripts, Biozen, Aims, BirnLex

@@ Linked to SKOS-I-IndexingRelationship @@

R-LexicalMappingLinks

Correspondence mapping links between lexical labels of concepts in different concept schemes

In the process of mapping different concept schemes, it should be possible to identify correspondence links not only between concepts from these concept schemes, but also between the labels that can be attached to these concepts.

Motivation: RadLex, BirnLex

@@ Linked to SKOS-I-LexicalMappingLinks @@

R-MappingProvenanceInformation

Ability to record provenance information on mappings between concepts in different concept schemes

It shall be possible to record provenance information on mappings between concepts in different concept schemes.

Motivation: MetadataRegistry

@@ Linked to SKOS-I-MappingProvenanceInformation @@

R-RelationshipsBetweenLabels

Representation of links between labels associated to concepts

The SKOS model shall provide means to represent relationships between the terms associated with concepts. Typical examples are translation links between labels from different languages, or the link between one label and its abbreviation, when

this stands for an alternative label for the concept.

Motivation: Manuscripts, Aims, RadLex

4 Conclusion

To elicit the requirements that a new version of the Simple Knowledge Organisation System (SKOS) should meet, the Semantic Web and Deployment working group has issued a call for use cases to the different communities that are concerned by the use of SKOS.

More than 25 submissions have been sent to the working group, which illustrates the variety of usages one can make of such a proposal. In this document, eight of them were selected as being the most representative.

Some of these use cases have come with very high-quality descriptions, and most correspond to development efforts that are presently being carried out, going therefore beyond pure research hypotheses. This gives a sound basis for the process of gathering requirements for SKOS, which the second part of this document describes.

Currently, requirements are divided into accepted and candidate requirements, reflecting the level of consensus they have reached in the Working Group at the time this document was created. In the near future, the Working Group will have to make a final decision regarding the candidate requirements, either accepting them or rejecting them. It will of course have to adapt the existing SKOS material so that it meets the accepted requirements.

References

[DCAM]
DCMI Abstract Model, A. Powell, M. Nilsson, A. Naeve, P. Johnston, 7 March 2005.

[ISO11179-3]
ISO/IEC 11179-3: 2003(E), Information Technology – Metadata Registries (MDR) – Part 3: Registry metamodel and basic attributes, Second edition. R. Gates, Editor, 15 February 2003.

[ISO2788]
ISO 2788:1986
Documentation - Guidelines for the establishment and development of monolingual thesauri. Second edition. ISO TC 46/SC 9, 1986.

[ISO5964]
ISO 5964:1985
Documentation - Guidelines for the establishment and development of multilingual thesauri. First edition. ISO TC 46/SC 9, 1985.

[SWBP-SKOS-CORE-GUIDE]
SKOS Core Guide, A. Miles, D. Brickley, Editors, W3C Working Draft (work in progress), 2 November 2005. Latest version available at http://www.w3.org/TR/swbp-skos-core-guide/.

[SWBP-SKOS-CORE-SPEC]
SKOS Core Vocabulary Specification, A. Miles, D. Brickley, Editors, W3C Working

Draft (work in progress), 2 November 2005. Latest version available at http://www.w3.org/TR/swbp-skos-core-spec/.

[SWBP-SKOS-MAPPING]
SKOS Mapping Vocabulary Specification, A. Miles, D. Brickley, Editors, W3C Working Draft (work in progress), 11 November 2004. Latest version available at http://www.w3.org/2004/02/skos/mapping/spec/.

[SWBPD]
The Semantic Web Best Practices and Deployment Working Group

[SWD]
The Semantic Web Deployment Working Group

[SWD-Charter]
Semantic Web Deployment Working Group (SWDWG) Charter

Acknowledgments

The editors gratefully acknowledge contributions from Lora Aroyo, Hugh Barnes, Bruce Bargmeyer, Sean Barker, Sean Bechhofer, Pieter Bellekens, Hennie Brugman, Dario Cerizza, Irene Celino, Thierry Cloarec, Francesco Corcoglioniti, Sarah Currier, Emanuele Della Valle, Diane Hillmann, Chris Holmes, Bernard Horan, Julian Johnson, Simon Jupp, Johannes Keizer, Walter Koch, Véronique Malaisé, George Macgregor, Frédéric Martin, John McCarthy, Emma McCulloch, Alistair Miles, Mitsuharu Nagamori, Dennis Nicholson, Matthias Samwald, Margherita Sini, Aida Slavic, Davide Sommacampagna, Robert Stevens, Doug Tudhope, Andrea Turati, Bernard Vatant, Anna Veronesi.

W3C Working Draft

Classes CollectableProperty Collection Concept ConceptScheme OrderedCollection

Properties altLabel altSymbol broader changeNote definition editorialNote example hasTopConcept hiddenLabel historyNote inScheme isPrimarySubjectOf isSubjectOf member memberList narrower note prefLabel prefSymbol primarySubject related scopeNote semanticRelation subject subjectIndicator symbol

SKOS Core Vocabulary Specification

W3C Working Draft 2 November 2005

This version:
http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102
Latest version:
http://www.w3.org/TR/swbp-skos-core-spec
Previous version:
http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20050510
Editors:
Alistair Miles, CCLRC
Dan Brickley, W3C
Translations:
en fr de nl pt

Abstract

SKOS Core is a model and an RDF vocabulary for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, 'folksonomies', other types of controlled vocabulary, and also concept schemes embedded in glossaries and terminologies.

The SKOS Core Vocabulary is an application of the Resource Description Framework (RDF), that can be used to express a concept scheme as an RDF graph. Using RDF allows data to be linked to and/or merged with other data, enabling data sources to be distributed across the web, but still be meaningfully composed and integrated.

This document gives a reference-style overview of the SKOS Core Vocabulary as it stands at the time of publication. It also describes the policies for ownership, naming, persistence and change by which the SKOS Core Vocabulary is managed.

This edition of the SKOS Core Vocabulary Specification is a W3C Public Working Draft. It is the authoritative human-readable account of the SKOS Core Vocabulary at the time of publication.

See also the SKOS Core Guide.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this

technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document is a W3C Working Draft published by the Semantic Web Best Practices and Deployment Working Group, part of the W3C Semantic Web Activity. The Working Group intends the SKOS Core Vocabulary Specification to become a W3C Working Group Note (see W3C document maturity-levels). However, other outcomes are possible within the framework of the W3C Process and will be considered in response to deployment experience and feedback from the W3C membership. The Working Group has discussed the potential for SKOS Core to evolve into possible future W3C Recommendation Track work items, and would value feedback on the level of formal standardization that is appropriate.

Changes from the previous version are noted in the Release Notes section.

We encourage public comments on this Working Draft. Please send comments to public-esw-thes@w3.org [archive] and start the subject line of the message with "comment:".

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

The English language variant of this document is normative, all other language variants are informative.

Contents

Introduction

SKOS stands for Simple Knowledge Organisation System. The name SKOS was chosen to emphasise the goal of providing a simple yet powerful model for expressing knowledge organisation systems in a machine-understandable way, within the framework of the Semantic Web.

SKOS Core is a model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, 'folksonomies', other types of controlled vocabulary, and also concept schemes embedded in glossaries and terminologies.

The SKOS Core Vocabulary is an application of the Resource Description Framework (RDF). RDF provides a simple data formalism for talking about things, their properties, inter-relationships, and categories (classes). Using RDF allows data to be linked to and/or merged with other RDF data by Semantic Web applications. In practice, this means that data sources can be distributed across the web in a decentralised way, but still be meaningfully composed and integrated by applications, often in novel and unanticipated ways. See RDF Concepts for an overview of RDF, RDF Semantics for its formal mathematical basis, and RDF Syntax for details of the RDF/XML document format used to exchange RDF data.

This document gives a reference-style overview of the SKOS Core Vocabulary as it stands at the time of publication. It also describes the policies for ownership, naming, persistence and change by which the SKOS Core Vocabulary is managed.

A formal representation of the SKOS Core Vocabulary is maintained in RDF/OWL [latest: http://www.w3.org/2004/02/skos/core]. Historical snapshots of the RDF/OWL description of the SKOS Core Vocabulary can be obtained from [http://www.w3.org/2004/02/skos/core/history/].

Translations

To obtain the definitive RDF/OWL description of the SKOS Core Vocabulary, serialised as RDF/XML, dereference the URI http://www.w3.org/2004/02/skos/core. This includes labels, comments and definitions in English only, which are normative.

To obtain labels, comments and definitions for classes and properties of the SKOS Core Vocabulary in another language, serialised as RDF/XML, dereference one of the following URIs:

- http://www.w3.org/2004/02/skos/core_de [de] (translated by Thomas Bandholtz)
- http://www.w3.org/2004/02/skos/core_fr [fr] (translated by Bernard Vatant)
- http://www.w3.org/2004/02/skos/core_nl [nl] (translated by Mark van Assem)
- http://www.w3.org/2004/02/skos/core_pt [pt] (translated by Tiago Murakami)

Labels, definitions and comments in any language other than English are informative.

Corresponding language variants of this document are also available at the following links:

- SKOS Core Vocabulary Specification (de)
- SKOS Core Vocabulary Specification (en)
- SKOS Core Vocabulary Specification (fr)
- SKOS Core Vocabulary Specification (nl)
- SKOS Core Vocabulary Specification (pt)

If you would like to submit a translation of labels, definitions and comments, or to make a comment on existing translations, send an email to public-esw-thes@w3.org.

See also the SKOS Core Translations web page.

Guide to Term Summary Tables

Each term (i.e. class or property) of the vocabulary is summarised in this document as a table. Each table may have the following rows:

Class or Property:	
URI:	The Universal Resource Identifier.
Label:	A human-readable label.
Definition:	An explanation of the meaning of a class or property.
Comment:	Additional information about meaning and/or proper use.
Example:	An example of the use of a class or property.
Status:	The status (stability level) of the class or property.
Issued:	Date on which the class or property was issued.
Modified:	Date on which the class or property was last modified.
Super-classes:	(Classes only) any declared super-classes.
Super-properties:	(Properties only) any declared super-properties.
Domain:	(Properties only) the declared domain for the property.

Range:	(Properties only) the declared range for the property.
Characteristics:	(Properties only) any declared logical characteristics for the property, e.g. Transitive, Symmetric, Functional, InverseFunctional.
Inverse of:	(Properties only) any declared inverse properties.
Replaces:	Any deprecated terms which the given term has replaced in recommended usage.
Version info:	A note about the modification and/or history of a class or property.
Replaced By:	(Deprecated terms only) the term to use instead of the deprecated term.
Deprecated:	(Deprecated terms only) the date of last modification (i.e. deprecation) of the term.

Policy Statements

N.B. The Working Group is committed to establishing clear expectations around the management of RDF vocabularies, through documentation of process and maintenance policies. This is itself an evolving process. Specifically, this document is itself situated within the W3C Process, and may change and evolve in the light of feedback on SKOS Core and on the SKOS Core policy statements. It should be noted that claims made by the Working Group using the (experimental) persistence and change terminology employed here have as their scope the currently chartered Working Group. They have only draft status within the wider W3C Process. W3C has not delegated to the Working Group any authority to make binding commitments on behalf of W3C beyond those implicit in the formal W3C Process.

Ownership

The W3C gives control over the SKOS Core Vocabulary to working groups within the overall framework of the W3C process. Currently that control resides with the Semantic Web Best Practices and Deployment Working Group, whose chairs have delegated responsibility for maintaining the SKOS Core Vocabulary to the editors of this specification (Alistair Miles and Dan Brickley). When this working group's charter expires, control will revert to W3C as an organization.

The Working Group is committed to a public, consensus-driven design environment for SKOS Core, and to this end conducts SKOS-related discussion in public, in particular drawing on feedback from the Semantic Web Interest Group mailing list public-esw-thes@w3.org.

Naming

The URI for the SKOS Core Vocabulary itself is:

```
http://www.w3.org/2004/02/skos/core
```

The URI for a class or property in the SKOS Core Vocabulary is constructed by appending a fragment identifier to the URI for the SKOS Core Vocabulary. E.g.

```
http://www.w3.org/2004/02/skos/core#Concept
http://www.w3.org/2004/02/skos/core#prefSymbol
```

A fragment identifier for a class always starts with an uppercase character. Where the fragment identifier is comprised of multiple concatenated words, the leading character of each word will be an uppercase character. E.g.

```
Concept
ConceptScheme
```

The fragment identifier for a property starts with a lowercase character. Where the fragment identifier is comprised of multiple concatenated words, the leading character of the second and

each subsequent word will be an uppercase character. E.g.

```
subject
prefLabel
isPrimarySubjectOf
```

Persistence

All editions of the SKOS Core Vocabulary Specification, all editions of the SKOS Core Guide, and the RDF/OWL description of the SKOS Core Vocabulary [http://www.w3.org/2004/02/skos/core] are all declared to be persistent resources, as defined by the persistence policy at [http://www.w3.org/Consortium/Persistence].

Change

The SKOS Core Vocabulary may change. The process for managing changes to the SKOS Core Vocabulary during the chartered lifetime of the Working Group is described below:

1. The Working Group undertakes to review the SKOS Core Vocabulary Specification and the SKOS Core Guide at intervals of 2 months. Subsequent to each review, new Public Working Draft editions of the SKOS Core Guide and the SKOS Core Vocabulary Specification will be published by the Working Group.
2. In the interim period between publication of Public Working Draft editions of the SKOS Core Guide and the SKOS Core Vocabulary Specification (hereafter 'the interim period'), **no changes will be made to the SKOS Core Vocabulary**. The formal RDF/OWL description of the SKOS Core Vocabulary [http://www.w3.org/2004/02/skos/core] will therefore also remain unchanged during this period.
3. In the interim period, the editor's will maintain a public list of proposed changes to the SKOS Core Vocabulary to be put forward at the next review.
4. Members of the public may make proposals for change, or comment on current proposals, by sending an email to the publicly archived Semantic Web Interest Group mailing list public-esw-thes@w3.org [archive].
5. Proposed changes to be put forward for review will be published at least 2 weeks before a scheduled review, to allow time for public comment.
6. At each review, the list of proposed changes to the SKOS Core Vocabulary is presented to the reviewers delegated by the Working Group for approval.
7. Those changes approved by the reviewers, or approved in a modified form after negotiation between the editors and the reviewers, will be implemented by the editors. New Public Working Draft editions of the SKOS Core Guide and SKOS Core Vocabulary Specification, reflecting these changes, will then be published by the Working Group. Approved changes will also be implemented in the formal representation of the SKOS Core Vocabulary in RDF/OWL [http://www.w3.org/2004/02/skos/core]. Therefore the formal representation of the SKOS Core Vocabulary in RDF/OWL [http://www.w3.org/2004/02/skos/core] will always be consistent with the latest Public Working Draft editions of the SKOS Core Guide and the SKOS Core Vocabulary Specification.
8. All changes made will also be reported on the SKOS Change Log [http://esw.w3.org/mt/esw/archives/cat_skos_changelog.html].

Furthermore, at any given time only certain types of change are allowed (following the example of the DCMI namespace policy [http://dublincore.org/documents/dcmi-namespace/]). The types of change allowed depend on the 'status' of the class or property to which the change relates. The status of a class or property may take one of three values:

unstable

The term is unstable, and feedback is welcomed on its current form and utility. It may currently be poorly defined. Its meaning and/or form may be expected to change at any time. Do not implement mission critical systems that depend on this term persisting in its current form. (Changes corresponding to DCMI Namespace Policy types A, B or C may occur.)

testing

The term has gone beyond the raw proposal stage, and is undergoing testing. This term may still change in response to feedback from testing, although it may be expected not to undergo any radical change. The cost to early implementors of changing the term will be considered, however the goal of achieving wider

interoperability and long-term stability may override those considerations. (Changes corresponding to DCMI Namespace Policy types A or B may occur.)

stable

No substantial (i.e. meaning-changing) alterations will take place. Implementors can expect the term to persist in its current form indefinitely. (Minor editorial changes corresponding to DCMI Namespace Policy type A may occur.)

New classes or properties may be added to the SKOS Core Vocabulary (changes corresponding to DCMI Namespace Policy type D), in accordance with the process described above.

A class or property at any of the above status levels may be marked as deprecated, in which case instructions will be given on what to use in its place. A deprecated class property may be expected to remain unchanged indefinitely.

Historical versions (snapshots) of the SKOS Core Vocabulary are maintained, and can be accessed via the web page at [http://www.w3.org/2004/02/skos/core/history/].

Release Notes

The following changes have been made since the previous (first) W3C Public Working Draft edition of the SKOS Core Vocabulary Specification:

Change 1: Use of skos:subjectIndicator

The property `skos:subjectIndicator` was previously used internally within the RDF/OWL description of the SKOS Core Vocabulary, to make statements about the other classes and properties of the SKOS Core Vocabulary. This usage, however, has implications which were both inconvenient and contentious, and has been removed. This change should not have any significant impact, and does not alter recommended usage as described in the SKOS Core Guide.

See [subjectIndicatorUse-1] for a description of the original change proposal.

Change 2: Documentation Properties

The properties `skos:publicNote` and `skos:privateNote` have been deprecated. They are replaced by the property `skos:note`. See the Documentation Properties section of the SKOS Core Guide for recommended usage. An example has been added to the Documentation Properties section of the Guide, illustrating the use of the `dc:audience` property to specify the intended audience of a particular note.

The motivation behind this change is to allow the function of a note to be specified independently from the intended audience, allowing for example different definitions for different audiences.

See [notes-2] for a description of the original change proposal.

Change 3: Range of Symbolic Labelling Properties

The range of the properties `skos:prefSymbol` and `skos:altSymbol` has been changed from `foaf:Image` to `dcmitype:Image`. This change should not have any significant impact, and has been made purely because of the greater stability of the DCMI type vocabulary at this time.

Also, a new property `skos:symbol` ('symbolic label') has been added, as the super-property of both `skos:prefSymbol` ('preferred symbolic label') and `skos:altSymbol` ('alternative symbolic label'). The `skos:symbol` property is roughly analagous to `rdfs:label`, but for labelling a resource with an image resource instead of an RDF literal. The addition of the `skos:symbol` property does not alter recommended usage of `skos:prefSymbol` and `skos:altSymbol` as described in the SKOS Core Guide.

See [symbolicLabelsRange-3] for a description of the original change proposal.

Change 4: Translations

Labels, comments and definitions for the classes and properties of the SKOS Core Vocabulary are now available as RDF in the following languages: French [fr], German [de], Dutch [nl] and Portuguese [pt]. Links have been added from the SKOS Core RDFS/OWL description to these additional sources. Corresponding language variants of this document are also available. This change does not alter recommended usage as described in the SKOS Core Guide.

See [seeAlsoTranslations-4] for a description of the original change proposal.

Classes

Class:CollectableProperty	
URI:	`http://www.w3.org/2004/02/skos/core#CollectableProperty`
Label:	Collectable Property
Definition:	A property which can be used with a skos:Collection.
Comment:	The following rule applies for this property: [(?x ?p ?c) (?c skos:member ?y) (?p rdf:type skos:CollectableProperty) implies (?x ?p ?y)]
Examples:	[link]
Status:	unstable
Issued:	2004-10-20
Modified:	2005-09-29
Class:Collection	
URI:	`http://www.w3.org/2004/02/skos/core#Collection`
Label:	Collection
Definition:	A meaningful collection of concepts.
Comment:	Labelled collections can be used with collectable semantic relation properties e.g. skos:narrower, where you would like a set of concepts to be displayed under a 'node label' in the hierarchy.
Examples:	[link]
Status:	unstable
Issued:	2004-10-20
Modified:	2005-09-29
Class:Concept	
URI:	`http://www.w3.org/2004/02/skos/core#Concept`
Label:	Concept
Definition:	An abstract idea or notion; a unit of thought.
Examples:	[link]
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Class:ConceptScheme	
URI:	`http://www.w3.org/2004/02/skos/core#ConceptScheme`
Label:	Concept Scheme
Definition:	A set of concepts, optionally including statements about semantic relationships between those concepts.
Comment:	Thesauri, classification schemes, subject heading lists, taxonomies, 'folksonomies', and other types of controlled vocabulary are all examples of concept schemes. Concept schemes are also embedded in glossaries and terminologies.

Comment:	A concept scheme may be defined to include concepts from different sources.
Examples:	[link]
Status:	testing
Issued:	2004-03-26
Modified:	2005-10-14
Class:OrderedCollection	
URI:	`http://www.w3.org/2004/02/skos/core#OrderedCollection`
Label:	Ordered Collection
Definition:	An ordered collection of concepts, where both the grouping and the ordering are meaningful.
Comment:	Ordered collections can be used with collectable semantic relation properties, where you would like a set of concepts to be displayed in a specific order, and optionally under a 'node label'.
Examples:	[link]
Super-classes:	`Collection`
Status:	unstable
Issued:	2004-10-20
Modified:	2005-09-29

Properties

Property:altLabel	
URI:	`http://www.w3.org/2004/02/skos/core#altLabel`
Label:	alternative label
Definition:	An alternative lexical label for a resource.
Comment:	Acronyms, abbreviations, spelling variants, and irregular plural/singular forms may be included among the alternative labels for a concept. Mis-spelled terms are normally included as hidden labels (see skos: hiddenLabel).
Examples:	[link]
Super-properties:	`http://www.w3.org/2000/01/rdf-schema#label`
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Property:altSymbol	
URI:	`http://www.w3.org/2004/02/skos/core#altSymbol`
Label:	alternative symbolic label
Definition:	An alternative symbolic label for a resource.
Examples:	[link]
Super-properties:	`symbol`
Status:	testing
Issued:	2004-03-26
Modified:	2005-10-06
Property:broader	
URI:	`http://www.w3.org/2004/02/skos/core#broader`
Label:	has broader
Definition:	A concept that is more general in meaning.
Comment:	Broader concepts are typically rendered as parents in a concept hierarchy (tree).
Examples:	[link]

Super-properties:	semanticRelation
Characteristics:	Transitive
Inverse of:	narrower
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Property:changeNote	
URI:	http://www.w3.org/2004/02/skos/core#changeNote
Label:	change note
Definition:	A note about a modification to a concept.
Examples:	[link]
Super-properties:	note
Status:	unstable
Issued:	2004-10-21
Modified:	2005-10-05
Property:definition	
URI:	http://www.w3.org/2004/02/skos/core#definition
Label:	definition
Definition:	A statement or formal explanation of the meaning of a concept.
Examples:	[link]
Super-properties:	note
Status:	testing
Issued:	2004-03-26
Modified:	2005-10-05
Property:editorialNote	
URI:	http://www.w3.org/2004/02/skos/core#editorialNote
Label:	editorial note
Definition:	A note for an editor, translator or maintainer of the vocabulary.
Examples:	[link]
Super-properties:	note
Status:	unstable
Issued:	2004-10-21
Modified:	2005-10-05
Property:example	
URI:	http://www.w3.org/2004/02/skos/core#example
Label:	example
Definition:	An example of the use of a concept.
Examples:	[link]
Super-properties:	note
Status:	testing
Issued:	2004-03-26
Modified:	2005-10-05
Property:hasTopConcept	
URI:	http://www.w3.org/2004/02/skos/core#hasTopConcept
Label:	has top concept
Definition:	A top level concept in the concept scheme.
Examples:	[link]
Domain:	ConceptScheme

Range:	Concept
Status:	testing
Issued:	2004-08-19
Modified:	2005-09-29
Replaces:	TopConcept
Property:hiddenLabel	
URI:	http://www.w3.org/2004/02/skos/core#hiddenLabel
Label:	hidden label
Definition:	A lexical label for a resource that should be hidden when generating visual displays of the resource, but should still be accessible to free text search operations.
Examples:	[link]
Super-properties:	http://www.w3.org/2000/01/rdf-schema#label
Status:	unstable
Issued:	2004-12-15
Modified:	2005-09-29
Property:historyNote	
URI:	http://www.w3.org/2004/02/skos/core#historyNote
Label:	history note
Definition:	A note about the past state/use/meaning of a concept.
Examples:	[link]
Super-properties:	note
Status:	unstable
Issued:	2004-10-21
Modified:	2005-10-05
Property:inScheme	
URI:	http://www.w3.org/2004/02/skos/core#inScheme
Label:	in scheme
Definition:	A concept scheme in which the concept is included.
Comment:	A concept may be a member of more than one concept scheme.
Examples:	[link]
Domain:	Concept
Range:	ConceptScheme
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Property:isPrimarySubjectOf	
URI:	http://www.w3.org/2004/02/skos/core#isPrimarySubjectOf
Label:	is primary subject of
Definition:	A resource for which the concept is the primary subject.
Examples:	[link]
Super-properties:	isSubjectOf
Inverse of:	primarySubject
Status:	unstable
Issued:	2004-10-22
Modified:	2005-09-29
Property:isSubjectOf	
URI:	http://www.w3.org/2004/02/skos/core#isSubjectOf
Label:	is subject of
Definition:	A resource for which the concept is a subject.

Examples:	[link]
Domain:	Concept
Inverse of:	subject
Status:	unstable
Issued:	2004-10-22
Modified:	2005-09-29
Property:member	
URI:	http://www.w3.org/2004/02/skos/core#member
Label:	member
Definition:	A member of a collection.
Examples:	[link]
Domain:	Collection
Range:	http://www.w3.org/1999/02/22-rdf-syntax-ns#Resource
Status:	unstable
Issued:	2004-10-20
Modified:	2005-09-29
Property:memberList	
URI:	http://www.w3.org/2004/02/skos/core#memberList
Label:	member list
Definition:	An RDF list containing the members of an ordered collection.
Comment:	The following rule applies for this property: [(?c skos:memberList ?l) elementOfList(?e,?l) implies (?c skos:member ?e)]
Examples:	[link]
Domain:	OrderedCollection
Range:	http://www.w3.org/1999/02/22-rdf-syntax-ns#List
Status:	unstable
Issued:	2004-10-20
Modified:	2005-09-29
Property:narrower	
URI:	http://www.w3.org/2004/02/skos/core#narrower
Label:	has narrower
Definition:	A concept that is more specific in meaning.
Comment:	Narrower concepts are typically rendered as children in a concept hierarchy (tree).
Examples:	[link]
Super-properties:	semanticRelation
Characteristics:	Transitive
Inverse of:	broader
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Property:note	
URI:	http://www.w3.org/2004/02/skos/core#note
Label:	note
Definition:	A general note, for any purpose.
Comment:	This property may be used directly, or as a super-property for more specific note types.
Examples:	[link]
Status:	unstable
Issued:	2005-10-05

Replaces:	publicNote privateNote
Property:prefLabel	
URI:	http://www.w3.org/2004/02/skos/core#prefLabel
Label:	preferred label
Definition:	The preferred lexical label for a resource, in a given language.
Comment:	No two concepts in the same concept scheme may have the same value for skos:prefLabel in a given language.
Examples:	[link]
Super-properties:	http://www.w3.org/2000/01/rdf-schema#label
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Property:prefSymbol	
URI:	http://www.w3.org/2004/02/skos/core#prefSymbol
Label:	preferred symbolic label
Definition:	The preferred symbolic label for a resource.
Comment:	No two concepts in the same concept scheme may have the same value for skos:prefSymbol.
Examples:	[link]
Super-properties:	symbol
Status:	testing
Issued:	2004-03-26
Modified:	2005-10-06
Property:primarySubject	
URI:	http://www.w3.org/2004/02/skos/core#primarySubject
Label:	has primary subject
Definition:	A concept that is the primary subject of the resource.
Comment:	A resource may have only one primary subject per concept scheme.
Examples:	[link]
Super-properties:	subject
Inverse of:	isPrimarySubjectOf
Status:	unstable
Issued:	2004-10-22
Modified:	2005-09-29
Property:related	
URI:	http://www.w3.org/2004/02/skos/core#related
Label:	related to
Definition:	A concept with which there is an associative semantic relationship.
Examples:	[link]
Super-properties:	semanticRelation http://www.w3.org/2000/01/rdf-schema#seeAlso
Characteristics:	Symmetric
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Property:scopeNote	
URI:	http://www.w3.org/2004/02/skos/core#scopeNote
Label:	scope note
Definition:	A note that helps to clarify the meaning of a concept.
Examples:	[link]

Super-properties:	note
Status:	testing
Issued:	2004-03-26
Modified:	2005-10-05
Property:semanticRelation	
URI:	http://www.w3.org/2004/02/skos/core#semanticRelation
Label:	semantic relation
Definition:	A concept related by meaning.
Comment:	This property should not be used directly, but as a super-property for all properties denoting a relationship of meaning between concepts.
Examples:	[link]
Domain:	Concept
Range:	Concept
Status:	testing
Issued:	2004-03-26
Modified:	2005-09-29
Property:subject	
URI:	http://www.w3.org/2004/02/skos/core#subject
Label:	has subject
Definition:	A concept that is a subject of the resource.
Comment:	The following rule may be applied for this property: [(?d skos:subject ?x)(?x skos:broader ?y) implies (?d skos:subject ?y)]
Examples:	[link]
Range:	Concept
Super-properties:	http://purl.org/dc/elements/1.1/subject
Inverse of:	isSubjectOf
Status:	unstable
Issued:	2004-10-22
Modified:	2005-09-29
Property:subjectIndicator	
URI:	http://www.w3.org/2004/02/skos/core#subjectIndicator
Label:	subject indicator
Definition:	A subject indicator for a concept. [The notion of 'subject indicator' is defined here with reference to the latest definition endorsed by the OASIS Published Subjects Technical Committee.]
Comment:	This property allows subject indicators to be used for concept identification in place of or in addition to directly assigned URIs.
Examples:	[link]
Domain:	Concept
Range:	http://xmlns.com/foaf/0.1/Document
Characteristics:	Inverse Functional
Status:	unstable
Issued:	2004-11-11
Modified:	2005-09-29
Property:symbol	
URI:	http://www.w3.org/2004/02/skos/core#symbol
Label:	symbolic label
Definition:	An image that is a symbolic label for the resource.
Comment:	This property is roughly analagous to rdfs:label, but for labelling resources with images that have retrievable representations, rather than RDF literals.
Examples:	[link]

Domain:	http://www.w3.org/1999/02/22-rdf-syntax-ns#Resource
Range:	http://purl.org/dc/dcmitype/Image
Status:	unstable
Issued:	2005-10-06

Deprecated Classes and Properties

Deprecated Class:TopConcept	
URI:	http://www.w3.org/2004/02/skos/core#TopConcept
Issued:	2004-03-26
Deprecated:	2004-08-19
Replaced By:	hasTopConcept
Version info:	This class is now deprecated. To indicate that a concept is a top concept in a scheme, now use the skos:hasTopConcept property.
Deprecated Property:broaderGeneric	
URI:	http://www.w3.org/2004/02/skos/core#broaderGeneric
Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	http://www.w3.org/2004/02/skos/extensions#broaderGeneric
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/
Deprecated Property:broaderInstantive	
URI:	http://www.w3.org/2004/02/skos/core#broaderInstantive
Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	http://www.w3.org/2004/02/skos/extensions#broaderInstantive
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/
Deprecated Property:broaderPartitive	
URI:	http://www.w3.org/2004/02/skos/core#broaderPartitive
Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	http://www.w3.org/2004/02/skos/extensions#broaderPartitive
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/
Deprecated Property:externalID	
URI:	http://www.w3.org/2004/02/skos/core#externalID
Issued:	2004-03-26
Deprecated:	2004-10-20
Replaced By:	http://purl.org/dc/elements/1.1/identifier
Version info:	This property is now deprecated. Use dc:identifier instead.
Deprecated Property:narrowerGeneric	
URI:	http://www.w3.org/2004/02/skos/core#narrowerGeneric
Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	http://www.w3.org/2004/02/skos/extensions#narrowerGeneric
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/
Deprecated Property:narrowerInstantive	
URI:	http://www.w3.org/2004/02/skos/core#narrowerInstantive

Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	`http://www.w3.org/2004/02/skos/extensions#narrowerInstantive`
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/
Deprecated Property:narrowerPartitive	
URI:	`http://www.w3.org/2004/02/skos/core#narrowerPartitive`
Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	`http://www.w3.org/2004/02/skos/extensions#narrowerPartitive`
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/
Deprecated Property:privateNote	
URI:	`http://www.w3.org/2004/02/skos/core#privateNote`
Issued:	2004-10-21
Deprecated:	2005-10-05
Replaced By:	`note`
Version info:	This property now replaced by skos:note. To describe a note for a particular audience (e.g. 'editor', 'indexer', 'general user') use a note property with a related resource description and the dc:audience property.
Deprecated Property:publicNote	
URI:	`http://www.w3.org/2004/02/skos/core#publicNote`
Issued:	2004-10-21
Deprecated:	2005-10-05
Replaced By:	`note`
Version info:	This property now replaced by skos:note. To describe a note for a particular audience (e.g. 'editor', 'indexer', 'general user') use a note property with a related resource description and the dc:audience property.
Deprecated Property:relatedHasPart	
URI:	`http://www.w3.org/2004/02/skos/core#relatedHasPart`
Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	`http://www.w3.org/2004/02/skos/extensions#relatedHasPart`
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/
Deprecated Property:relatedPartOf	
URI:	`http://www.w3.org/2004/02/skos/core#relatedPartOf`
Issued:	2004-03-26
Deprecated:	2004-10-18
Replaced By:	`http://www.w3.org/2004/02/skos/extensions#relatedPartOf`
Version info:	This term has been moved to the 'SKOS Extensions' vocabulary. See http://www.w3.org/2004/02/skos/extensions/

Acknowledgements

This work has benefited greatly from contributions made by members of the public-esw-thes@w3.org mailing list. Their feedback, suggestions and encouragement are gratefully acknowledged. SKOS Core draws heavily on conventions and traditions in the thesaurus and library communities, whose practitioners have very generously shared their experiences and insights during the design of SKOS Core.

The following are gratefully acknowledged for their translations: Thomas Bandholtz, Bernard Vatant, Mark van Assem, Tiago Murakami.


W3C
XHTML
1.0
W3C
CSS

RDFa Primer 1.0

Embedding Structured Data in Web Pages

Editors' Draft

This version:
http://www.w3.org/2006/07/SWD/RDFa/primer/20070918/ $Id: Overview.xml,v 1.1 2007/09/18 23:01:44 adida Exp $
Latest version:
http://www.w3.org/2006/07/SWD/RDFa/primer
Previous version:
- http://www.w3.org/2006/07/SWD/RDFa/primer/20070910/
- http://www.w3.org/2006/07/SWD/RDFa/primer/20070302/
- http://www.w3.org/2006/07/SWD/RDFa/primer/20070227/
- http://www.w3.org/2006/07/SWD/RDFa/primer/20070104/
- http://www.w3.org/2001/sw/BestPractices/HTML/2006-04-24-rdfa-primer

Editors:
Ben Adida, Creative Commons <ben@adida.net>
Mark Birbeck, x-port.net Ltd. <mark.birbeck@x-port.net>

Status of this Document

This is an internal draft produced by the Semantic Web Deployment Working Group [SWD-WG], in cooperation with the XHTML2 Working Group [XHTML2-WG]. Initial work on RDFa began with the Semantic Web Best Practices and Deployment Working Group [SWBPD-WG] and the HTML Working Group [HTML-WG].

This document is for internal review only and is subject to change without notice. This document has *no formal standing within the W3C*.

Changes

Since Working Draft #3 of this document:

- Section 3 on Shutr was split up and moved to "advanced concepts", rather than be

a semi-syntax'ish thing.
- the `class` attribute is no longer used to declare `rdf:type`, as this was found to be too confusing. We now use the new `instanceof` attribute.
- the updated syntax for chaining (previously referred to as striping) is introduced.

Table of Contents

---

1 Purpose and Preliminaries

Current web pages, written in XHTML, contain inherent structured data: calendar events, contact information, photo captions, song titles, copyright licensing information, etc. When authors and publishers can express this data precisely, and when tools can read it robustly, a new world of user functionality becomes available, letting users transfer structured data between applications and web sites. An event on a web page can be directly imported into a desktop calendar. A license on a document can be detected to inform the user of his rights automatically. A photo's creator, camera setting information, resolution, and topic can be published as easily as the original photo itself.

RDFa lets XHTML authors express this structured data using extra XHTML attributes. Where the data is already present on the page, e.g. a photo caption, the author need not repeat it. A web publisher can easily reuse concepts, e.g. an event's date, defined by other publishers, or create new ones altogether. RDFa gets its expressive power from RDF, though the reader need not understand RDF before reading this document.

RDFa uses Compact URIs, which express a URI using a prefix, e.g. `dc:title` where `dc:` stands for `http://purl.org/dc/elements/1.1/`. In this document, for simplicity's sake, the following prefixes are assumed to be already declared: `dc` for Dublin Core, `foaf` for Friend-Of-A-Friend, `cc` for Creative Commons, and `xsd` for XML Schema Definitions:

- `dc`: http://purl.org/dc/elements/1.1/
- `foaf`: http://xmlns.com/foaf/0.1/
- `cc`: http://web.resource.org/cc/
- `xsd`: http://www.w3.org/2001/XMLSchema#

We use standard XHTML notation for elements and attributes: both are denoted using fixed-width lowercase font, e.g. `div`, and attributes are differentiated using a preceding '@' character, e.g. `@href`.

2 Simple Data: Publishing Events and Contacts

Jo keeps a private blog for her friends and family.

2.1 The Basic XHTML, before RDFa

Jo is organizing one last summer Barbecue, which she hopes all of her friends and family will attend. She blogs an announcement of this get-together at her private blog, `http://jo-blog.example.org/`. Her blog also includes her contact information:

```
<html>
    <head><title>Jo's Friends and Family Blog</title></head>
    <body>
...
    <p>
        I'm holding one last summer Barbecue, on September 16th at 4pm.
    </p>
...
    <p class="contactinfo">
        Jo Smith. Web hacker
        at
        <a href="http://example.org">
            Example.org
        </a>.
        You can contact me
        <a href="mailto:jo@example.org">
            via email
        </a>.
    </p>
...
    </body>
</html>
```

This short piece of mark-up contains important structured data.

The markup describes an *event*: a Barbecue that Jo is hosting. This Barbecue *starts* at 4pm on September 16th. A *summary* of the event is "one last summer Barbecue." We also have contact information for Jo: she works for the *organization* Example.org, with *job title* of "Web Hacker." She can be contacted at the *email* address "jo@example.org."

At the moment, it is very difficult for software — like web browsers and search engines — to make use of this data's implicit structure. We need a standard mechanism to explicitly express it, so that it can be extracted consistently. This is precisely where RDFa comes in.

2.2 Publishing An Event

Jo would like to label this blog entry so that her friends and family can add her Barbecue directly to their calendar. RDFa allows her to express this structure using a small handful of extra attributes. Since this is a calendar event, Jo will specifically use the iCal vocabulary [ICAL-RDF] to denote the data's structure.

The first step is to reference the iCal vocabulary within the XHTML page, so that Jo's friends' web browsers can look up the calendar concepts and make use of them:

```
<html xmlns:cal="http://www.w3.org/2002/12/cal/ical#">
    ...
```

then, Jo declares a new event:

```
 <p instanceof="cal:Vevent"> ... </p>
```

Note how `@instanceof`
is used here to define the type of data being expressed. The use of this attribute on the `p` element ensures that, by default, data expressed inside this element refers to the same event. There, Jo can set up the event fields, reusing the existing XHTML. For example, the event summary can be declared as:

```
  I'm holding
  <span property="cal:summary">one last summer Barbecue</span>,
```

`@property` on `span` declares the data field `cal:summary`. The existing content, "one last summer Barbecue", is the value of this field. Sometimes, this isn't the desired effect. Specifically, the start time of the event should be displayed pleasantly — "September 16th" —, but should likely be represented in a machine-parsable way, the standard iCal format: `20070916T1600-0500`
(which is clearly not pleasant for human display). In this case, the markup needs only a slight modification:

```
  <span property="cal:dtstart" content="20070916T1600-0500">
    September 16th at 4pm
  </span>
```

The actual content of the `span`, "September 16th at 4pm", is ignored by the RDFa portion of the browser: it has been replaced by the explicit `@content`. The full markup is then:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/D
<html xmlns:cal="http://www.w3.org/2002/12/cal/ical#">
    <head><title>Jo's Friends and Family Blog</title></head>
```

```
    <body>
...
    <p instanceof="cal:Vevent">
        I'm holding
        <span property="cal:summary">one last summer Barbecue</span>,
        on
        <span property="cal:dtstart" content="20070916T1600-0500">
            September 16th at 4pm.
        </span>
    </p>
...
    </body>
</html>
```

Note that Jo could have used any other XHTML element, not just `span`, to mark up her event. In other words, when the event information is already laid out in the XHTML using elements such as `h1`, `em`, `div`, etc..., Jo can simply add the `@instanceof` data type declaration, `@property`, and optionally `@content` to mark up the event.

(For the RDF-inclined reader, the RDF triples that correspond to the above markup are available in Section **4.1 Events and Contact Information**.)

2.3 Publishing Contact Information

Now that Jo has published her event in a human-and-machine-readable way, she realizes there is much data on her blog that she can mark up in the same way. Her contact information, in particular, is an easy target:

```
...
  <p class="contactinfo">
    Jo Smith. Web hacker
    at
    <a href="http://example.org">
        Example.org
    </a>.
    You can contact me
      <a href="mailto:jo@example.org">
        via email
      </a>.
  </p>
...
```

Jo discovers the vCard RDF vocabulary [VCARD-RDF], which she adds to her existing page. Since Jo thinks of vCards as a way to publish her contact information, she uses the prefix `contact`
to designate this vocabulary. Note that adding the vCard vocabulary is just as easy and does not interfere with the already added iCal vocabulary:

```
<html xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
      xmlns:contact="http://www.w3.org/2001/vcard-rdf/3.0#">
...
```

Jo then sets up her vCard using RDFa, by deciding that the `p` will be the container for her vcard. She notes that the vCard schema does not require declaring a vCard type. Instead, it is recommended that a vCard refer to a web page that identifies the individual.

Jo thus uses RDFa's `@about`
for just for this purpose, indicating that all contained XHTML pertains to Jo's designated URI. `@about` is inherited from parent elements in the XHTML: the value of the `@about` on the nearest containing element applies.

```
...
  <p class="contactinfo" about="http://example.org/staff/jo">

      <!-- everything here pertains to http://example.org/staff/jo -->

  </p>
...
```

"Simple enough!" Jo realizes, noting that RDFa does not interfere with her existing markup, in particular the `@class` she uses for styling. She adds her first vCard fields: name, title, organization and email.

```
...
  <p class="contactinfo"  about="http://example.org/staff/jo">

    <span property="contact:fn">Jo Smith</span>.

    <span property="contact:title">Web hacker</span>

    at

    <a rel="contact:org" href="http://example.org">
        Example.org
    </a>.

    You can contact me

    <a rel="contact:email" href="mailto:jo@example.org">
        via email
    </a>.

  </p>
...
```

Notice how Jo was able to use `@rel`
directly within the anchor element to designate her organization and email address. `@rel` indicates the type of *relationship* between the current URI, designated by `@about`, and the target URI, designated by `@href`. Specifically, `contact:org` indicates a relationship of type "vCard organization", while `contact:email` indicates a relationship of type "vCard email".

For simplicity's sake, we have slightly abused the vCard vocabulary above: vCard technically requires that the *type* of the email address be specified, e.g. work or home email. In Section **3.4 Layers of Structure — Subresources**, we show how `rel` can be used without a corresponding `href`, in order to create subresources and provide correct markup for expressing data such as full vCards.

2.4 The Complete XHTML with RDFa

Jo's complete XHTML with RDFa is thus:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN" "http://www.w3.org/MarkUp/D
<html xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
      xmlns:contact="http://www.w3.org/2001/vcard-rdf/3.0#">
  <head>
    <title>Jo's Friends and Family Blog</title>
  </head>
  <body>
...
  <p instanceof="cal:Vevent">
    I'm holding
    <span property="cal:summary">
      one last summer Barbecue,
    </span>
    on
    <span property="cal:dtstart" content="20070916T1600-0500">
      September 16th at 4pm.
    </span>
  </p>
...
  <p class="contactinfo" about="http://example.org/staff/jo">
    <span property="contact:fn">Jo Smith</span>.
    <span property="contact:title">Web hacker</span>
    at
    <a rel="contact:org" href="http://example.org">
      Example.org
    </a>.
    You can contact me
    <a rel="contact:email" href="mailto:jo@example.org">
      via email
    </a>.
  </p>
...
    </body>
</html>
```

If Jo changes her email address link, her organization, or the description of her event, RDFa-enabled browsers will automatically pick up these changes in the marked up, structured data. The only places where this doesn't happen is when `@content` overrides the element's content, which is inevitable when human and machine readability are at odds.

(Once again, the RDF-inclined reader will want to consult the resulting RDF triples **4.1 Events and Contact Information**.)

2.5 Working Within a Fragment of the XHTML

What if Jo does not have complete control over the XHTML of her blog? For example, she may be using a content management system which makes it particularly difficult to add the vocabularies in the `html`
element at the top of her page without adding it to every page on her site. Or, she may be using a web provider that doesn't allow her to change the header of the page to begin with.

Fortunately, RDFa uses compact URIs, where prefixes can be declared using standard XML namespace conventions. Thus, vocabularies can be imported "locally" to an XHTML element. Jo's blog page could express the exact same structured data with the following markup:

```
<html>
  <head>
    <title>Jo's Friends and Family Blog</title>
  </head>
  <body>
...
  <p instanceof="cal:Vevent"
     xmlns:cal="http://www.w3.org/2002/12/cal/ical#">
    I'm holding
    <span property="cal:summary">
      one last summer Barbecue,
    </span>
    on
    <span property="cal:dtstart" content="20070916T1600-0500">
      September 16th at 4pm.
    </span>
  </p>
...
  <p class="contactinfo" about="http://example.org/staff/jo"
     xmlns:contact="http://www.w3.org/2001/vcard-rdf/3.0#">
    <span property="contact:fn">
      Jo Smith
    </span>.
    <span property="contact:title">
      Web hacker
    </span>
    at
    <a rel="contact:org" href="http://example.org">
      Example.org
    </a>.
    You can contact me
    <a rel="contact:email" href="mailto:jo@example.org">
      via email
    </a>.
  </p>
...
  </body>
</html>
```

In this case, each `p`
only needs one vocabulary: the first uses iCal, the second uses vCard. When needed, more than one vocabulary can be imported into any element, not just `html`. This makes copying and pasting XHTML with RDFa much easier. In particular, it allows web widgets to carry their own RDFa in a self-contained manner.

3 Advanced Concepts: Custom Vocabularies, Document Fragments, Complex Data, ...

RDFa can do much more than the simple examples described above. This next section explores some of its advanced capabilities:

- the creation of a custom vocabulary,
- the use of precise datatypes,
- the description of resources beyond the current web page, and
- the definition and annotation of "subresources".

3.1 Creating a Custom Vocabulary and Using Compact URIs

All field names and data types in RDFa are URIs, e.g. `http://purl.org/dc/elements/1.1/title` is the "Dublin Core title" field. In RDFa, we often use compact versions of those URIs, by

- defining a prefix using XML namespace conventions, and
- using the prefixed notation to designate the URI.

This helps keep the markup short and clean:

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/">
   <span property="dc:title">Yowl</span>,
   created by
   <span property="dc:creator">Mark Birbeck</span>.
</div>
```

Because concepts are simply URIs, it is trivial to create one's own vocabulary: simply mint new URIs in a domain you control, and use them in RDFa markup. This is, in fact, the power of RDF, RDFa's underlying technology.

Consider a (fictional) photo management web site called *Shutr*, whose web site is `http://www.shutr.net`. Users of Shutr can upload their photos at will, annotate them, organize them into albums, and share them with the world. They can choose to keep these photos private, or make them available for public consumption under licensing terms of their choosing.

Shutr chooses to mark up its photos with RDFa, so that browsers may be able to extract information automatically. Some concepts, such as `dc:title`, `dc:date`, etc. can be clearly reused from the Dublin Core vocabulary, but other concepts, such as lens settings, camera model, and other photographer parameters, may need to be defined from scratch. For this purpose, Shutr defines a vocabulary namespace URI:

`http://shutr.net/vocab/1.0/`

Shutr can then publish terms such as `http://shutr.net/vocab/1.0/takenWithCamera`, `http://shutr.net/vocab/1.0/aperture`, etc.

The main benefit of RDF is precisely that all vocabulary components are simply URIs. anyone can define a vocabulary, extend it with a few additional concepts, make these new concepts immediately accessible to others, and provide a description of the concept simply by "filling in" the web page that is returned when the concept URI is accessed. No centralized organization is needed to coordinate the creation and extension of vocabularies: RDF allows one to declare relationships (including full equivalence) between concepts that develop organically in the community.

3.2 Qualifying Other Documents and Document Chunks

Shutr may choose to present many photos in a given XHTML page. In particular, at the URI `http://www.shutr.net/user/markb/album/12345`, all of the album's photos will appear

inline. Structured data about each photo can be included simply by specifying `@about`, which indicates the resource that fields refer to within that XHTML element.

```
<ul>
  <li> <a href="/user/markb/photo/23456">
         <img src="/user/markb/photo/23456_thumbnail" />
       </a>

    <span about="/user/markb/photo/23456"
          property="dc:title">Sunset in Nice</span>

  </li>

  <li> <a href="/user/markb/photo/34567">
         <img src="/user/markb/photo/34567_thumbnail" />
       </a>

    <span about="/user/markb/photo/34567"
          property="dc:title">W3C Meeting in Mandelieu</span>

  </li>
</ul>
```

This same approach applies when the field value is a URI. For example, each photo in the album has a creator and may have its own copyright license. We can use the convenient inheritance of `@about`
to refer to the photo once, and add as many fields as we need:

```
<ul>
  <li about="/user/markb/photo/23456">

    <a href="/user/markb/photo/23456">
      <img src="/user/markb/photo/23456_thumbnail" />
    </a>

    <span property="dc:title">Sunset in Nice</span>

    taken by photographer

    <a property="dc:creator"
       href="/user/markb">Mark Birbeck</a>,

    licensed under a

    <a rel="cc:license"
       href="http://creativecommons.org/licenses/by-nc/3.0/">
      Creative Commons Non-Commercial License
    </a>.

  </li>

  <li about="/user/markb/photo/34567">

    <a href="/user/markb/photo/34567">
      <img src="/user/markb/photo/34567_thumbnail" />
    </a>

    <span property="dc:title">W3C Meeting in Mandelieu</span>

    taken by photographer
```

```
      <a property="dc:creator"
         href="/user/stevenp">Steven Pemberton</a>,

      licensed under a

      <a rel="cc:license"
         href="http://creativecommons.org/licenses/by/3.0/">
           Creative Commons Commercial License
      </a>.

    </li>
  </ul>
```

While it makes sense for Shutr to have a whole web page dedicated to each photo album, it might not make as much sense to have a single page for each camera owned by a user. A single page that describes *all* cameras belonging to a single user is more appropriate. For this purpose, RDFa provides ways to mark up document fragments using natural XHTML constructs.

Consider the page `http://www.shutr.net/user/markb/cameras`, which, as its URI implies, lists Mark Birbeck's cameras. Its XHTML contains:

```
  <ul>
    <li id="nikon_d200"> Nikon D200, 3 pictures/second.
    </li>

    <li id="canon_sd550"> Canon Powershot SD550, 5 pictures/second.
    </li>
  </ul>
```

and the photo page will then include information about which camera was used to take each photo:

```
  <ul>
    <li>
      <img src="/user/markb/photo/23456_thumbnail" />
      ...
      using the <a href="/user/markb/cameras#nikon_d200">Nikon D200</a>,
      ...
    </li>
  ...
  </ul>
```

The RDFa syntax for formally specifying the relationship is exactly the same as before, as expected:

```
  <ul>
    <li about="/user/markb/photo/23456">
      ...
      using the <a rel="shutr:takenWithCamera"
           href="/user/markb/cameras#nikon_d200">Nikon D200</a>,
      ...
    </li>
  ...
  </ul>
```

Then, the XHTML snippet at `http://www.shutr.net/user/markb/cameras` is:

```
<ul>
  <li id="nikon_d200" about="#nikon_d200">

    <span property="dc:title">Nikon D200</span>

    <span property="shutr:frameRate">3 pictures/second</span>

  </li>

  <li id="canon_sd550" about="#canon_sd550">

    <span property="dc:title">Canon Powershot SD550</span>

    <span property="shutr:frameRate">5 pictures/second</span>

  </li>
</ul>
```

Notice, again, how text can serve both for the human and machine readable versions: there is no need to keep a separate file up-to-date.

The RDF interpretation of the above markup can be found in **4.2 Custom Vocabularies and Datatypes**.

3.3 Data Types

When dealing with fields of structured data, one may well want (or need) to specify a data type so that computer programs that read the data can make sense of it. Consider the expression of a date. We have already seen how the human-rendered and machine-readable data may not be the same, and how we can use `@content` to provide a machine-readable value. Adding a datatype is only one more attribute: `@datatype`. For example, when expressing the date on which a photo was taken:

```
<ul>

  <li about="/user/markb/photo/23456">

    ...

    taken on
    <span property="dc:date" content="2007-05-12" datatype="xsd:date">
      May 12th, 2007
    </span>

    ...

  </li>

</ul>
```

RDFa uses XML schema data types [XSD-DT].

The RDF interpretation of the above markup can be found in **4.2 Custom Vocabularies and Datatypes**.

3.4 Layers of Structure — Subresources

Sometimes, one may need to mark up a resource with a number of fields, all without giving it a URL, or even a fragment identifier. Consider the case where Shutr decides to let users annotate photos in order to indicate which individuals are depicted in the photo. The barebones XHTML is:

```
<div>
  This photo depicts Mark Birbeck (mark@example.org) and Steven Pemberton (steven@
</div>
```

The simplest way to mark this up without attempting to resolve unique identities for photo subjects is to define *subresources*, effectively new resources that are not given a name. (In RDF, we call these blank nodes.) The following markup will do just that:

```
<div about="/user/markb/photo/23456">
  This photo depicts

  <span rel="foaf:depicts">
    <span property="foaf:firstname">Mark</span>
    <span property="foaf:lastname">Birbeck</span>
    (<span property="foaf:mbox">mark@example.org</span>)
  </span>

  and

  <span rel="foaf:depicts">
    <span property="foaf:firstname">Steven</span>
    <span property="foaf:lastname">Pemberton</span>
    (<span property="foaf:mbox">steven@example.org<span>).
  </span>

</div>
```

The above markup uses the FOAF (Friend-Of-A-Friend) vocabulary which includes the field `foaf:depicts`
that relates a photograph with a person depicted in the photograph. The use of `@rel` without `@href` triggers the definition of a new subresource, which is then the value of the `foaf:depicts`
field. Then, all contained markup applies to this new subresource. In RDFa-speak, we call this "chaining," as it allows one to easily chain one resource (e.g. the photo) to a subresource (e.g. each of the two persons depicted).

The RDF interpretation of the above markup can be found in **4.3 Layered Data and Subresources**.

3.5 Using `@src` on `img`, and the use of `@rev`

Shutr authors may notice that, in a number of cases, the URI of the photos it displays inline using the `img` element is actually the same as `@about` for marking up the photo's fields. In order to minimize markup, RDFa allows authors to make use of `@src` on an `img` element: it behaves just like `@href`.

Consider Mark's profile page on Shutr, which lists all of his albums and cameras. This

page will likely include a picture of Mark himself:

```
<div>
  <h1>Mark Birbeck's Photos</h1>
  <img src="/user/markb/profile_photo.jpg" />
  ...
</div>
```

Shutr may want to indicate that this is Mark's photo, using the FOAF field `foaf:img` defined specifically for this purpose. This can be accomplished as follows:

```
<div about="/user/markb">
  <h1>Mark Birbeck's Photos</h1>
  <img rel="foaf:img" src="/users/markb/profile_photo.jpg" />
  ...
</div>
```

Shutr then notes that the profile photo isn't only Mark's profile photo, it also happens to depict Mark, since Mark obviously appears in his own profile photo (hopefully). This requires expressing an *inverse relationship*, where the field is actually added to the image's URI, not to Mark's profile.

For this purpose, RDFa provides `@rev`, which can be applied to `img` or any other element. `@rev` functions much like `@rel`, except the direction of the relationship is reversed:

```
<div about="/user/markb">
  <h1>Mark Birbeck's Photos</h1>
  <img rel="foaf:img" rev="foaf:depicts"
       src="/user/markb/profile_photo.jpg" />
  ...
</div>
```

In other words, Mark has, as his main image, `/user/markb/profile_photo.jpg`, which of course happens to depict Mark.

The RDF interpretation of the above markup can be found in **4.4 Using @src on img, and the use of @rev**.

3.6 Overriding `@href` and `@src`

When the displayed content is not quite the correct machine-readable data, we used `@content` to override it. In some cases, the navigable link is not quite the right machine-readable data, either. Consider, for example, the case where the displayed photo on Mark Birbeck's profile is, in fact, just a thumbnail, while his official FOAF image is the full-sized version. In this case, and in any case where `@href` or `@src` appears and needs to be overridden by another URI, another RDFa attribute, `@resource`, is used.

The XHTML written above can then be transformed to:

```
<div about="/user/markb">
  <h1>Mark Birbeck's Photos</h1>
```

```
    <img rel="foaf:img"
         resource="/user/markb/profile_photo.jpg"
         src="/user/markb/profile_photo_thumbnail.jpg" />
  ...
</div>
```

Here, the loaded image will use the thumbnail, but an RDFa-aware browser will know that the machine-readable data only cares about the full-sized version specified in `@resource`.

`@resource`
can be particularly useful in cases where the URI is not navigable in any way, e.g. a book's ISBN number represented as `URN:ISBN:0-395-36341-1`.

The RDF interpretation of this markup can be found in **4.5 Overriding @href and @src**

4 RDF Correspondence

RDF [RDF]
is the W3C's standard for interoperable structured data. Though one need not be versed in RDF to understand the basic concepts of RDFa, it helps to know that RDFa is effectively the embedding of RDF in XHTML.

Briefly, RDF is an abstract generic data model. An RDF statement is a triple, composed of a subject, a predicate, and an object. For example, the following triple has `/photos/123` as subject, `dc:title` as predicate, and the literal "Beautiful Sunset" as object:

```
   </photos/123> dc:title "Beautiful Sunset" .
```

A triple effectively relates its subject and object by its predicate: the document
`/photos/123`
has, as title, "Beautiful Sunset". Structured data in RDF is represented as a set of triples. The notation above is called N3 [N3]. URIs are written using angle brackets, literals are written in quotation marks, and compact URIs are written directly (with prefixes declared earlier).

All subjects and predicates are nodes, while objects can be nodes or literals. Nodes can be URIs, or they can be blank, in which case they are not addressable by other documents. Blank nodes, denoted `_:bnodename`, are particularly useful when expressing layered data without having to assign URIs to intermediate nodes.

4.1 Events and Contact Information

In Section **2.2 Publishing An Event**, Jo published an event without giving it a URI. The RDF triples extracted from her markup are:

```
_:blanknode0
   rdf:type cal:Vevent;
   cal:summary "last summer Barbecue";
   cal:dtstart "20070916T1600-0500" .
```

In Section **2.3 Publishing Contact Information**, Jo published contact information. The

RDFa is parsed to generate the following RDF triples:

```
<http://example.org/staff/jo>
   contact:fn "Jo Smith";
   contact:title "Web Hacker";
   contact:org <http://example.org>;
   contact:email <mailto:jo@example.org>.
```

4.2 Custom Vocabularies and Datatypes

The XHTML+RDFa in the **3.2 Qualifying Other Documents and Document Chunks** yields the following triples:

```
</user/markb/photo/23456> dc:title "Sunset in Nice" .

</user/markb/photo/34567> dc:title "W3C Meeting in Mandelieu" .
```

The more complete example, including licensing information, yields the following triples:

```
</user/markb/photo/23456>
    dc:title "Sunset in Nice" ;
    dc:creator "Mark Birbeck" ;
    cc:license <http://creativecommons.org/licenses/by-nc/3.0/> .

</user/markb/photo/34567>
    dc:title "W3C Meeting in Mandelieu" ;
    dc:creator "Steven Pemberton" ;
    cc:license <http://creativecommons.org/licenses/by/3.0/> .
```

The example that links a photo to the camera it was taken with in corresponds to the following triple:

```
</user/markb/photo/23456> shutr:takenWithCamera </user/markb/cameras#nikon_d200> .
```

while the complete camera descriptions yields:

```
<#nikon_d200>
  dc:title "Nikon D200" ;
  shutr:frameRate "3 pictures/second" .

<#canon_sd550>
  dc:title "Canon Powershot SD550" ;
  shutr:frameRate "5 pictures/second" .
```

Finally, `@datatype` as in **3.3 Data Types** indicates a datatype as follows:

```
</user/markb/photo/23456> dc:date "2007-05-12"^^xsd:date .
```

4.3 Layered Data and Subresources

The subresources example in **3.4 Layers of Structure — Subresources**, with photos annotated with the individuals depicted, correspond to RDF blank nodes as follows:

```
</user/markb/photo/23456>
    foaf:depicts _:span0 ;
    foaf:decpits _:span1 ;

_:span0
    foaf:firstname "Mark" ;
    foaf:lastname "Birbeck" ;
    foaf:mbox "mark@example.org" .

_:span1
    foaf:firstname "Steven" ;
    foaf:lastname "Pemberton" ;
    foaf:mbox "steven@example.org" .
```

Note specifically how the bnode, in this example, is both the object of a first triple, and the subject of a number of follow-up triples. This is specifically the point of the layered markup approach: to create an unnamed subresource.

4.4 Using `@src` on `img`, and the use of `@rev`

The use of `@src` on an image, as per **3.5 Using @src on img, and the use of @rev**, yields exactly the same triple as if `@src` were `@href`:

```
</user/markb> foaf:img </user/markb/profile_photo.jpg> .
```

`@rev` specifies a triple with the subject and object reversed:

```
</user/markb> foaf:img </user/markb/profile_photo.jpg> .

</user/markb/profile_photo.jpg> foaf:depicts </user/markb> .
```

4.5 Overriding `@href` and `@src`

Where `@resource` is present, as per **3.6 Overriding @href and @src**, the same triples are generated, with the value of `@resource` replacing the value of `@href`. Even though the `@href` points to `/user/markb/profile_photo_thumbnail.jpg`, the corresponding triple is:

```
</user/markb> foaf:img </user/markb/profile_photo.jpg> .
```

5 Case Studies

A number of RDFa Case Studies are under development and available at http://rdfa.info/rdfa-case-studies/.

6 Acknowledgments

This document is the work of the RDF-in-HTML Task Force, including (in alphabetical order) Ben Adida, Mark Birbeck, Jeremy Carroll, Michael Hausenblas, Shane McCarron, Steven Pemberton, Ralph Swick, and Elias Torres. This work would not have been possible without the help of the Semantic Deployment Working Group and its previous

incarnation, the Semantic Web Deployment and Best Practices Working Group, in particular chairs Guus Schreiber and David Wood, the XHTML2 Working Group, and Ivan Herman, head of the Semantic Web Activity. Earlier versions of this document were officially reviewed by Gary Ng and David Booth, both of whom provided insightful comments that significantly improved the work. Manu Sporny provided crucial feedback in the end stages.

7 Bibliography

FOAF
The Friend of a Friend (FOAF) Project (See http://www.foaf-project.org/.)

RDFHTML
RDF-in-HTML Task Force (See http://www.w3.org/2001/sw/BestPractices/HTML/.)

SWD-WG
Semantic Web Best Deployment Working Group (See http://www.w3.org/2006/07/SWD/.)

SWBPD-WG
Semantic Web Best Practices and Deployment Working Group (See http://www.w3.org/2001/sw/BestPractices/.)

ICAL-RDF
RDF Calendar Interest Group Note (See http://www.w3.org/TR/rdfcal/.)

VCARD-RDF
Representing vCard Objects in RDF/XML (See http://www.w3.org/TR/vcard-rdf.)

XHTML2-WG
XHTML2 Working Group (See http://www.w3.org/MarkUp/.)

XSD-DT
XML Schema Datatypes (See http://www.w3.org/TR/xmlschema-2/.)

RDFa in XHTML: Syntax

A collection of attributes and processing rules for extending XHTML to support RDF

W3C Editor's Draft 27 September 2007

This version:
http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070927
Latest version:
http://www.w3.org/TR/rdfa-syntax
Previous Editor's Draft:
http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070921
Diff from previous Editor's Draft:
rdfa-syntax-diff.html
Editors:
Ben Adida, Creative Commons ben@adida.net
Mark Birbeck, x-port.net Ltd. mark.birbeck@x-port.net
Shane McCarron, Applied Testing and Technology, Inc. shane@aptest.com
Steven Pemberton, CWI

This document is also available in these non-normative formats: PostScript version, PDF version, ZIP archive, and Gzip'd TAR archive.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

The modern Web is made up of an enormous number of documents that have been created using HTML. These documents contain significant amounts of structured data, which is largely unavailable to tools and applications. When publishers can express this data more completely, and when tools can read it, a new world of user functionality becomes available, letting users transfer structured data between applications and web sites, and allowing browsing applications to improve the user experience: an event on a web page can be directly imported into a user's desktop calendar; a license on a document can be detected so that users can be informed of their rights automatically; a photo's creator, camera setting information, resolution, location and topic can be published as easily as the original photo itself, enabling structured search and sharing.

RDFa is a syntax for expressing this structured data in XHTML. The rendered, hypertext data of XHTML is reused by the RDFa markup, so that publishers don't repeat themselves. The underlying abstract representation is RDF [RDF-PRIMER], which lets publishers build their own vocabulary, extend others, and evolve their vocabulary with maximal interoperability over time. The expressed structure is closely tied to the data, so that rendered data can be copied and pasted along with its relevant structure.

The rules for interpreting the data are generic, so that there is no need for different rules for different formats; this allows authors and publishers of data to define their own formats without having to update software, register formats via a central authority, or worry that two formats may interfere with each other.

This document is a detailed syntax specification for RDFa, aimed at:

- those looking to create an RDFa parser, and who therefore need a detailed description of the parsing rules;
- those looking to recommend the use of RDFa within their organisation, and who would like to create some guidelines for their users;
- anyone familiar with RDF, and who wants to understand more about what is happening 'under the hood', when an RDFa parser runs.

For those looking for an introduction to the use of RDFa and some real-world examples, please consult the RDFa Primer.

How to Read this Document

If you are already familiar with RDFa, and you want to examine the processing rules—perhaps to create a parser—then you'll find the Processing Model section of most interest. It contains an overview of each of the processing steps, followed by more detailed sections, one for each rule.

If you are not familiar with RDFa, but you *are* familiar with RDF, then you might find reading the Syntax Overview useful, before looking at the Processing Model since it gives a range of examples of XHTML mark-up that use RDFa. Seeing some examples first should make reading the processing rules easier.

If you are not familiar with RDF, then you might want to take a look at the section on RDF Terminology before trying to do too much with RDFa. Although RDFa is designed to be easy to author—and authors don't need to understand RDF to use it—anyone writing applications that *consume* RDFa will need to understand RDF. There is a lot of material on RDF on the web, and a growing range of tools that will support RDFa, so all we try to do in this document is provide enough background on RDF to make the goals of RDFa clearer.

And finally, if you are not familiar with either RDFa *or* RDF, and simply want to add RDFa to your documents, then you may find the RDFa Primer [RDFaPRIMER] to be a better introduction.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is an internal draft produced jointly by the Semantic Web Deployment Working Group [SWD-WG] and the XHTML 2 Working Group [XHTML2-WG]. Initial work on RDFa began in the XHTML 2 Working Group [XHTML2-WG].

This document has no official standing within the W3C. It is also a work in progress, which means it may change at any time, without warning, and you shouldn't rely on anything in this document.

Table of Contents

1. Motivation

This section is informative.

RDF/XML [RDF-SYNTAX] provides sufficient flexibility to represent all of the abstract concepts in RDF [RDF-CONCEPTS]. However, it presents two challenges; first it is difficult or impossible to validate documents that contain RDF/XML using XML Schemas or DTDs, which makes it difficult to import RDF/XML into other markup languages. Whilst newer schema languages such as RELAX NG [RELAXNG] do provide a way to validate documents that contain arbitrary RDF/XML, it will be a while before they gain wide support.

Second, even if one could add RDF/XML directly into an XML dialect like XHTML, there would be significant data duplication between the rendered data and the RDF/XML structured data. It would be far better to add RDF to a document without repeating the document's existing data. For example, an XHTML document that explicitly renders its author's name in the text—perhaps as a byline on a news site—should not need to repeat this name for the RDF expression of the same concept: it should be possible to supplement the existing markup in such a way that it can also be interpreted as RDF.

Third, as users often want to transfer structured data from one application to another, sometimes to or from a non-web-based application, it is highly beneficial to express the web data's structure "in context." The user experience could then be enhanced, for example by providing contextual information about specific rendered data, perhaps when the user "right-clicks" on an item of interest.

In the past, many attributes were 'hard-wired' directly into the markup language to represent specific concepts. For

example, in XHTML 1.1 [XHTML11] and HTML [HTML4] there is @cite; the attribute allows an author to add information to a document which is used to indicate the origin of a quote.

However, these 'hard-wired' attributes make it difficult to define a generic process for extracting metadata from any document since a parser would need to know about each of the special attributes. One motivation for RDFa has been to devise a means by which documents can be augmented with metadata in a general rather than hard-wired manner. This has been achieved by creating a fixed set of attributes and parsing rules, but allowing those attributes to contain properties from any of a number of the growing range of available RDF vocabularies. The *values* of those properties are in most cases the information that is already in an author's XHTML document.

RDFa alleviates the pressure on XML format authors to anticipate all the structural requirements users of their format might have, by outlining a new syntax for RDF that relies only on XML attributes. This specification deals specifically with the use of RDFa in XHTML, and defines an RDF mapping for a number of XHTML attributes, but RDFa can be easily imported into other XML-based markup languages.

2. Syntax Overview

This section is informative.

The following examples are intended to help readers who are not familiar with RDFa to quickly get a sense of how it works. For a more thorough introduction, please read the RDFa Primer [RDFaPRIMER].

2.1. The RDFa Attributes

RDFa in XHTML makes use of a number of XHTML attributes, as well as providing a few new ones. Attributes that already exist in XHTML will have the same meaning as in XHTML, although their syntax may be slightly modified. For example, in XHTML, @rel already defines the relationship between one document and another. However, in XHTML there is no clear way to add new values; RDFa sets out to explicitly solve this problem, and does so by allowing URIs as values. It also introduces the idea of 'compact URIs'—referred to as CURIEs in this document—which allow a full URI value to be expressed succinctly.

The XHTML attributes that are relevant are:

@rel,
: a whitespace separated list of CURIEs, used for expressing relationships between two resources (a 'predicate' in RDF)

@rev,
: a whitespace separated list of CURIEs, used for expressing reverse relationships between two resources (also a 'predicate')

@href,
: a URI for expressing the partner resource of a relationship (the 'object' in RDF)

@src,
: a URI for expressing the partner resource of a relationship when the resource is embedded (also an 'object')

The new—RDFa-specific—attributes are:

@about,
: a URIorCURIE, used for stating what the data is about (the 'subject' in RDF terminology)

@property,
: a whitespace separated list of CURIEs, used for expressing relationships between the subject and some literal text (also a 'predicate')

@resource,
: a URIorCURIE for expressing the partner resource of a relationship that is not intended to be 'clickable' (also an 'object')

@datatype,
: a CURIE representing a datatype, to express the datatype of a literal

@content,
: a string, for supplying alternative, machine-readable content for a literal.

@instanceof
: a whitespace separated list of CURIEs that indicate the RDF type(s) to associate with the subject.

For a normative definition of these attributes see the XHTML Metainformation Attributes Module.

2.2. Examples

As an XHTML author you will already be familiar with using `meta` and `link` to add additional information to your documents:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Page 7</title>
    <meta name="author" content="Mark Birbeck" />
    <link rel="prev" href="page6.html" />
    <link rel="next" href="page8.html" />
  </head>
  <body>...</body>
</html>
```

RDFa makes use of this concept, enhancing it with the ability to make use of other vocabularies by using compact URIs:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
>
  <head>
    <title>My home-page</title>
    <meta property="dc:creator" content="Mark Birbeck" />
    <link rel="foaf:workplaceHomepage" href="http://www.formsPlayer.com/" />
  </head>
  <body>...</body>
</html>
```

Although not widely used, XHTML already supports the use of @rel and @rev on the a element. This becomes more useful in RDFa with the addition of support for different vocabularies:

```
This document is licensed under a
<a xmlns:cc="http://creativecommons.org/licenses/"
  rel="cc:license"
  href="http://creativecommons.org/licenses/by/nc-nd/3.0/">
  Creative Commons License
</a>.
```

Not only can URLs in the document be re-used to provide metadata, but so can inline text:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
  >
  <head><title>Jo's Friends and Family Blog</title></head>
  <body>
    <p>
      I'm holding
      <span property="cal:summary">
        one last summer Barbecue
      </span>,
      on September 16th at 4pm.
    </p>
  </body>
</html>
```

If some displayed text is different to the actual 'value' it represents, more precise values can be added, which can optionally include datatypes:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  >
  <head><title>Jo's Friends and Family Blog</title></head>
  <body>
    <p>
      I'm holding
      <span property="cal:summary">
        one last summer Barbecue
      </span>,
      on
      <span property="cal:dtstart" content="20070916T1600-0500"
            datatype="xsd:datetime">
        September 16th at 4pm
      </span>.
    </p>
  </body>
</html>
```

In many cases a block of mark-up will contain a number of properties that relate to the same item; it's possible with RDFa to indicate the type of that item:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:cal="http://www.w3.org/2002/12/cal/ical#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  >
  <head><title>Jo's Friends and Family Blog</title></head>
  <body>
    <p instanceof="cal:Vevent">
      I'm holding
      <span property="cal:summary">
        one last summer Barbecue
      </span>,
      on
      <span property="cal:dtstart" content="20070916T1600-0500"
            datatype="xsd:datetime">
        September 16th at 4pm
      </span>.
    </p>
  </body>
</html>
```

The metadata features available in XHTML only allow information to be expressed about the document itself. RDFa provides a means of referring to other documents and resources:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:bib="http://example.org/"
>
  <head>
    <title>Books by Marco Pierre White</title>
  </head>
  <body>
    I think
    <span about="urn:ISBN:0091808189" instanceof="bib:book">
      White's book 'Canteen Cuisine'
    </span>
    is well worth getting since although it's quite advanced stuff, he
    makes it pretty easy to follow. You might also like his
    <span about="urn:ISBN:1596913614" instanceof="bib:book">
```

```
        autobiography
      </span>.
    </body>
  </html>
```

3. RDF Terminology

This section is informative.

The previous section gave examples of typical mark-up in order to illustrate what RDFa in XHTML looks like. But what RDFa in XHTML *represents* is RDF. In order to author RDFa in XHTML you do not need to understand RDF, although it would certainly help. However, if you are building a system that consumes the RDF output of an RDFa in XHTML document you will almost certainly need to understand RDF. In this section we introduce the basic concepts, and terminology of RDF. For a more thorough explanation of RDF, please refer to the RDF Concepts document [RDF-CONCEPTS] and tne RDF Sytax Document [RDF-SYNTAX].

3.1. Statements

The structured data that RDFa provides access to is a collection of *statements*. A statement is a basic unit of information that has been constructed in a specific format to make it easier to process. In turn, by breaking large sets of information down into a collection of statements, even very complex metadata can be processed using simple rules.

To illustrate, suppose we have the following set of facts:

```
Albert was born on March 14, 1879, in Germany. There is a picture of him at
the web address, http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

This would be quite difficult for a machine to interpret, and it is certainly not in a format that could be passed from one data application to another. However, if we convert the information to a set of statements it begins to be more manageable. The same information could therefore be represented by the following shorter 'statements':

```
Albert was born on March 14, 1879.
Albert was born in Germany.
Albert has a picture at
  http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

3.2. Triples

To make this information machine-processable, RDF defines a structure for these statements. A statement is formally called a *triple*, meaning that it is made up of three components. The first is the *subject* of the triple, and is what we are making our statements about. In these examples the subject is always 'Albert'.

The second part of a triple is the property of the subject that we want to define. In the examples here, the properties would be 'was born on', 'was born in', and 'has a picture at'. These are more usually called *predicates* in RDF.

The final part of a triple is called the *object*. In the examples here the object values are 'March 14, 1879', 'Germany', and 'http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg'.

3.3. URI references

Breaking complex information into manageable units helps us be specific about our data, but there is still some ambiguity. For example, which 'Albert' are we talking about? If another system has more facts about 'Albert', how could we know whether they are about the same person, and so add them to the list of things we know about that person? If we wanted to find people born in Germany, how could we know that the predicate 'was born in' has the same purpose as the predicate 'birthplace' that exists in some other system? RDF solves this problem by replacing our vague terms with *URI references*.

URIs are most commonly used to identify web pages, but RDF makes use of them as a way to provide unique identifiers for concepts. For example, we could identify the subject of all of our statements by using the DBPedia [?ref] URI for Albert Einstein, rather than 'Albert':

```
<http://dbpedia.org/resource/Albert_Einstein>
   has the name
   Albert Einstein.
<http://dbpedia.org/resource/Albert_Einstein>
   was born on
   March 14, 1879.
<http://dbpedia.org/resource/Albert_Einstein>
   was born in
   Germany.
<http://dbpedia.org/resource/Albert_Einstein>
   has a picture at
   http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg.
```

URI references are also used to uniquely identify the objects in metadata statements. The picture of Einstein is already a URI, but we can also use one to uniquely identify the country Germany (note that we put literals into quotes to distinguish them from URIs):

```
<http://dbpedia.org/resource/Albert_Einstein>
   has the name
   "Albert Einstein".
<http://dbpedia.org/resource/Albert_Einstein>
   was born on
   "March 14, 1879".
<http://dbpedia.org/resource/Albert_Einstein>
   was born in
   <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
   has a picture at
   <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

URI references are also used to ensure that predicates are unambiguous; now we can be sure that 'birthplace', 'place of birth', 'place de nee' [???] and so on, all mean the same thing:

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/name>
  "Albert Einstein".
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/dateOfBirth>
  "March 14, 1879".
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/birthPlace>
  <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/depiction>
  <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

3.4. Plain literals

Although URI resources are always used for subjects and predicates, the object part of a triple can be either a URI or a *literal*. In the example triples, Einstein's name is represented by a *plain literal*, which means that it is a basic string with no type or language information:

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://xmlns.com/foaf/0.1/name> "Albert Einstein".
```

3.5. Typed literals

Some literals, such as dates and numbers, have very specific meanings, so RDF provides a mechanism for indicating the type of a literal. A *typed literal* is indicated by attaching a URI to the end of a plain literal, and this URI indicates the literal's datatype. This URI is usually based on datatypes defined in the XML Schema Datatypes specification [XMLSCHEMA]. The following syntax would be used to unambiguously express Einstein's date of birth as a literal of type < code>http://www.w3.org/2001/XMLSchema#date:

```
<http://dbpedia.org/resource/Albert_Einstein>
  <http://dbpedia.org/property/dateOfBirth>
  "1879-03-14"^^<http://www.w3.org/2001/XMLSchema#date>.
```

3.6. N-Triples

RDF itself does not have one set way to express triples, since the key ideas of RDF are the triple and the use of URIs, and *not* any particular syntax. However, there are a number of mechanisms, such as RDF/XML, N-Triples [N-TRIPLES], and of course RDFa. Most discussions of RDF make use of the *N-Triple* syntax to explain their ideas, since it is quite compact. The examples we have just seen are already using this syntax, and we'll continue to use it throughout this document when we need to talk about the RDF that could be generated from some RDFa. There is one small change that we make to N-Triples, which is to allow long URIs to be abbreviated by using a URI mapping. This is indicated by expressing a compact URI as follows:

```
<http://dbpedia.org/resource/Albert_Einstein>
  foaf:name "Albert Einstein" .
<http://dbpedia.org/resource/Albert_Einstein>
  p:dateOfBirth "1879-03-14"^^xsd:date .
<http://dbpedia.org/resource/Albert_Einstein>
  p:birthPlace <http://dbpedia.org/resource/Germany>.
<http://dbpedia.org/resource/Albert_Einstein>
  foaf:depiction <http://en.wikipedia.org/wiki/Image:Albert_Einstein_Head.jpg>.
```

Here 'p:' has been mapped to the URI for DBPedia, and 'foaf:' has been mapped to the URI for the 'Friend of a Friend' taxonomy.

Note that this is merely a way to make examples more compact and the actual triples generated would always use the full URIs.

When writing examples, you will often see the following URI:

```
<>
```

This indicates the 'current document', i.e., the document being processed. In reality there would always be a full URI, but this serves to make examples more compact.

3.7. Graphs

A collection of triples is called a *graph*.

For more information on the concepts described above, see [RDF-CONCEPTS]. RDFa additionally defines the following terms:

3.8. Compact URIs

In order to allow for the compact expression of RDF statements, RDFa allows the contraction of all [URI reference]s into a form called a 'compact URI', or [CURIE]. Until recently QNames [QNames] have been the most common way to abbreviate URIs, but there is a well-known limitation that the syntax for QNames does not allow all possible [URI reference]s to be expressed. CURIEs have been specificially designed to look like QNames, but at the same time to get around their syntactic limitations.

Note that CURIEs are only used in the mark-up and N-Triples examples, and will never appear in the generated [triple]s, which will always use [URI reference]s.

3.9. A description of RDFa in RDF terms

The following is a brief description of RDFa in terms of the RDF terminology introduced here. It may be useful to readers with an RDF background:

The aim of RDFa is to allow a single [RDF graph]s to be carried in XML documents of any type, although this specification deals only with RDFa in XHTML. An [RDF graph] comprises [node]s linked by relationships. The basic unit of a graph is a [triple], in which a subject [node] is linked to an object [node] via a [predicate]. The subject [node] is always either an [URI reference] or a [blank node], the predicate is *always* an [URI reference], and the object of a statement can be an [URI reference], a [literal], or a [blank node].

In RDFa, a subject [URI reference] is generally indicated using @about, and predicates are represented using one of @property, @instanceof, @rel, or @rev. Objects which are [URI reference]s are represented using @href, @resource or @src, whilst objects that are [literal]s are represented either with @content (with an optional [datatype] expressed using @datatype), or the content of the element in question.

4. Conformance Requirements

This section is normative.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Note that all examples in this document are informative, and are not meant to be interpreted as normative requirements.

4.1. Document Conformance

A strictly conforming XHTML+RDFa document is a document that requires only the facilities described as mandatory in this specification. Such a document MUST meet all the following criteria:

1. The document must conform to the constraints expressed in the schemas in Appendix B - XHTML+RDFa Document Type Definition.
2. The local part of the root element of the document must be `html`.
3. The start tag of the root element of the document must explicitly contain an `xmlns` declaration for the XHTML namespace [XMLNS]. The namespace URI for XHTML is defined to be `http://www.w3.org/1999/xhtml`.

   Sample root element

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
   ```

4. There MUST be a DOCTYPE declaration in the document prior to the root element. If present, the public identifier included in the DOCTYPE declaration must reference the DTD found in Appendix B - XHTML+RDFa Document Type Definition using its Public Identifier. The system identifier may be modified appropriately.

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
       "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
   ```

Example of an XHTML+RDFa 1.0 document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
    "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://example.org/">example.org</a>.</p>
  </body>
</html>
```

Note that in this example, the XML declaration is included. An XML declaration like the one above is not required in all XML documents. XHTML document authors should use XML declarations in all their documents. XHTML document authors must use an XML declaration when the character encoding of the document is other than the default UTF-8 or UTF-16 and no encoding is specified by a higher-level protocol.

XHTML 1.1 documents SHOULD be labeled with the Internet Media Type "application/xhtml+xml" as defined in [RFC3236]. For further information on using media types with XHTML, see the informative note [XHTMLMIME].

4.2. User Agent Conformance

A conforming user agent MUST support all of the features required in this specification. A conforming user agent must also support the User Agent conformance requirements as defined in XHTML Modularization [XHTMLMOD] section on "XHTML Family User Agent Conformance".

4.3. RDFa Processor Conformance

A conforming RDFa Processor MUST make available to a consuming application a single RDF [graph] containing all possible triples generated by using the rules in the Processing Model section. This is the 'default [graph]'.

A conforming RDFa Processor MAY make available additional triples that have been generated using rules not described here, but these triples MUST be made available in one or more additional RDF [graph]s, and not in the default [graph].

A conforming RDFa Processor MUST process whitespace according to the rules of [CSS2]. *Note that this same requirement is imposed upon conforming User Agents via [XHTMLMOD].*

Test Suite

We have a test suite - we should likely reference it here, but I need to find the exact way to do that. -Shane

Assertion Annotation

All the assertions in this spec need to be annotated with the appropriate markup (must, should, etc.).

5. Processing Model

This section is normative.

This section looks at a generic set of processing rules for creating a set of triples that represent the structured data present in an XHTML+RDFa document. Processing need not follow the DOM traversal technique outlined here, although the effect of following some other manner of processing must be the same as if the processing outlined here were

followed. The processing model is explained using the idea of DOM traversal which makes it easier to describe (particularly in relation to the 'evaluation context').

5.1. Overview

Parsing a document for RDFa triples is carried out by starting at the root element of the document, and visiting each of its child elements in turn, applying processing rules. Processing is recursive in that for each child element the processor also visits each of *its* child elements, and applies the same processing rules.

As processing continues, rules are applied which will either generate triples, or change the [evaluation context] information which will be used in subsequent processing. Some of the rules will be determined by the host language—in this case XHTML—and some of the rules will be part of RDFa.

Note that we don't say anything about what should happen to the triples generated, or whether more triples might be generated during processing than are outlined here. However, to be conformant, an RDFa processor needs to act as if at a minimum the rules in this section are applied.

5.2. Evaluation Context

During processing, each rule is applied within an 'evaluation context'. Rules may further modify this evaluation context, or create triples that can be established by making use of this evaluation context. The context itself consists of the following pieces of information:

- The [base]. This will usually be the URL of the document being processed, but it could be some other URL, set by some other mechanism, such as the XHTML `base` element. The important thing is that it establishes a URL against which relative paths can be evaluated.
- The [current resource]. The initial value will be the same as the initial value of `base`, but it will usually change during the course of processing.
- The [current element identifier]. This is an identifier for the current element, which will not always be the same as [current resource].
- A list of current in-scope [URI mappings].
- The [language]. Note that there is no default language.

5.3. Processing

Processing would normally begin after the document to be parsed has been completely loaded. However, there is no requirement for this to be the case, and it is certainly possible to use a stream-based approach, such as SAX [http://en.wikipedia.org/wiki/SAX] to extract the RDFa information. However, if some approach other than the DOM traversal technique defined here is used, it is important to ensure that any `meta` or `link` elements processed in the `head` of the document honour any occurrences of `base` which may appear *after* those elements. (In other words, XHTML processing rules must still be applied, even if document processing takes place in a non-HTML environment such as a search indexer.)

At the beginning of processing, the [current evaluation context] is initialised as follows:

- the [base] is set to either the URL of the document or the value specified in the `base` element, if present;
- the [current resource] is set to the [base] value;
- the [current element identifier] is cleared;
- the [list of URI mappings] is cleared;
- the [language] is cleared.

Processing then begins with the root element, and all nodes in the tree are processed according to the following rules, depth-first:

1. Any changes to the [current evaluation context] are made first:
    - the [current element] is parsed for [URI mappings] and these are added to the [list of URI mappings]. Note that a [URI mapping] will simply overwrite any current mapping in the list that has the same name;

These mappings are provided by @xmlns. The value to be mapped is set by the XML namespace prefix, and the value to map is the value of the attribute—a URI. Note that the URI is not processed in any way; in particular if it is a relative path it is not resolved against the [current base]. Authors are advised to follow best practice for using namespaces, which includes not using relative paths. (See [xyz].)

2. the [current element] is parsed for any language information, and [language] is set in the [current evaluation context];

   Language information can be provided using the general-purpose XML attribute @xml:lang.

3. the [current element] is parsed for any subject information, and it is used to set the [current resource] value, in the [current evaluation context];

   The [current resource] can be set using @about. Note that the final value of the [current resource] is an absolute IRI, which means that if @about contains a relative path the value must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986. The value can also be provided by a CURIE, and is processed as defined in CURIE Syntax Definition. Note that since this attribute can take both URIs and CURIEs, the latter will have been expressed using the [safe CURIE] syntax.

4. the [current element identifier] is set;

   The [current element identifier] is set to the value of @about, if present, *or* the value of @resource, if present, *or* the value of @href, if present, *or* the value of @src, if present, *or* a [blank node]. Note that this means that the value of the [current element identfier] will not be the same as the [current resource], since in the absence of an attribute to set its value it will be set to a [blank node], whilst [current resource] inherits from its context.
   Note that the final value of [current element identifier] is an absolute IRI, so any relative paths must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986. The value might also have been provided by a CURIE, and if so, it is processed as defined in CURIE Syntax Definition.

5. the [recurse] flag is set to true;

   Processing will generally continue recursively through the entire tree of nodes available. However, if an author indicates that some branch of the tree should be treated as an XML literal, no further processing should take place on that branch. This flag is used to inhibit this processing.

6. the [chaining] flag is set to false;

   If a collection of statements is contained by a [URI reference] then this may become the subject of further statements.

7. Once the [current evaluation context] has been set, object resolution is carried out, as follows:
   - the [current object resource] is established;

     Since only one [current object resource] is set per element then some attributes will have a higher priority than others. The highest priority is given to @resource. If there is no @resource then @href is used, and if that is not present, @src is used. If none of these are present then a unique identifier or [blank node] is created. Note that the final value of the [current object resource] is an absolute URI or a [blank node], which means that if any of these attributes contain relative paths they must be normalised against [base] in the [current evaluation context], using the algorithm defined in RFC 3986. Note also that since @resource can take both URIs and CURIEs, the latter will have been expressed using the [safe CURIE] syntax.

8. the [current object literal] is established;

   The [current object literal] will be set as a [plain literal] if @content is present, *or* the body of the [current element] contains only text (i.e., there are no child elements), *or* the body of the [current

element] *does* have child elements but @datatype has an empty value. Additionally, if there is a value for [current language] then the value of the [plain literal] should include this language information, as described in [RDFCONCEPTS]. The actual literal is either the value of @content (if present) *or* a string created by concatenating the text content of each of the child elements of the [current element] in document order, and then normalising white-space according to [WHITESPACERULES].

Whitespace normalising

So far we defer to CSS2, but I think we should copy the prose into here, so that it's clearer.

The [current object literal] will be set as a [typed literal] if @datatype is present, and does not have an empty value. The actual literal is either the value of @content (if present) *or* a string created by concatenating the inner content of each of the children in turn, of the [current element]. The final string includes the datatype, as described in [RDFCONCEPTS].

The [current object literal] will be set as an [XML literal] if the [current element] has child elements, and @datatype is not present, or is set to `rdf:XMLLiteral`. The value of the [XML literal] is a string created from the inner content of the [current element], i.e., not including the element itself, with the datatype of `rdf:XMLLiteral`.

9. Once object resolution is complete the processor will have two objects, one a resource and the other a literal. These objects can now be used to create triples with the [current resource], depending on the presence of other attributes. This is achieved using the following processing steps:
    - Predicates for the [current object literal] are established;

      Predicates for the [current object literal] can be set by using @property. If present, the attribute must contain one or more [curie]s, each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
      subject

      [current resource]

      predicate

      expanded value from the curie

      object

      [current object literal]

      Note that *literal*may include language and datatype information as discussed in the section on object resolution. Once the triple has been created, the [recurse] flag is set to false.
10. predicates for the [current object resource] are established. If any triples are generated then the [chaining] flag is set to `true`.;

    Predicates for the [current object resource] can be set by using one or both of the @rel and @rev attributes. If present, @rel must contain one or more [CURIE]s, each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
    subject

    [current resource]

    predicate

    expanded value from the curie

    object

    [current object resource]

    If present, @rev must contain one or more [CURIE]s, each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
    subject

    [current object resource]

    predicate

    expanded value from the curie

    object

    [current resource]

11. Type values for the [current element identifier] are established. If any triples are generated then the [chaining] flag is set to `true`.;

    One or more 'types' for the [current element identifier] can be set by using @instanceof. If present, the attribute must contain one or more [curie]s, each of which is converted to an absolute URI using CURIE processing rules, and then used to generate a triple as follows:
    subject

    [current element identifier]

    predicate

    http://www.w3.org/1999/02/22-rdf-syntax-ns#type

    object

    expanded value from the curie

12. If the [chaining] flag is set to `true` then the [current resource] is set to the value of the [current object resource], and the [chaining] flag is set to `false`.
13. If the [recurse] flag is true, the [current evaluation context] is pushed onto a stack, and all nodes that are children of the [current element] are processed using the rules described here. Once all of the children have been processed then the [current evaluation context] is popped back off the stack.

6. RDFa in detail

This section is normative.

This section provides an in-depth examination of the processing steps described in the previous section. It also includes examples which may help clarify some of the steps involved.

@instanceof situation

This section still needs the detail on whether @instanceof should use @about if it is present, or use the subject from chaining.

NOTE: There isn't quite enough detail on chaining yet.

In the following examples, for brevity assume that the following namespace prefixes have been defined:

cc:	http://creativecommons.org/ns#
dc:	http://purl.org/dc/elements/1.1/
ex:	http://example.org/
foaf:	http://xmlns.com/foaf/0.1/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
p:	http://dbpedia.org/property/
rdfa:	http://www.w3.org/ns/rdfa/
svg:	http://www.w3.org/2000/svg
xh11:	http://www.w3.org/1999/xhtml
xsd:	http://www.w3.org/2001/XMLSchema#
biblio:	http://example.org/biblio/0.1
taxo:	http://purl.org/rss/1.0/modules/taxonomy/

The key to processing is that a triple is generated whenever a predicate/object combination is detected. The actual triple generated will include a subject that may have been set previously, so this is tracked in the [current evaluation context] and is called the [current resource]. Since the subject will default to the current document if it hasn't been set explicitly, then a predicate/object combination is always enough to generate one or more triples.

The attributes for setting a predicate are @rel, @rev and @property, whilst the attributes for setting an object are @resource, @href, @content, and @src.

Note that there are actually two special cases—the use of @instanceof to set type information, and @rel or @rev appearing on an element on its own. Both of these cases are discussed in more details below.

6.1. Changing the evaluation context

6.1.1. Setting the [current resource]

When triples are created they will always be in relation to the [current resource]. When parsing begins the [current resource] will be the URI of the document being parsed, or a value as set by `base`.

Metadata about the document itself is usually placed in the `head`:

```
<html>
  <head>
    <title>Jo's Friends and Family Blog</title>
    <link rel="foaf:primaryTopic" href="#bbq" />
    <meta property="dc:creator" content="Jo" />
  </head>
  <body>
    ...
  </body>
</html>
```

although it is possible for the data to appear elsewhere:

```
<html>
  <head>
    <title>Jo's Blog</title>
  </head>
  <body>
    <h1><span property="dc:creator">Jo</span>'s blog</h1>
    <p>
      Welcome to my blog.
    </p>
  </body>
</html>
```

The value of `base` may change the initial value of [current resource]:

```
<html>
  <head>
    <title>Jo's Friends and Family Blog</title>
    <link rel="foaf:primaryTopic" href="#bbq" />
    <meta property="dc:creator" content="Jo" />
    <base href="http://www.example.org/jo/blog" />
  </head>
  <body>
    ...
  </body>
</html>
```

As processing progresses, any @about attributes will change the [current resource]. The value of @about is a URI or a CURIE. If it is a relative URI then it needs to be resolved against the current [base] value. In this mark-up the properties `cal:summary` and `cal:dtstart` become part of the 'event' object, rather than referring to the document:

```
<html>
  <head>
    <title>Jo's Friends and Family Blog</title>
    <link rel="foaf:primaryTopic" href="#bbq" />
    <meta property="dc:creator" content="Jo" />
  </head>
  <body>
    <p about="#bbq" instanceof="cal:Vevent">
      I'm holding
      <span property="cal:summary">
        one last summer Barbecue
      </span>,
      on
      <span property="cal:dtstart" content="20070916T1600-0500"
            datatype="xsd:datetime">
        September 16th at 4pm
      </span>.
    </p>
  </body>
</html>
```

Other kinds of resources can be used to set the [current resource], not just references to web-pages:

```
Daniel knows
<a about="mailto:daniel.brickley@bristol.ac.uk"
  rel="foaf:knows" href="mailto:libby.miller@bristol.ac.uk">Libby</a>.

Libby knows
<a about="mailto:libby.miller@bristol.ac.uk"
  rel="foaf:knows" href="mailto:ian.sealy@bristol.ac.uk">Ian</a>.
```

```
<div about="photo1.jpg">
  <span class="attribution-line">this photo was taken by
    <span property="dc:creator">Mark Birbeck</span>
  </span>
</div>
```

6.2. Object resolution

There are two types of object, [URI resource]s and [literal]s.

A [literal] object can be set by using @property to express a [predicate], and then using either @content, or the inline text of the element that @property is on.

A [URI resource] object can be set using one of @rel or @rev to express a [predicate], and then using one of @href, @resource or @src to provide the object.

6.2.1. Literal object resolution

An [object literal] will be generated when @property is present. @property provides the predicate, and the following sections describe how the actual literal to be generated is determined.

6.2.1.1. Plain Literals

@content can be used to indicate a [plain literal], as follows:

```
<meta about="http://internet-apps.blogspot.com/"
      property="dc:creator" content="Mark Birbeck" />
```

The [plain literal] can also be specified by using the content of the element:

```
<span about="http://internet-apps.blogspot.com/"
      property="dc:creator">Mark Birbeck</span>
```

Both of these examples give the following triple:

```
  <http://internet-apps.blogspot.com/>
    dc:creator "Mark Birbeck" .
```

The value of @content attribute is given precedence over any element content, so the following would give exactly the same triple:

```
<span about="http://internet-apps.blogspot.com/"
      property="dc:creator" content="Mark Birbeck">John Doe</span>
```

6.2.1.1.1. Language Tags

RDF allows [plain literal]s to have a language tag, as illustrated by the following example from [RDFTESTS-RDFMS-XMLLANG-TEST006]:

```
<http://example.org/node>
  <http://example.org/property> "chat"@fr .
```

In RDFa the XML language attribute @xml:lang is used to add this information, whether the plain literal is designated by @content, or by the inline text of the element:

```
<meta about="http://example.org/node"
  property="ex:property" xml:lang="fr" content="chat" />
```

Note that the language value can be inherited as defined in [XML-LANG], so the following syntax will give the same triple as above:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ex="http://www.example.com/ns/" xml:lang="fr">
  <head>
    <title xml:lang="en">Example</title>
    <meta about="http://example.org/node"
      property="ex:property" content="chat" />
  </head>
  ...
</html>
```

6.2.1.2. Typed literals

[Literal]s can be given a data type using @datatype.

This can be represented in RDFa as follows:

```
<span property="cal:dtstart" content="20070916T1600-0500"
      datatype="xsd:datetime">
```

```
    September 16th at 4pm
  </span>.
```

The triples that this mark-up generates include the datatype after the literal:

```
  <>
    cal:dtstart "20070916T1600-0500"^^xsd:datetime .
```

6.2.1.3. XML Literals

XML documents cannot contain XML mark-up in their attributes, which means it is not possible to represent XML within @content (the following would cause an XML parser to generate an error):

```
  <head about="">
    <meta property="dc:title"
      content="E = mc<sup>2</sup>: The Most Urgent Problem of Our Time" />
  </head>
```

It does not help to escape the content, since the output would simply be a string of text containing numerous ampersands:

```
  <>
  dc:title "E = mc&lt;sup&gt;2&amp;lt;/sup&gt;: The Most Urgent Problem of Our Time" .
```

RDFa therefore supports the use of normal mark-up to express XML literals, by using @datatype:

```
      <h2 property="dc:title" datatype="rdf:XMLLiteral">
        E = mc<sup>2</sup>: The Most Urgent Problem of Our Time
      </h2>
```

This would generate the following triple, with the XML preserved in the literal:

```
      <>
        dc:title "E = mc<sup>2</sup>: The Most Urgent Problem of Our Time"^^rdf:
  XMLLiteral .
```

Note that this requires that a URI mapping for the prefix `rdf` has been defined. To make authoring easier, if there are child elements and no @datatype attribute, then the effect is the same as if @datatype have been explicitly set to `rdf:XMLLiteral`:

```
      <h2 property="dc:title">
        E = mc<sup>2</sup>: The Most Urgent Problem of Our Time
      </h2>
```

In the examples we've been using the `sup` element is actually part of the meaning of the literal, but there will be situations where the extra mark-up means nothing, and can therefore be ignored. In this situation an empty @datatype value can be used to override the XML literal behaviour:

```
<p>You searched for <strong>Einstein</strong>:</p>
<p about="http://dbpedia.org/resource/Albert_Einstein">
  <span property="foaf:name" datatype="">Albert <strong>Einstein</strong></span>
  (March 14, 1879 â€“ April 18, 1955) was a German-born theoretical physicist.
</p>
```

Although the rendering of this page has highlighted the term the user searched for, setting @datatype to nothing ensures that the data is interpreted as a plain literal, giving the following triples:

```
<http://dbpedia.org/resource/Albert_Einstein>
  foaf:name "Albert Einstein" .
```

Note that the value of this [XML Literal] is the exclusive canonicalization of the RDFa element's value.

Although the RDFa processing model requires visiting each node in the tree, if the processor meets an [XML literal] then it MUST NOT process any further down the tree. This is to prevent triples being generated from mark-up that is not actually in the hierarchy. For example, we might want to set the title of something to some XHTML that includes RDFa:

```
<h2 property="dc:title">
  Example 3: <span about="#bbq" instanceof="cal:Vevent">...</span>
</h2>
```

This does effectively mean that the presence of @property without @content will inhibit any further processing, so authors should watch out for stray attributes, especially if they find that they are getting fewer triples than they had expected.

6.2.2. URI object resolution

One or more [URI object]s are needed when @rel or @rev is present. Each attribute will cause triples to be generated when used with @href, @resource or @src.

@rel and @rev are essentially the inverse of each other; whilst @rel establishes a relationship between the [current resource] as subject, and the [object resource] as the object, @rev does the exact opposite, and uses the [object resource] as the subject, and the [current resource] as the object.

6.2.2.1. Using @resource to set the object

RDFa provides the @resource attribute as a way to set the object of statements. This is particularly useful when referring to resources that are not themselves navigable links:

```
<html>
  <head>
    <title>On Crime and Punishment</title>
  </head>
  <body>
    <blockquote about="#q1" rel="dc:source" resource="urn:isbn:0140449132" >
      <p id="q1">
        Rodion Romanovitch! My dear friend! If you go on in this way
        you will go mad, I am positive! Drink, pray, if only a few drops!
      </p>
    </blockquote>
  </body>
</html>
```

The `blockquote` element generates the following triple:

```
<http://www.example.com/candp.xhtml#q1>
  <http://purl.org/dc/elements/1.1/source> <urn:isbn:0140449132>.
```

6.2.2.2. Using @href

If no @resource is present, then @href can be used to set the object.

When a triple predicate has been expressed using @rel, the @href on the [RDFa statement]'s element is used to indicate the object as a [URI reference]. Its type, just like that of @about, is a URI:

```
<link about="mailto:daniel.brickley@bristol.ac.uk"
      rel="foaf:knows" href="mailto:libby.miller@bristol.ac.uk" />
```

It's also possible to use both @rel and @rev at the same time on an element. This is particularly useful when two things stand in two different relationships with each, for example when a picture is taken *by* Mark, but that picture also *depicts* him:

```
<p>This photo was taken by
<a about="photo1.jpg" rel="dc:creator" rev="foaf:img"
   href="http://www.blogger.com/profile/1109404">Mark Birbeck</a>.</p>
```

which then yields two triples:

```
<photo1.jpg>
  dc:creator <http://www.blogger.com/profile/1109404> .
<http://www.blogger.com/profile/1109404>
  foaf:img <photo1.jpg> .
```

6.2.2.3. Using @about to set the subject

```
This photo, entitled
<span about="photo1.jpg" property="dc:title">Portrait of Mark</span>
was taken by
<a about="photo1.jpg" rel="dc:creator" rev="foaf:img"
  href="http://www.blogger.com/profile/1109404">Mark himself</a>.
```

The value of @about sets the subject for any nested triples which means that the same triples can be expressed using this, more compact, syntax:

```
  <div about="photo1.jpg">
    This photo, entitled
    <span property="dc:title">Portrait of Mark</span>
    was taken by
    <a rel="dc:creator" rev="foaf:img"
      href="http://www.blogger.com/profile/1109404">Mark himself</a>.
  </div>
```

6.2.2.4. Using a blank node to set the object

When a triple predicate has been expressed using @rel, and no @href, @src, or @resource exists on the same [RDFa element], then the CURIE represented by this element is used as the object. This CURIE is affected by @about, but if none is present the object is a [blank node] (blank nodes are discussed further in @@@section bnode [REF]@@@). In all cases, the subject resolution for child elements is affected: where they do not override the subject, their subject is this same CURIE here resolved as the object.

Consider, for example, a simple fragment of XHTML for describing the creator of a web page, with further information about the creator, including his name and email address:

```
<div rel="dc:creator">
  <span property="foaf:name">Ben Adida</span>
  (<a property="foaf:mbox" href="mailto:ben@adida.net">ben@adida.net</a>)
</div>
```

The above yields the following triples:

```
<>
   dc:creator _:div0 .

_:div0
   foaf:name "Ben Adida" .
_:div0
   foaf:mbox <mailto:ben@adida.net> .
```

6.2.2.4.1. Referencing Blank Nodes

To establish relationships between [blank node]s, the [unique anonymous ID] must be set explicity using a CURIE [blank node] as subject or object. For example, if our desired output is the following [triple]s:

```
_:a
  foaf:mbox <mailto:daniel.brickley@bristol.ac.uk> .
_:b
  foaf:mbox <mailto:libby.miller@bristol.ac.uk> .
_:a
  foaf:knows _:b .
```

we could use the following XHTML:

```
<link about="[_:a]" rel="foaf:mbox"
  href="mailto:daniel.brickley@bristol.ac.uk" />
<link about="[_:b]" rel="foaf:mbox"
  href="mailto:libby.miller@bristol.ac.uk" />
<link about="[_:a]" rel="foaf:knows"
  href="[_:b]" />
```

or, alternatively, if we wish to partly render the information in XHTML:

```
<div about="[_:a]">
  DanBri can be reached via
  <a rel="foaf:mbox"
    href="mailto:daniel.brickley@bristol.ac.uk">
    email
  </a>.
  <span rel="foaf:knows" resource="[_:b]">He knows Libby.</span>
</div>

<div about="[_:b]">
  Libby can be reached via
  <a rel="foaf:mbox"
    href="mailto:libby.miller@bristol.ac.uk">
    email
  </a>
</div>
```

7. CURIE Syntax Definition

This section is normative.

Note that this syntax definition will ultimately be defined in an external document [CURIE].

The key component of RDF is the URI, but they are usually long and unwieldy. RDFa therefore supports a mechanism by which URIs can be abbreviated, called 'compact URIs' or simply, CURIEs.

A CURIE is comprised of two components, a *prefix* which maps to a URI, and a *reference*. The prefix is separated from the reference by a colon (`:`). It is possible to omit the prefix, and make use of the default prefix. It is also possible to omit both the prefix *and* the colon, leaving just a *reference*.

```
curie       :=   [ [ prefix ] ':' ] reference

prefix      :=   NCName

reference   :=   irelative-ref (as defined in [IRI])
```

In some situations an attribute will allow either a CURIE, or a normal IRI. Since it is difficult to distinguish between CURIEs and IRIs, RDFa adds the notion of a [safe CURIE]. The syntax is simply to surround the CURIE with square brackets:

```
safe_curie  :=   '[' curie ']'
```

NOTE: The following language-independent prose will be removed shortly, once we have finalised this.

To evaluate CURIEs during processing the following context needs to be set:

- a set of mappings from prefixes to URIs;

  The prefix mappings are provided by the current in-scope namespace declarations of the [current element] during parsing.

- a mapping to use with the default prefix (for example, `:p`);

  The mapping to use with the default prefix is the current default namespace.

- a mapping to use when there is no prefix (for example, p);

  The mapping to use when there is no prefix is `http://www.w3.org/1999/xhtml#`.

- a mapping to use with the '_' prefix, which is used to generate unique identifiers (for example, `_:p`).

  the mapping to use with the '_' prefix is not explicitly stated, but should be chosen by the processor to ensure that there is no possibility of collision with other documents.

clarify the 'no prefix' situation

The advantage of setting the 'no prefix' mapping to the XHTML namespace is that we no longer need a preprocessing step to handle XHTML link types, such as a `next`. However, this does have the effect of

moving all other values into the XHTML namespace, such as `openid.delegate`. An alternative is to prohibit unprefixed CURIEs, other than those defined by XHTML.

A CURIE is a representation of a full IRI. This IRI is obtained by taking the currently in-scope mapping that is associated with `prefix`, and concatenating it with the `reference`. The result MUST be a syntactically valid IRI [IRI].

8. XHTML+RDFa Definition

This section is normative.

The XHTML+RDFa document type is a fully functional document type with rich semantics. It is a superset of [XHTML11]. See that document for the details of the underlying language.

The XHTML+RDFa 1.0 document type is made up of the following XHTML modules. The elements, attributes, and content models associated with these modules are defined in "XHTML Modularization" [XHTMLMOD]. The elements are listed here for information purposes, but the definitions in "XHTML Modularization" should be considered authoritative. In the on-line version of this document, the module names in the list below link into the definitions of the modules within the current versions of "XHTML Modularization".

Structure Module
: `body, head, html, title`

Text Module
: `abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var`

Hypertext Module
: a, and @href is available on all elements.

List Module
: `dl, dt, dd, ol, ul, li`

Object Module
: `object, param`

Presentation Module
: `b, big, hr, i, small, sub, sup, tt`

Edit Module
: `del, ins`

Bidirectional Text Module
: `bdo`

Forms Module
: `button, fieldset, form, input, label, legend, select, optgroup, option, textarea`

Table Module
: `caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr`

Image Module
: `img`

Client-side Image Map Module
: `area, map`

Server-side Image Map Module
: Attribute `ismap` on `img`

Intrinsic Events Module
: Events attributes

Metainformation Module
: `meta`

Scripting Module
: `noscript, script`

Stylesheet Module
: `style` element

Style Attribute Module ***Deprecated***
: @style

Target Attribute Module
: @target

Link Module
: `link`

Base Module
: `base`

Metainformation Attributes Module
: @about, @content, @datatype, @instanceof, @property

XHTML+RDFa also uses the Ruby Annotation module as defined in [RUBY]:

Ruby Annotation Module
`ruby, rbc, rtc, rb, rt, rp`

There are no additional definitions required by this document type. An implementation of this document type as an XML DTD is defined in Appendix B.

9. Metainformation Attributes Module

This section is *normative*.

The Metainformation Attributes Module defines the Metainformation attribute collection. This collection allows elements to be annotated with metadata throughout an XHTML-family document. When this module is included in a markup language, this collection is added to the Common attribute collection as defined in [XHTMLMOD].

9.1. Datatypes

Some of the attributes in this section use the following datatype:

Data type	Description
CURIE	A Compact URI or CURIE.
URIorCURIE	A URI or safe_curie.

9.2. Metadata Attribute Collection

9.2.1. The about attribute

This attribute specifies a URIorCURIE that indicates which resource has a specified property.

```
<meta about="http://www.example.com/" property="dc:created">2004-03-20</meta>
```

9.2.2. The content attribute

This attribute specifies a value of type CDATA that defines the metadata associated with an element. If not specified, then the metadata for an element is its content. If it is specified, and there is no `property` attribute, then the property is considered to be `reference`.

```
<meta about="http://www.example.com/" property="dc:created" content="2004-03-20"/>
```

9.2.3. The datatype attribute

This attribute defines as a CURIE the datatype of the content metadata of the element. If the attribute is not specified, then the default value is string as defined by [XMLSCHEMA].

```
<meta about="http://www.example.com/" property="dc:created" datatype="xsd:date">2004-
03-20</meta>
```

9.2.4. The instanceof attribute

This attribute indicates the `rdf:type` of the associated triple(s).

Default instanceof value

What is the default value for this attribute?

9.2.5. The property attribute

This attribute specifies a space-separated list of CURIEs that indicates which property is being defined by the element.

```
<meta about="http://www.example.com/" property="dc:creator">John Smith</meta>
```

The list of predefined values (in the XHTML namesapce) is given below. Users may extend this collection of relationships, however new values must be defined in their own vocabulary, and the relationship names must be referenced in documents as CURIEs (e.g., `dc:creator` for the Dublin Core "creator" relationship).

```
<html ....  xmlns:dc="http://purl.org/dc/elements/1.1/">
```

description
: Gives a description of the resource.

generator
: Identifies the software used to generate the resource.

keywords
: Gives a comma-separated list of keywords describing the resource.

reference
: The default value, gives no explicit information about the relationship with the resource.

robots
: Gives advisory information intended for automated web-crawling software. This specification does not define values for this property.

title
: Specifies a title for the resource.

Note that previous versions of XHTML included an `author` property; this has now been replaced with the Dublin Core `creator` property.

Note that the @title attribute is just a shorthand for a common case:

```
<a href="Jakob.html" title="Author biography">Jakob Nielsen</a>'s Alertbox for
January 11, 1998
```

is equivalent to:

```
<h2 about="#jakob" property="title">Author biography</h2>
<p><a href="Jakob.html" id="jakob">Jakob Nielsen</a>'s Alertbox for January 11,
1998</p>
```

This allows you to specify richer, marked-up text for a title when needed.

9.2.6. The rel attribute

This attribute describes the relationship between the resource specified by @about (or its default value) and the resource referred to by @href as defined in XHTML. The type for this attribute is a space-separated list of CURIEs.

```
<link href="top.html" rel="contents" />
```

This example defines a link to a table of contents for the current document.

```
<link href="doc.ps"
      rel="alternate"
      media="print"
      type="application/postscript" />
```

This example defines a link to an alternate version of the document especially suited to printing.

Authors may use the following relationship names, listed here with their conventional interpretations.

User agents, search engines, etc. may interpret these relationships in a variety of ways. For example, user agents may provide access to linked documents through a navigation bar.

Users may extend this collection of relationships. However, extensions must be defined in their own vocabulary, and the relationship names must be referenced in documents as CURIEs (e.g., dc:creator for the Dublin Core "creator" relationship).

Note that in order to reference relationship definitions via CURIE, their prefix must be defined via an xmlns attribute somewhere suitable:

```
<html ....  xmlns:dc="http://purl.org/dc/elements/1.1/">
```

alternate
: Designates alternate versions for the document.

appendix
: Refers to a resource serving as an appendix in a collection.

bookmark
: Refers to a bookmark. A bookmark is a link to a key entry point within an extended document. The @title attribute may be used, for example, to label the bookmark. Note that several bookmarks may be defined for a document.

cite

: Refers to a resource that defines a citation. In the following example, the `cite` is used to reference the book from which the quotation is taken:

cite as book reference

```
As Gandalf the White said in
<span rel="cite" about="http://www.example.com/books/the_two_towers">
    The Two Towers
</span>,
<quote xml:lang="en">"The hospitality of
your hall is somewhat lessened of late, Theoden King."</quote>
```

cite to reference another specification

```
More information can be found in
<span property="cite" about="http://www.w3.org/TR/REC-xml">[XML]</cite>.
```

chapter
: Refers to a resource serving as a chapter in a collection.

contents
: Refers to a resource serving as a table of contents.

copyright
: Refers to a copyright statement for the resource.

glossary
: Refers to a resource providing a glossary of terms.

help
: Refers to a resource offering help (more information, links to other sources of information, etc.)

icon
: Refers to a resource that represents an icon.

index
: Refers to a resource providing an index.

meta
: Refers to a resource that provides metadata, for instance in RDF.

next
: Refers to the next resource (after the current one) in an ordered collection.

p3pv1
: Refers to a P3P Policy Reference File. See [P3P].

prev
: Refers to the previous resource (before the current one) in an ordered collection.

role
: Indicates the purpose of the resource. For some possible values, see [XHTMLROLE] module.

section
: Refers to a resource serving as a section in a collection.

subsection
: Refers to a resource serving as a subsection in a collection.

start
: Refers to the first resource in a collection of resources. A typical use case might be a collection of chapters in a book.

No end or last value

We have a value of "start", but no corresponding "end" value. Do we need one?

up
: Refers to the resource "above" in a hierarchically structured set.

9.2.7. The resource attribute

This attribute takes a URIorCURIE, and can be used to define the resource referenced by a @rel, @rev, or @property attribute. When provided, the value of @resource supercedes any value for the @href attribute on the same element.

9.2.8. The rev attribute

This attribute is the complement of the @rel attribute and describes the reverse relationship between the resource specified by the @about attribute (or its default value) and the resource referred to by the @href attribute. Its value is a space-separated list of CURIEs. For a list of relationship names, see the @rel attribute.

```
<link href="doc.html" rev="contents"/>
```

This example states that the current document is the table of contents for the referenced document.

An implementation of this module can be found in Appendix B.

A. Other Host Languages

This section is informative.

While outside the scope of this specification, RDFa is intended to be extensible for use in host languages beyond XHTML 1.1. The XHTML 2 Working Group is producing a separate specification [XHTMLRDFA] that defines the XHTML Modularization-compatible [XHTMLMOD] modules to facilitate such host languages.

If a language includes @xml:base [XMLBASE], an RDFa parser for that host language must process it, and use its value to set [base].

An example follows to show how @xml:base affects the subject:

```
<span xml:base="http://internet-apps.blogspot.com/">
  <span about="" rel="dc:creator" href="http://www.blogger.com/profile/1109404" />
  <span about="" property="dc:title" content="Internet Applications" />
</span>
```

The triples generated would be as follows:

```
<http://internet-apps.blogspot.com/>
  dc:creator <http://www.blogger.com/profile/1109404> .
<http://internet-apps.blogspot.com/>
  dc:title "Internet Applications" .
```

B. XHTML+RDFa DTD

This appendix is *normative*.

This appendix includes an implementation of the XHTML+RDFa 1.0 language as an XML DTD. It is implemented by by combining the XHTML 1.1 DTD with the XHTML Metainformation Attribute Module. This is done by using a content model module, and then a driver module:

B.1. XHTML RDFa Module

```
<!-- ..................................................................... -->
<!-- XHTML Common Attributes Module  ..................................... -->
<!-- file: xhtml-attribs-1.mod

     This is XHTML-RDFa, modules to annotate XHTML family documents.
     Copyright 2007 W3C (MIT, ERCIM, Keio), All Rights Reserved.
     Revision: $Id: Overview.html,v 1.2 2007/09/27 16:16:26 jigsaw Exp $

     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML MetaAttributes 1.0//EN"
       SYSTEM "http://www.w3.org/MarkUp/DTD/xhtml-metaAttributes-1.mod"

     Revisions:
     (none)
     ..................................................................... -->

<!-- Common Attributes

     This module declares a collection of meta-information related
     attributes.

     %NS.decl.attrib; is declared in the XHTML Qname module.

     This file also includes declarations of "global" versions of the
     attributes.  The global versions of the attributes are for use on
     elements in other namespaces.
-->

<!ENTITY % QName.datatype "CDATA" >
<!ENTITY % QNames.datatype "CDATA" >
```

```
<!ENTITY % about.attrib
     "about        %URI.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.about.attrib
     "%XHTML.prefix;:about           %URI.datatype;        #IMPLIED"
>
]]>

<!ENTITY % instanceof.attrib
     "instanceof        %QName.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.instanceof.attrib
     "%XHTML.prefix;:instanceof           %QName.datatype;        #IMPLIED"
>
]]>

<!ENTITY % property.attrib
     "property        %QNames.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.property.attrib
     "%XHTML.prefix;:property           %QNames.datatype;        #IMPLIED"
>
]]>

<!ENTITY % resource.attrib
     "resource        %URI.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.resource.attrib
     "%XHTML.prefix;:resource           %URI.datatype;        #IMPLIED"
>
]]>

<!ENTITY % content.attrib
     "content        CDATA             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.content.attrib
     "%XHTML.prefix;:content           CDATA        #IMPLIED"
>
]]>

<!ENTITY % datatype.attrib
     "datatype        %QName.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.datatype.attrib
     "%XHTML.prefix;:datatype           %QName.datatype;        #IMPLIED"
>
]]>

<!ENTITY % rel.attrib
     "rel        %QNames.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.rel.attrib
     "%XHTML.prefix;:rel           %QNames.datatype;        #IMPLIED"
>
]]>

<!ENTITY % rev.attrib
     "rev        %QNames.datatype;             #IMPLIED"
>

<![%XHTML.global.attrs.prefixed;[
<!ENTITY % XHTML.global.rev.attrib
```

```
     "%XHTML.prefix;:rev           %QNames.datatype;        #IMPLIED"
>
]]>

<!ENTITY % Metainformation.extra.attrib "" >

<!ENTITY % Metainformation.attrib
     "%about.attrib;
      %content.attrib;
      %datatype.attrib;
      %instanceof.attrib;
      %property.attrib;
      %rel.attrib;
      %resource.attrib;
      %rev.attrib;
      %Metainformation.extra.attrib;"
>

<!ENTITY % XHTML.global.metainformation.extra.attrib "" >

<![%XHTML.global.attrs.prefixed;[

<!ENTITY % XHTML.global.metainformation.attrib
     "%XHTML.global.about.attrib;
      %XHTML.global.content.attrib;
      %XHTML.global.datatype.attrib;
      %XHTML.global.instanceof.attrib;
      %XHTML.global.property.attrib;
      %XHTML.global.rel.attrib;
      %XHTML.global.resource.attrib;
      %XHTML.global.rev.attrib;
      %XHTML.global.metainformation.extra.attrib;"
>
]]>

<!ENTITY % XHTML.global.metainformation.attrib "" >


<!-- end of xhtml-metaAttributes-1.mod -->
```

B.2. XHTML+RDFa Content Model Module

```
<!-- ..................................................................... -->
<!-- XHTML+RDFa Document Model Module  ..................................... -->
<!-- file: xhtml-rdfa-model-1.mod

     This is XHTML+RDFa.
     Copyright 1998-2007 W3C (MIT, ERCIM, Keio), All Rights Reserved.
     Revision: $Id: Overview.html,v 1.2 2007/09/27 16:16:26 jigsaw Exp $ SMI

     This DTD module is identified by the PUBLIC and SYSTEM identifiers:

       PUBLIC "-//W3C//ENTITIES XHTML+RDFa Document Model 1.0//EN"
       SYSTEM "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-model-1.mod"

     Revisions:
     (none)
     ..................................................................... -->

<!-- XHTML+RDFa Document Model

     This module describes the groupings of elements that make up
     common content models for XHTML elements.

     XHTML has three basic content models:

         %Inline.mix;  character-level elements
         %Block.mix;   block-like elements, eg., paragraphs and lists
         %Flow.mix;    any block or inline elements

     Any parameter entities declared in this module may be used
     to create element content models, but the above three are
     considered 'global' (insofar as that term applies here).
```

```
     The reserved word '#PCDATA' (indicating a text string) is now
     included explicitly with each element declaration that is
     declared as mixed content, as XML requires that this token
     occur first in a content model specification.
-->
<!-- Extending the Model

     While in some cases this module may need to be rewritten to
     accommodate changes to the document model, minor extensions
     may be accomplished by redeclaring any of the three *.extra;
     parameter entities to contain extension element types as follows:

         %Misc.extra;    whose parent may be any block or
                         inline element.

         %Inline.extra;  whose parent may be any inline element.

         %Block.extra;   whose parent may be any block element.

     If used, these parameter entities must be an OR-separated
     list beginning with an OR separator ("|"), eg., "| a | b | c"

     All block and inline *.class parameter entities not part
     of the *struct.class classes begin with "| " to allow for
     exclusion from mixes.
-->

<!-- ..............  Optional Elements in head  .................. -->

<!ENTITY % HeadOpts.mix
     "( %script.qname; | %style.qname; | %meta.qname;
      | %link.qname; | %object.qname; )*"
>

<!-- ..................  Miscellaneous Elements  .................. -->

<!-- ins and del are used to denote editing changes
-->
<!ENTITY % Edit.class "| %ins.qname; | %del.qname;" >

<!-- script and noscript are used to contain scripts
     and alternative content
-->
<!ENTITY % Script.class "| %script.qname; | %noscript.qname;" >

<!ENTITY % Misc.extra "" >

<!-- These elements are neither block nor inline, and can
     essentially be used anywhere in the document body.
-->
<!ENTITY % Misc.class
     "%Edit.class;
      %Script.class;
      %Misc.extra;"
>

<!-- ....................  Inline Elements  ..................... -->

<!ENTITY % InlStruct.class "%br.qname; | %span.qname;" >

<!ENTITY % InlPhras.class
     "| %em.qname; | %strong.qname; | %dfn.qname; | %code.qname;
      | %samp.qname; | %kbd.qname; | %var.qname; | %cite.qname;
      | %abbr.qname; | %acronym.qname; | %q.qname;" >

<!ENTITY % InlPres.class
     "| %tt.qname; | %i.qname; | %b.qname; | %big.qname;
      | %small.qname; | %sub.qname; | %sup.qname;" >

<!ENTITY % I18n.class "| %bdo.qname;" >

<!ENTITY % Anchor.class "| %a.qname;" >

<!ENTITY % InlSpecial.class
     "| %img.qname; | %map.qname;
      | %object.qname;" >
```

```
<!ENTITY % InlForm.class
     "| %input.qname; | %select.qname; | %textarea.qname;
      | %label.qname; | %button.qname;" >

<!ENTITY % Inline.extra "" >

<!ENTITY % Ruby.class "| %ruby.qname;" >

<!-- %Inline.class; includes all inline elements,
     used as a component in mixes
-->
<!ENTITY % Inline.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %I18n.class;
      %Anchor.class;
      %InlSpecial.class;
      %InlForm.class;
      %Ruby.class;
      %Inline.extra;"
>

<!-- %InlNoRuby.class; includes all inline elements
     except ruby, used as a component in mixes
-->
<!ENTITY % InlNoRuby.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %I18n.class;
      %Anchor.class;
      %InlSpecial.class;
      %InlForm.class;
      %Inline.extra;"
>

<!-- %NoRuby.content; includes all inlines except ruby
-->
<!ENTITY % NoRuby.content
     "( #PCDATA
      | %InlNoRuby.class;
      %Misc.class; )*"
>

<!-- %InlNoAnchor.class; includes all non-anchor inlines,
     used as a component in mixes
-->
<!ENTITY % InlNoAnchor.class
     "%InlStruct.class;
      %InlPhras.class;
      %InlPres.class;
      %I18n.class;
      %InlSpecial.class;
      %InlForm.class;
      %Ruby.class;
      %Inline.extra;"
>

<!-- %InlNoAnchor.mix; includes all non-anchor inlines
-->
<!ENTITY % InlNoAnchor.mix
     "%InlNoAnchor.class;
      %Misc.class;"
>

<!-- %Inline.mix; includes all inline elements, including %Misc.class;
-->
<!ENTITY % Inline.mix
     "%Inline.class;
      %Misc.class;"
>

<!-- .....................  Block Elements  ..................... -->

<!-- In the HTML 4.0 DTD, heading and list elements were included
     in the %block; parameter entity. The %Heading.class; and
```

```
     %List.class; parameter entities must now be included explicitly
     on element declarations where desired.
-->

<!ENTITY % Heading.class
     "%h1.qname; | %h2.qname; | %h3.qname;
      | %h4.qname; | %h5.qname; | %h6.qname;" >

<!ENTITY % List.class "%ul.qname; | %ol.qname; | %dl.qname;" >

<!ENTITY % Table.class "| %table.qname;" >

<!ENTITY % Form.class  "| %form.qname;" >

<!ENTITY % Fieldset.class  "| %fieldset.qname;" >

<!ENTITY % BlkStruct.class "%p.qname; | %div.qname;" >

<!ENTITY % BlkPhras.class
     "| %pre.qname; | %blockquote.qname; | %address.qname;" >

<!ENTITY % BlkPres.class "| %hr.qname; " >

<!ENTITY % BlkSpecial.class
     "%Table.class;
      %Form.class;
      %Fieldset.class;"
>

<!ENTITY % Block.extra "" >

<!-- %Block.class; includes all block elements,
     used as an component in mixes
-->
<!ENTITY % Block.class
     "%BlkStruct.class;
      %BlkPhras.class;
      %BlkPres.class;
      %BlkSpecial.class;
      %Block.extra;"
>

<!-- %Block.mix; includes all block elements plus %Misc.class;
-->
<!ENTITY % Block.mix
     "%Heading.class;
      | %List.class;
      | %Block.class;
      %Misc.class;"
>

<!-- ................  All Content Elements  .................. -->

<!-- %Flow.mix; includes all text content, block and inline
-->
<!ENTITY % Flow.mix
     "%Heading.class;
      | %List.class;
      | %Block.class;
      | %Inline.class;
      %Misc.class;"
>

<!-- end of xhtml-rdfa-model-1.mod -->
```

B.3. XHTML+RDFa Driver Module

```
<!-- ..................................................................... -->
<!-- XHTML 1.1 + RDFa DTD  ................................................ -->
<!-- file: xhtml-rdfa.dtd
-->

<!-- XHTML 1.1 + RDFa DTD
```

```
     This is an example markup language combining XHTML 1.1 and the RDFa
     modules.

     XHTML+RDFa
     Copyright 1998-2007 World Wide Web Consortium
        (Massachusetts Institute of Technology, European Research Consortium
         for Informatics and Mathematics, Keio University).
         All Rights Reserved.

     Permission to use, copy, modify and distribute the XHTML DTD and its
     accompanying documentation for any purpose and without fee is hereby
     granted in perpetuity, provided that the above copyright notice and
     this paragraph appear in all copies.  The copyright holders make no
     representation about the suitability of the DTD for any purpose.

     It is provided "as is" without expressed or implied warranty.

-->
<!-- This is the driver file for version 1 of the XHTML + RDFa DTD.

     Please use this public identifier to identify it:

         "-//W3C//DTD XHTML+RDFa 1.0//EN"
-->
<!ENTITY % XHTML.version  "-//W3C//DTD XHTML+RDFa 1.0//EN" >

<!-- Use this URI to identify the default namespace:

         "http://www.w3.org/1999/xhtml"

     See the Qualified Names module for information
     on the use of namespace prefixes in the DTD.

     Note that XHTML namespace elements are not prefixed by default,
     but the XHTML namespace prefix is defined as "xhtml" so that
     other markup languages can extend this one and use the XHTML
     prefixed global attributes if required.

-->
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % XHTML.prefix "xhtml" >

<!-- Be sure to include prefixed global attributes - we don't need
     them, but languages that extend XHTML 1.1 might.
-->
<!ENTITY % XHTML.global.attrs.prefixed "INCLUDE" >

<!-- Reserved for use with the XLink namespace:
-->
<!ENTITY % XLINK.xmlns "" >
<!ENTITY % XLINK.xmlns.attrib "" >

<!-- For example, if you are using XHTML 1.1 directly, use the public
     identifier in the DOCTYPE declaration, with the namespace declaration
     on the document element to identify the default namespace:

       <?xml version="1.0"?>
       <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
                             "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
       <html xmlns="http://www.w3.org/1999/xhtml"
             xml:lang="en">
       ...
       </html>

     Revisions:
     (none)
-->

<!-- reserved for future use with document profiles -->
<!ENTITY % XHTML.profile  "" >

<!-- ensure XHTML Notations are disabled -->
<!ENTITY % xhtml-notations.module "IGNORE" >

<!-- Bidirectional Text features
     This feature-test entity is used to declare elements
     and attributes used for bidirectional text support.
```

```
-->
<!ENTITY % XHTML.bidi  "INCLUDE" >

<!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Pre-Framework Redeclaration placeholder  ..................... -->
<!-- this serves as a location to insert markup declarations
     into the DTD prior to the framework declarations.
-->
<!ENTITY % xhtml-prefw-redecl.module "IGNORE" >
<!ENTITY % xhtml-prefw-redecl.mod "" >
<![%xhtml-prefw-redecl.module;[
%xhtml-prefw-redecl.mod;
<!-- end of xhtml-prefw-redecl.module -->]]>

<!-- we need the datatypes now -->
<!ENTITY % xhtml-datatypes.module "INCLUDE" >
<![%xhtml-datatypes.module;[
<!ENTITY % xhtml-datatypes.mod
     PUBLIC "-//W3C//ENTITIES XHTML Datatypes 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-datatypes-1.mod" >
%xhtml-datatypes.mod;]]>

<!-- bring in the RDFa attributes cause we need them in Common -->
<!ENTITY % xhtml-metaAttributes.module "INCLUDE" >
<![%xhtml-metaAttributes.module;[
<!ENTITY % xhtml-metaAttributes.mod
     PUBLIC "-//W3C//ENTITIES XHTML MetaAttributes 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-metaAttributes-1.mod" >
%xhtml-metaAttributes.mod;]]>

<!ENTITY % xhtml-events.module "INCLUDE" >

<!ENTITY % Common.extra.attrib
   "href         %URI.datatype;           #IMPLIED
    %Metainformation.attrib;"
>

<!-- Inline Style Module  ....................................... -->
<!ENTITY % xhtml-inlstyle.module "INCLUDE" >
<![%xhtml-inlstyle.module;[
<!ENTITY % xhtml-inlstyle.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Inline Style 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-inlstyle-1.mod" >
%xhtml-inlstyle.mod;]]>

<!-- declare Document Model module instantiated in framework
-->
<!ENTITY % xhtml-model.mod
     PUBLIC "-//W3C//ENTITIES XHTML+RDFa Document Model 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-model-1.mod" >

<!-- Modular Framework Module (required) ........................ -->
<!ENTITY % xhtml-framework.module "INCLUDE" >
<![%xhtml-framework.module;[
<!ENTITY % xhtml-framework.mod
     PUBLIC "-//W3C//ENTITIES XHTML Modular Framework 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-framework-1.mod" >
%xhtml-framework.mod;]]>

<!-- Post-Framework Redeclaration placeholder  ................... -->
<!-- this serves as a location to insert markup declarations
     into the DTD following the framework declarations.
-->
<!ENTITY % xhtml-postfw-redecl.module "IGNORE" >
<!ENTITY % xhtml-postfw-redecl.mod "">
<![%xhtml-postfw-redecl.module;[
%xhtml-postfw-redecl.mod;
<!-- end of xhtml-postfw-redecl.module -->]]>



<!-- Text Module (Required)  .................................... -->
<!ENTITY % xhtml-text.module "INCLUDE" >
<![%xhtml-text.module;[
<!ENTITY % xhtml-text.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Text 1.0//EN"
```

```
           "http://www.w3.org/MarkUp/DTD/xhtml-text-1.mod" >
%xhtml-text.mod;]]>

<!-- Hypertext Module (required) ..............................  -->
<!ENTITY % a.attlist  "IGNORE" >
<!ENTITY % xhtml-hypertext.module "INCLUDE" >
<![%xhtml-hypertext.module;[
<!ENTITY % xhtml-hypertext.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Hypertext 1.0//EN"
           "http://www.w3.org/MarkUp/DTD/xhtml-hypertext-1.mod" >
%xhtml-hypertext.mod;]]>
<!ATTLIST %a.qname;
      %Common.attrib;
      charset      %Charset.datatype;       #IMPLIED
      type         %ContentType.datatype;   #IMPLIED
      accesskey    %Character.datatype;     #IMPLIED
      tabindex     %Number.datatype;        #IMPLIED
>

<!-- Lists Module (required)  ..................................  -->
<!ENTITY % xhtml-list.module "INCLUDE" >
<![%xhtml-list.module;[
<!ENTITY % xhtml-list.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Lists 1.0//EN"
           "http://www.w3.org/MarkUp/DTD/xhtml-list-1.mod" >
%xhtml-list.mod;]]>

<!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: -->

<!-- Edit Module  ..............................................  -->
<!ENTITY % xhtml-edit.module "INCLUDE" >
<![%xhtml-edit.module;[
<!ENTITY % xhtml-edit.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Editing Elements 1.0//EN"
           "http://www.w3.org/MarkUp/DTD/xhtml-edit-1.mod" >
%xhtml-edit.mod;]]>

<!-- BIDI Override Module  .....................................  -->
<!ENTITY % xhtml-bdo.module "%XHTML.bidi;" >
<![%xhtml-bdo.module;[
<!ENTITY % xhtml-bdo.mod
     PUBLIC "-//W3C//ELEMENTS XHTML BIDI Override Element 1.0//EN"
           "http://www.w3.org/MarkUp/DTD/xhtml-bdo-1.mod" >
%xhtml-bdo.mod;]]>

<!-- Ruby Module  ..............................................  -->
<!ENTITY % Ruby.common.attlists "INCLUDE" >
<!ENTITY % Ruby.common.attrib "%Common.attrib;" >
<!ENTITY % xhtml-ruby.module "INCLUDE" >
<![%xhtml-ruby.module;[
<!ENTITY % xhtml-ruby.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Ruby 1.0//EN"
           "http://www.w3.org/TR/ruby/xhtml-ruby-1.mod" >
%xhtml-ruby.mod;]]>

<!-- Presentation Module  ......................................  -->
<!ENTITY % xhtml-pres.module "INCLUDE" >
<![%xhtml-pres.module;[
<!ENTITY % xhtml-pres.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Presentation 1.0//EN"
           "http://www.w3.org/MarkUp/DTD/xhtml-pres-1.mod" >
%xhtml-pres.mod;]]>

<!-- Document Metainformation Module  ..........................  -->
<!ENTITY % xhtml-meta.module "INCLUDE" >
<![%xhtml-meta.module;[
<!ENTITY % xhtml-meta.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Metainformation 1.0//EN"
           "http://www.w3.org/MarkUp/DTD/xhtml-meta-1.mod" >
%xhtml-meta.mod;]]>

<!-- Base Element Module  ......................................  -->
<!ENTITY % xhtml-base.module "INCLUDE" >
<![%xhtml-base.module;[
<!ENTITY % xhtml-base.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Base Element 1.0//EN"
           "http://www.w3.org/MarkUp/DTD/xhtml-base-1.mod" >
```

```
%xhtml-base.mod;]]>

<!-- Scripting Module  ..................................... -->
<!ENTITY % xhtml-script.module "INCLUDE" >
<![%xhtml-script.module;[
<!ENTITY % xhtml-script.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Scripting 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-script-1.mod" >
%xhtml-script.mod;]]>

<!-- Style Sheets Module  ..................................... -->
<!ENTITY % xhtml-style.module "INCLUDE" >
<![%xhtml-style.module;[
<!ENTITY % xhtml-style.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Style Sheets 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-style-1.mod" >
%xhtml-style.mod;]]>

<!-- Image Module  ........................................... -->
<!ENTITY % xhtml-image.module "INCLUDE" >
<![%xhtml-image.module;[
<!ENTITY % xhtml-image.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Images 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-image-1.mod" >
%xhtml-image.mod;]]>

<!-- Client-side Image Map Module  ............................ -->
<!ENTITY % area.attlist  "IGNORE" >
<!ENTITY % xhtml-csismap.module "INCLUDE" >
<![%xhtml-csismap.module;[
<!ENTITY % xhtml-csismap.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Client-side Image Maps 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-csismap-1.mod" >
%xhtml-csismap.mod;]]>
<!ATTLIST %area.qname;
      %Common.attrib;
      shape        %Shape.datatype;         'rect'
      coords       %Coords.datatype;        #IMPLIED
      nohref       ( nohref )               #IMPLIED
      alt          %Text.datatype;          #REQUIRED
      tabindex     %Number.datatype;        #IMPLIED
      accesskey    %Character.datatype;     #IMPLIED
>

<!-- Server-side Image Map Module  ............................ -->
<!ENTITY % xhtml-ssismap.module "INCLUDE" >
<![%xhtml-ssismap.module;[
<!ENTITY % xhtml-ssismap.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Server-side Image Maps 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-ssismap-1.mod" >
%xhtml-ssismap.mod;]]>

<!-- Param Element Module  .................................... -->
<!ENTITY % xhtml-param.module "INCLUDE" >
<![%xhtml-param.module;[
<!ENTITY % xhtml-param.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Param Element 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-param-1.mod" >
%xhtml-param.mod;]]>

<!-- Embedded Object Module  .................................. -->
<!ENTITY % xhtml-object.module "INCLUDE" >
<![%xhtml-object.module;[
<!ENTITY % xhtml-object.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Embedded Object 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-object-1.mod" >
%xhtml-object.mod;]]>

<!-- Tables Module ........................................... -->
<!ENTITY % xhtml-table.module "INCLUDE" >
<![%xhtml-table.module;[
<!ENTITY % xhtml-table.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Tables 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-table-1.mod" >
%xhtml-table.mod;]]>

<!-- Forms Module  ........................................... -->
```

```
<!ENTITY % xhtml-form.module "INCLUDE" >
<![%xhtml-form.module;[
<!ENTITY % xhtml-form.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Forms 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-form-1.mod" >
%xhtml-form.mod;]]>

<!-- Target Attribute Module  ................................... -->
<!ENTITY % xhtml-target.module "INCLUDE" >
<![%xhtml-target.module;[
<!ENTITY % xhtml-target.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Target 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-target-1.mod" >
%xhtml-target.mod;]]>

<!-- Legacy Markup .............................................. -->
<!ENTITY % xhtml-legacy.module "IGNORE" >
<![%xhtml-legacy.module;[
<!ENTITY % xhtml-legacy.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Legacy Markup 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-legacy-1.mod" >
%xhtml-legacy.mod;]]>

<!-- Document Structure Module (required)  ...................... -->
<!ENTITY % head.attlist  "IGNORE" >
<!ENTITY % xhtml-struct.module "INCLUDE" >
<![%xhtml-struct.module;[
<!ENTITY % xhtml-struct.mod
     PUBLIC "-//W3C//ELEMENTS XHTML Document Structure 1.0//EN"
            "http://www.w3.org/MarkUp/DTD/xhtml-struct-1.mod" >
%xhtml-struct.mod;]]>
<!ENTITY % profile.attrib
     "profile      %URI.datatype;           '%XHTML.profile;'"
>
<!ATTLIST %head.qname;
      %Common.attrib;
      %profile.attrib;
>

<!-- end of XHTML-RDFa DTD  ..................................................... -->
<!-- ........................................................................... -->
```

B.4. SGML Open Catalog Entry for XHTML+RDFa

This section contains the SGML Open Catalog-format definition [CATALOG] of the public identifiers for XHTML+RDFa 1.0.

```
-- ........................................................................... --
-- File catalog  ............................................................. --

--  XHTML+RDFa Catalog Data File

    Revision:  $Revision: 1.2 $

    See "Entity Management", SGML Open Technical Resolution 9401 for detailed
    information on supplying and using catalog data. This document is available
    from OASIS at URL:

        <http://www.oasis-open.org/html/tr9401.html>
--

-- ........................................................................... --
-- SGML declaration associated with XHTML  ................................... --

OVERRIDE YES

SGMLDECL "xml1.dcl"

-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --

-- XHTML+RDFa modules          ................................................ --
```

```
PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"       "xhtml-rdfa-1.dtd"


PUBLIC "-//W3C//ENTITIES XHTML+RDFa Document Model 1.0//EN"      "xhtml-rdfa-model-1.
mod"

PUBLIC "-//W3C//ENTITIES XHTML MetaAttributes 1.0//EN"           "xhtml-
metaAttributes-1.mod"

-- End of catalog data  ..................................................... --
-- ......................................................................... --
```

C. References

C.1. Related Specifications

This section is normative.

HTML4
"*HTML 4.01 Specification*", W3C Recommendation, D. Raggett *et al.*, *eds.*, 24 December 1999.
Available at: http://www.w3.org/TR/1999/REC-html401-19991224

IRI
"*Internationalized Resource Identifiers (IRI)*", RFC 3987, M.Duerst, M. Suignard January 2005.
Available at: http://www.ietf.org/rfc/rfc3987.txt

[RFC2119]
"*Key words for use in RFCs to indicate requirement levels*", RFC 2119, S. Bradner, March 1997.
Available at: http://www.rfc-editor.org/rfc/rfc2119.txt

[RUBY]
Ruby Annotation, W3C Recommendation, Marcin Sawicki, et al., 31 May 2001.
See: http://www.w3.org/TR/2001/REC-ruby-20010531

XHTML 1.1
"*XHTML 1.1 - Module-based XHTML*", W3C Recommendation, M. Altheim, S. McCarron, 31 May 2001.
Available at: http://www.w3.org/TR/2001/REC-xhtml11-20010531/.

XMLBASE
"*XML Base*", W3C Recommendation, J. Marsh, *ed.*, 27 June 2001.
Available at: http://www.w3.org/TR/2001/REC-xmlbase-20010627/

XML-LANG
"*Extensible Markup Language (XML) 1.0 (Third Edition)*", W3C Recommendation, T. Bray *et al.*, *eds.*, 4 February 2004.
Available at: http://www.w3.org/TR/2004/REC-xml-20040204

[XMLSCHEMA]
"*XML Schema Part 1: Structures Second Edition*", W3C Recommendation, H. S. Thompson *et al.*, *eds.*, 28 October 2004.
Available at: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
See also "*XML Schema Part 2: Datatypes Second Edition*", available at: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

C.2. Related Activities

This section is informative.

[CATALOG]
Entity Management: OASIS Technical Resolution 9401:1997 (Amendment 2 to TR 9401), Paul Grosso, Chair, Entity Management Subcommittee, SGML Open, 10 September 1997.
See: http://www.oasis-open.org/html/a401.htm

DC
Dublin Core Metadata Initiative (DCMI) (See http://dublincore.org/.)

FOAF-PROJECT
The FOAF Project (See http://www.foaf-project.org.)

N-TRIPLES
RDF Test Cases, N-Triples (See http://www.w3.org/TR/rdf-testcases/#ntriples.)

N3-PRIMER
N3 Primer (See http://www.w3.org/2000/10/swap/Primer.)

RDFa Primer
RDFa Primer 1.0 - Embedding Structured Data in Web Pages (see http://www.w3.org/2006/07/SWD/RDFa/primer.)

RDF-CONCEPTS
Resource Description Framework (RDF): Concepts and Abstract Syntax (See http://www.w3.org/TR/rdf-concepts/.)

RDF-PRIMER
RDF Primer (See http://www.w3.org/TR/rdf-primer/.)

RDF-SYNTAX
RDF/XML Syntax and Grammar (See http://www.w3.org/TR/rdf-syntax-grammar/.)

RDFTESTS-DATATYPES-TEST001
datatypes/test001.nt (See http://www.w3.org/2000/10/rdf-tests/rdfcore/datatypes/test001.nt.)

RDFTESTS-RDFMS-XMLLANG-TEST006
rdfms-xmllang/test006.nt (See http://www.w3.org/2000/10/rdf-tests/rdfcore/rdfms-xmllang/test006.nt.)

RELAXNG
RELAX NG Home Page (See http://www.relaxng.org/.)

SWD-WG
Semantic Web Best Deployment Working Group (See http://www.w3.org/2006/07/SWD/.)

RDFHTML
RDF-in-HTML Task Force (See http://w3.org/2001/sw/BestPractices/HTML/.)

SWBPD-WG
Semantic Web Best Practices and Deployment Working Group (See http://w3.org/2001/sw/BestPractices/.)

XHTML2-WG
XHTML 2 Working Group (See http://w3.org/MarkUp/Group/.)

CURIE
CURIEs (See http://w3.org/TR/curie.)

D. Change History

This section is informative.

2007-09-04: Migrated to XHTML 2 Working Group Publication System. Converted to a format that is consistent with REC-Track documents. Updated to reflect current processing model. Added normative definition of CURIEs. Started updating prose to be consistent with current task force agremeents. [ShaneMcCarron], [StevenPemberton], [MarkBirbeck]

2007-04-06: fixed some of the language to talk about "structure" rather than metadata. Added note regarding space-separated values in predicate-denoting attributes. [BenAdida]

2006-01-16: made the use of CURIE type for @rel, @rev, @property consistent across document (particularly section 2.4 was erroneous). [BenAdida]

E. Acknowledgments

This section is informative.

This section is informative.

At the time of publication, the participants in the W3C XHTML 2 Working Group were:

[RDFa] review of the syntax editor's draft

From: Diego Berrueta <diego.berrueta@fundacionctic.org>
Date: Thu, 04 Oct 2007 11:41:34 +0200
To: Ben Adida <ben@adida.net>
Cc: SWD WG <public-swd-wg@w3.org>, RDFa <public-rdf-in-xhtml-tf@w3.org>
Message-Id: <1191490894.9138.11.camel@duncan.fundacionctic.org>

```
El jue, 27-09-2007 a las 12:15 -0700, Ben Adida escribió:
> We noticed two small bugs in the RDFa syntax document editors' draft.
> Since they may lead to misunderstandings if you're looking at the
> details, we immediately pushed out a new draft:
>
> http://www.w3.org/MarkUp/2007/ED-rdfa-syntax-20070927

Hi all,

Find below my review of the "RDFa in XHTML: Syntax" Editor's Draft.
I apologize for any comment that might come from my
misunderstandings and not from actual issues in the draft.
I've skipped some trivial issues, such as broken or missing
references (DBpedia, for instance). I have little knowledge of DTD's,
so I couldn't properly review Appendix B.

I'd be glad to discuss these items in the Amsterdam f2f, or through the
mailing lists.

Best regards,

**********

- The <title> and the <h1> of the draft doesn't match. Which one is
the correct title of the document?

[[[
<title>RDFa in XHTML: Syntax and Processing</title>
...
<h1><a id="title" name="title"> RDFa in XHTML: Syntax</a></h1>
]]]

**********

- In Section 3.9, third paragraph:

[[[
predicates are represented using one of @property, @instanceof, @rel,
or @rev
]]]

IMO, @instanceof does not "represent" a predicate in the same sense as
@property or @rel. While the content of @property and @rel are URIs
(CURIEs) of RDF properties, the content of @instanceof is a class URI.
The actual predicate (rdf:type) is implied by the atribute.

My proposal is to remove "@instanceof" from the list.

Note that, two paragraphs before Section 6.1, we find this:

[[[
The attributes for setting a predicate are @rel, @rev and @property
]]]

which I think is correct, and apparently conflicts with the previous
quote.

**********

- In Section 5.3 ("Processing"), references are made to two flags
("recurse" and "chaining") which are undefined. It is not clear if
they're part of the "evaluation context" (Section 5.2) or not. I think
```

a note should be added to point explicitly they aren't part of the "evaluation context" (this is important because we don't want them pushed into the stack in rule 6). A similar remark can be done for the variables "current object resource" and "current object literal" which appear for the first time in the 2nd rule.

****

- In Section 6.2.2.2 ("Using @href"), the text following the first blue box is actually about the usage of "@rel" and "@rev", not about "@href". As the former can be used with other attributes besides @href (such as @resource, described in the preceding section), I think this text shouldn't be hidden in the Section concerning @href. Please consider moving these paragraphs about @rel/@rev to a section of their own.

****

- Section 6.2.2.4 ("Using a blank node to set the object") is a bit awkward to understand. Consider the first phrase:

[[[
When a triple predicate has been expressed using @rel, and no @href, @src, or @resource exists on the same [RDFa element], then the CURIE represented by this element is used as the object.
]]]

What is "the CURIE represented by this element"? How is it computed?

****

- In the example in Section 9.2.1, the @property attribute is highlighted, while I think it should be the @about attribute the one which is highlighted.

****

- In Section 9.2.4 ("The instanceof attribute"):

[[[
This attribute indicates the rdf:type of the associated triple(s).
]]]

I would rather say "the associated resource" or "the associated subject" instead of "the associated triple", because RDF triples have no "rdf:type" (unless we reify them, of course).

****

- At the bottom of Section 9.2.5 there are two examples that are supposed to be equivalent (i.e.: they should yield the same RDF triplets). Are RDFa user agents expected to process both syntaxes?

****

- In Section C.2 (Related Activities), the name of the SWD WG is not correct:

[[[
Semantic Web Best Deployment Working Group
]]]

****

- Regarding the processing model, in the first paragraph of Section 5.1 ("Overview"):

[[[
Parsing a document for RDFa triples is carried out by starting at the root element of the document, and visiting each of its child elements in turn, applying processing rules. Processing is recursive in that for each child element the processor also visits each of its child elements, and applies the same processing rules.
]]]

Later, in Section 5.3:

[[[
Processing then begins with the root element, and all nodes in the
tree are processed according to the following rules, depth-first.
]]]

>From the order of the rules, it follows that each node in the
tree is processed _before_ any of its children. IIRC, the precise
term for this is "pre-order".

Moreover, the rules can be applied only to nodes of type _element_.
Other kinds of DOM nodes, such as text nodes, attributes or
comments don't participate directly in the rules. Therefore, I
suggest to replace the term "node(s)" by the more precise
"element(s)" (e.g.: in the second quote above, also in rules 1 and 6,
and at the end of section 6.2.1.3). Note that, if we don't explicitly
restrict the rules to "elements", we can't safely use expressions
such as "current element" and "current element identifier".

****

- The 6th processing rule is not clear enough regarding how
the evaluation context is transmitted to the child elements:

[[[
If the [recurse] flag is true, the [current evaluation context] is
pushed onto a stack, and all nodes that are children of the [current
element] are processed using the rules described here. Once all of
the children have been processed then the [current evaluation context]
is popped back off the stack.
]]]

Does this mean that _all_ the children share _the same_ evaluation
context? IMO, this would be an error. Consider the following example:

```
<div about="#a">
  <span about="#b" property="foaf:name">Albert</span>
  <span property="foaf:name">Stephen</span>
</div>
```

Let's trace the rules: after the outer has been processed, the
current evaluation context contains current_resource="#a". By the
6th rule, we now push the current evaluation context into the
stack, and descend to the child elements. Nothing is said about
how to initialize the current evaluation context for the
children, so we assume that it's the same context as the one that we
just pushed into the stack. Processing the first changes the
current_resource to "#b". As the first has no child elements,
there is nothing to push or pop from the stack. We still continue
using the same evaluation context. Then the second
is processed, but the current_resource is no longer "#a" (as we could
expect), but "#b". A wrong triplet (#b,foaf:surname,Einstein) is
generated. Finally, the old evaluation context is popped back off the
stack.

I propose two alternative fixes:

a) To remove the stack, as it's just an implementation detail. The
6th rule could be rewritten as follows:

"If the [recurse] flag is true, all nodes that are children of the
[current element] are processed using the rules described here. Each
one of the children receives an identical copy of the [current
evaluation context]."

b) To keep the stack, but add a new rule to be executed before all
the other ones (the 0th rule):

"The [current evaluation context] is copied from the top of the
stack (but not popped!). If the stack is empty, an initial
[evaluation context] is generated as described in Section 5.2."

****

- Which is the value of the [current object literal] when the
current element has child nodes that are neither elements nor

text nodes (for instance, comments)?:

```
<span property="foaf:name">
  <!-- this is the name of the person -->
  Albert
</span>
```

Is it a rdf:XMLLiteral or a xsd:string?

****

--
Diego Berrueta
R&D Department - CTIC Foundation
E-mail: diego.berrueta@fundacionctic.org
Phone: +34 984 29 12 12
Parque Científico Tecnológico Gijón-Asturias-Spain
www.fundacionctic.org

Received on Thursday, 4 October 2007 09:42:15 GMT

SWD WG review of "Cool URIs for the Semantic Web"

Based on

- https://gnowsis.opendfki.de/repos/gnowsis/papers/2006_11_conceptуri/html/cooluris_sweo_note.
- http://lists.w3.org/Archives/Public/public-swd-wg/2007Sep/0062.html
- http://lists.w3.org/Archives/Public/public-swd-wg/2007Sep/0061.html
- http://lists.w3.org/Archives/Public/public-swd-wg/2007Sep/0068.html
- http://www.w3.org/2007/09/25-swd-minutes.html

Strengths

- General structure of the document is good.
- Provides reasonable solutions for many practical scenarios in the scope of (URI) naming strategies on the Semantic Web. The presented running examples facilitate easy and enjoyable reading of the document.
- Provides an excellent, pragmatic discussion of what it means to create URIs in the context of the semantic web. For a relative newcomer it seems that the semweb community has spent quite a bit of energy on the httpRange-14 issue This document represents the logical application of that work in providing some simple guidelines for both those who seek to publish RDF graphs on the web, and for application developers who are building agents to process those data sets.
- Will be useful to cite in documents like the SWD's Best Practice Recipes and SKOS Core Guide
  because it will provide sufficient background information on what the recipes are trying to accomplish. In some ways I think the document highlights the limited focus the Recipes on RDFS and OWL vocabularies instead of RDF graphs in general.

Editorial issues

- A 'Scope' section right after the Abstract would help to identify the intended audience.
- In Sec. 1 you write ' ... URIs and URLs share the same syntax ... '. Please, be more specific here; add references to the according RFCs (http://www.ietf.org/rfc/rfc2396.txt and http://www.ietf.org/rfc/rfc1738.txt)
- In Sec. 1, between the paragraph 3 and 4 there seems to be a logical break, IMHO.
- In Sec. 1, the last paragraph could go for example in the 'Scope' section.
- Sec. 4 heading - please rephrase to something less marketing-like

- In Sec. 4.2 the Fig. 4 seems a bit lost. Please provide more explanation and put in context.
- Sec. 4.4 needs a major rewrite. For example add a proper reference to CHIPS and explain it. See for example the Manual of Style for how to reference ...
- Sec. 4.6 would definitely benefit from references and some more details ...
- Sec. 6.1: the sentence 'For a more complete list, see here.' needs to be rewritten; see also noClickHere. Put a proper reference as well into the sentence 'The problems with new URI schemes are discussed at length by Thompson and Orchard.'
- Sec 6.2: The sentence 'Regarding FOAF's practice of avoiding URIs for people, we agree with Tim Berners-Lee: "Go ahead and give yourself a URI. You deserve it!"' seems not appropriate to me. Though I'm with you I don't see how this fits into this section. Please reformulate.
- Sec. 9: Can you please check the IPR issues. I'm note sure if this is in accordance with W3C policies
- As noted in the header this IG Note needs to be run through the pub rules checker
- p. 4, paragraph 5: Since the AWWW is being cited did the authors consider using 'information resource' and 'non-information resource' instead of 'web document' and 'non-document resource'? The AWWW seems to prefer 'information resource' and I couldn't find any mention of 'web document' in it. A change like this would ripple out across the document. Also did the authors consider a few examples of an 'information-resource sniff test'? I think a few examples of how to determine if a resource is an information resource or a non-information resource would be useful.
- p.5, paragraph 3: 'Requests for HTML would be redirected to the web page URLs we gave in section 2' might be clearer as 'HTTP requests for HTML content would be redirected to the HTML URLs we gave in section 2'.
- p.9, paragraph 3: In addition to seeing the use of <link> to to allow agents to discover RDF associated with an HTML document, it would be useful to see a similar example using RDFa and GRDDL. Does the fact that RDFa isn't a recomendation yet preclude it from being used in this document?
- p. 10, section 5: I would like to see dbpedia included since it uses the 303 redirect technique to link directly to a SPARQL query, and it is such a rich and evolving dataset. It would also be useful to be referred to a good hash URI real world example.
- p. 14, section 9: Would the restriction against derivative works prevent this document from being used as a W3C reference document? For example the SWD had talked about possibly including portions of the document in the Recipes document or elsewhere and this license was perceived to restrict that usage.
- The first paragraph of Section 1 could be a little more descriptive when mentioning the challenge of "...distributed modelling of the world with a shared data model...". If the audience of the document is (partially) not supposed to be absolutely familiar with the Semantic Web principles, the meaning of this could be explained in a little more detail.
- The note about public archive in the fifth paragraph of the "Status of this document" section is perhaps not needed - the reader who is familiar with the W3C mailinglists will know that, others will be notified anyway when sending an e-mail there.
- Maybe it would be better to change the font in the example statements presented in the second paragraph of Section 1 - for instance put "subjects" and "objects" in bold face

and "predicates" in italic. This could improve the readability of this piece in line with the intended impact.

- The remark on the possible MediaWiki adoption for Wikipedia is perhaps not completely appropriate in the end of Section 5, until the software would have been actually adopted.
- The following comment about the last note in Section 6.2 about personal URIs is rather "philosophical". Even though the note comes out from a strongly backed opinion;), it is really questionable how to actually establish such personal URIs in an ideal, practical and systematic way. Imagine creating URI of John Smith in a company XYZ - we can use company's dedicated namespace to distinguish among this John Smith and other John Smiths around the world. But what if more John Smiths are in a company? We can use perhaps a namespace or URI prefix according to the departments these guys work in. But what if they work in the same department? So maybe we can use a time-stamp or a respective slashed namespace inferred from the date when they joined the company. Etc... Such situations may of course rarely occur in practice for persons, but we should have a recommended way how to solve them, since similar problems may be encountered also with names of products, services and so on. Thus, the document should either propose a recommended way of how to deal with such problems, or be a little more "careful" and avoid such potentially questionable strict statements.
- Though the overall language quality of the document is quite high, it would be good to do some spell-checking and language clean-up by a native speaker. Just to mention few things noticed at the first sight even by a reviewing non-native speaker - the fourth paragraph of Section 1 is inconsistent in the tense used (from the sentence "In the remainder of this paper..." on - future and present tense are mixed); the third paragraph of Section 6.1 contains a typo ("resourcse").

Bigger issues

- The note suggests basically two ways how to publish RDF-based descriptions on the Web. There seems to be an implicit assumption that the vocabulary is published externally serialised using RDF/XML or an alike serialisation. The question now is how vocabularies should be treated that are defined 'inline', say, using RDFa; see for example a recent post by post by Dan Brickley regarding FOAF. Will the proposed recipes (hash and slash) still be applicable in the XHTML+RDFa setup? Is there a need for a third option?
- One rather conceptual issue is not covered by the document at all. Maybe the solution of the following problem is implicitly "entailed" by the document, but if it is indeed, it is not very clear from the presented strategies how to actually deal with such situations. Consider Gene Ontology
(GO in the following text), which is being monthly updated in many formats, RDF/XML being one of them. According to the requirement number 1 of the document under review, every resource identifiable by a URI should be on the web. But it is not very clear how to actually publish all resources GO contains following any of the strategies presented in the document. As stated in the Conclusion frame in Section 4.3, the hash URI strategy is not appropriate for GO, since its RDF/XML serialisation has currently about 30MB. It is definitely neither "rather small", nor "stable" (changes may occur every month). So, should we use the 303 URI approach? Possibly yes, but this would mean that we should establish and regularly maintain tens of thousands of

different URIs for every resource represented in GO, each of this URIs representing generally very small piece of information (since the GO descriptions are usually rather shallow). It is questionable whether such approach is either reasonable or practical... Maybe we could combine both approaches, which is one of the possibilities mentioned in Section 4.3 as well. However, it is not very clear from the document, how we should actually combine the two approaches in order to deal with situations similar to this "GO problem" optimally. In conclusion, there may be two possible solutions - either allow and discuss "off-line" exceptions from the requirement number 1 (be on the web), or propose a reasonable way of combining the two approaches and give respective examples to cover the aforementioned problem. The former seems to be not very systematic and would also possibly require to change the "attitude" of the whole document substantially, so the latter seems to be more appropriate.

ReviewCoolURIs (last edited 2007-09-27 09:21:56 by TomBaker)

@@@ THIS DOCUMENT NEEDS TO BE REWORKED AND FINISHED ACCORDING TO THE pubrules, the rules of the pub @@@

Cool URIs for the Semantic Web

W3C Interest Group Note @@@@@DD Month 2007

This version:
@@@@@http://www.w3.org/TR/2007/NOTE-sampleml1-2007MMDD/
Latest version:
@@@@@http://www.w3.org/TR/sampleml1/
Previous version:
@@@@@<previous version uri>

Authors:
Leo Sauermann (DFKI GmbH)
Richard Cyganiak (Freie Universität Berlin)
Max Völkel (FZI Karlsruhe)

Abstract

The *Resource Description Framework* RDF allows you to describe Web documents and resources from the real world—people, organisations, things—in a computer-processable way. Publishing such descriptions on the Web creates the *Semantic Web*. URIs are very important as the link between RDF and the Web. This article presents guidelines for their effective use. We discuss two strategies, called *303 URIs* and *hash URIs*. We give pointers to several Web sites that use these solutions, and briefly discuss why several other proposals have problems.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This is a W3C Working Group Note giving a tutorial explaining decisions of the TAG for semantic web beginners. This is an HTML conversion of DFKI Technical Memo TM-07-01, *Cool URIs for the Semantic Web* (PDF), reviewed by the Technical Architecture Group TAG. of the POWDER Use Cases and Requirements, developed by the POWDER Working Group as part of the Semantic Web Activity, to aid public discussion and solicit feedback on the group's aims. The group is particularly keen to learn of other potential use cases or additional features that should be considered for POWDER.

This document was developed by the Semantic Web Education and Outreach (SWEO) Interest Group. A complete list of changes to this document is available.

@@@@@@...indicate the level of endorsement within the group for the material, set expectations that the group has completed work on the topics covered by the document, and set expectations about the group's commitment to respond to comments about the document....

Please send comments about this document to public-sweo-ig@w3.org (with public archive).

Publication as an Interest Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

The disclosure obligations of the Participants of this group are described in the charter of SWEO.

Table of Contents

1. Introduction

The Semantic Web is envisioned as a decentralised world-wide information space for sharing machine-readable data with a minimum of integration costs. Its two core challenges are the distributed modelling of the world with a shared data model, and the infrastructure where data and schemas can be published, found and used. A basic question is thus how to publish information about resources in a way that allows interested users and software applications to find them.

On the Semantic Web, all information has to be expressed as *statements* about *resources*, like *the members of the company Acme are Alice and Bob* or *Bob's telephone number is "+1 800 262* or *this Web page was created by Alice*. Resources are identified by *Uniform Resource Identifiers* (URIs) [*RFC3986*]. This modelling approach is the *Resource Description Framework* (RDF) [*RDFPrimer*].

At the same time, Web documents have always been addressed with *Uniform Resource Locators*
(URLs). URIs and URLs share the same syntax, every URL is also a URI. This is good because it means we can easily make RDF statements about Web pages, but it is also dangerous because we can easily mix up Web pages and the things, or resources, described on the page.

In general, what URIs should we use in RDF? As an example, to identify the frontpage of the Web site of ACME Inc., we may use `http://www.acme.com/`. But what URI identifies the company as an organisation, not a Web site? And do we have to serve any content—HTML pages, RDF files—at those URIs? In the remainder of this paper we will answer these questions. We explain how to use URIs for things that are not Web pages, such as people, products, places, ideas, and ontology classes. We give detailed examples how the Semantic Web can (and we think: should) be realised as a part of the Web.

We assume that you are familiar with the basics of the RDF data model [*RDFPrimer*]. We also assume some familiarity with the HTTP protocol [*RFC2616*]. Wikipedia's article [*WP-HTTP*] serves as a good primer.

2. URIs for Web Documents

Let us begin with an example. Assume that ACME Inc., the fictional company from many Warner Brothers movies, has a Web site at `http://www.acme.com/`. Imagine, part of the site is a white-pages service listing the names and contact details of the employees. Alice and Bob both work at ACME. The structure of ACME's Web site might thus be:

`http://www.acme.com/`
: the homepage of ACME, Inc.

`http://www.acme.com/people/alice`

the homepage of Alice

http://www.acme.com/people/bob

the homepage of Bob

Like everything on the traditional Web, each of the pages mentioned above are *Web documents*. Every Web document has its own URI. Note that a Web document is not the same as a file: A single Web document can be available in many different formats and languages, and a single file, for example a PHP script, may be responsible for generating a large number of Web documents with different URIs. A Web document is defined as something that has a URI and can return *representations* (responses in a format such as HTML or JPEG or RDF) in response to HTTP requests. In technical literature, such as *Architecture of the World Wide Veb, Volume One* [*AWWW*], the term *information resource* is used instead of *Web document*.

On the traditional Web, URIs were used *primarily* for Web documents—to link to them, and to access them in a browser. In short, to *locate* a Web document—hence the term *URL* (Uniform Resource Locator). The notion of resource *identity* was not so important on the traditional Web, a URL simply identifies whatever we see when we type it into a browser.

2.1 HTTP and Content Negotiation

Today's Web clients and servers use the HTTP protocol [*RFC2616*] to request representations of Web documents and send back the responses. HTTP has a powerful mechanism for offering different formats and language versions of the same Web document: *content negotiation*.

When a user agent (e.g. a browser) makes an HTTP request, it sends along some HTTP headers to indicate what data formats and language it prefers. The server then selects the best match from its hard disk or generates the desired content on demand, and sends it back to the client. For example, a browser could send this HTTP request to indicate that it wants an HTML or XHTML version of `http://www.acme.com/people/alice` in English or German:

```
GET /people/alice HTTP/1.1
Host: www.acme.com
Accept: text/html, application/xhtml+xml
Accept-Language: en, de
```

The server could answer:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Language: en
```

followed by the content of the HTML document in English. Content negotiation [*TAG-Alt*] is often implemented with a twist: Instead of a direct answer, the server *redirects* to another URL where the appropriate version is found:

```
HTTP/1.1 302 Found
Location: http://www.acme.com/people/alice.en.html
```

The redirect is indicated by a special *status code*, here `302 Found`. The client would now send another HTTP request to the new URL. By having separate URLs for all versions, this approach allows Web authors to link directly to a specific version.

RDF/XML, the standard serialisation format of RDF, has its own content type too, `application/rdf+xml`. Content negotiation thus allows publishers to serve HTML versions of a Web document to traditional Web browsers and RDF versions to Semantic Web-enabled user agents. And it allows servers to provide alternative RDF serialisation formats like Notation3 [*N3*] or TriX [*TriX*].

3. URIs for Real-World Objects

On the Semantic Web, URIs identify not just Web documents, but also real-world objects like people and cars, and even abstract ideas and non-existing things like a mythical unicorn. We call all these things *resources*.

Given such a URI, how can we find out what resource it identifies? We need some way to answer this question, because otherwise it will be hard to achieve interoperability between independent information systems. We could imagine a service where we can look up a description for a URI, similar to today's search engines. But such a single point of failure is against the Web's decentralised nature.

Instead, we should use the Web itself—an extremely robust and scalable information publishing system—as a lookup service for resource descriptions. Whenever a URI is mentioned, we can look it up to retrieve a description containing relevant information and links to related data. This is so important that we make it our number one requirement for good URIs:

> **1. Be on the Web.**
>
> Given only a URI, machines and people should be able to retrieve a description about the resource identified by the URI from the Web. Such a look-up mechanism is important to establish shared understanding of what a URI identifies. Machines should get RDF data and humans should get a readable representation, such as HTML. The standard Web transfer protocol, HTTP, should be used.

Let's assume ACME Inc. wants to publish contact data of their employees on the Semantic Web so their business partners can import it into their address books. For example, the published data would contain these statements about Alice, written here in N3 syntax [*N3*]:

```
<URI-of-alice> a foaf:Person;
    foaf:name "Alice";
    foaf:mbox <mailto:alice@acme.com>;
    foaf:homepage <http://www.acme.com/people/alice> .
```

What URI should we use instead of the placeholder `<URI-of-alice>`? Certainly not `http://www.acme.com/people/alice`, because that would confuse a person with a Web

document, leading to misunderstandings: Is the homepage of Alice also named “Alice”? Has the homepage an email address? And why has the homepage a homepage? So we need another URI. (For in-depth treatments of this issue, see Tim Berners-Lee [*HTTP-URI2*] and David Booth [*Booth*]).

Therefore our second requirement:

> **2. Don't be ambiguous.**
>
> There should be no confusion between identifiers for documents (URLs) and identifiers for other resources. URIs are meant to identify only one thing, and one URI can't stand for both a Web-retrievable document and another real-world object.

We note that our requirements seem to conflict with each other. If we can't use URLs of documents to identify real-world object, then how can we retrieve a description about real-world objects based on their URL? The challenge is to find a solution that allows us to find the describing documents if we have just the resource's URI, using standard Web technologies. This would look somewhat like this:

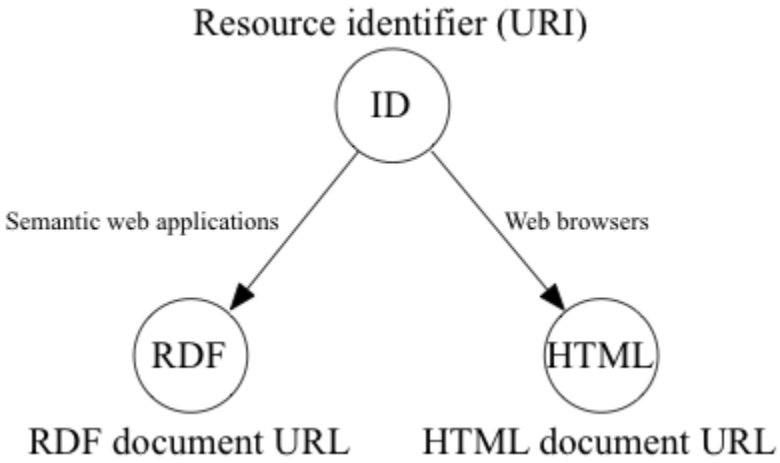


Figure 1: A resource and its describing documents

Another question is where to draw the line between traditional Web documents and other, non-document resources. According to W3C guidelines [*AWWW*], we may have a Web document if *all its essential characteristics can be conveyed in a message*. This is not a very precise definition. Our recommendation is to *err on the side of caution*: Whenever an object of interest is not clearly and obviously a document, then it's better to use two distinct URIs, one for the resource and another one for the document describing it.

4. Two Good Solutions

There are two solutions that meet our requirements: *303 URIs* and *hash URIs*. Which one to use depends on the situation, both have advantages and disadvantages.

4.1 303 URIs

The first solution is to use a special HTTP status code, "`303 See Other`", to distinguish other resources from regular Web documents. Since 303 is a redirect status code, the server can also give the location of a document that describes the resource. If, on the other hand, a request is answered with one of the usual status codes in the 2XX range, like `200 OK`, then the client knows that the URI identifies a Web document. This practice has been embraced by the W3C's Technical Architecture Group in its *httpRange-14 resolution* [*httpRange*].

If ACME adopts this solution, they could use these URIs to represent the company, Alice and Bob:

`http://www.acme.com/id/acme`
: ACME, the company

`http://www.acme.com/id/bob`
: Bob, the person

`http://www.acme.com/id/alice`
: Alice, the person

The Web server would be configured to answer requests to all these URIs with a 303 status code and a `Location`
HTTP header that provides the URL of a document that describes the resource:

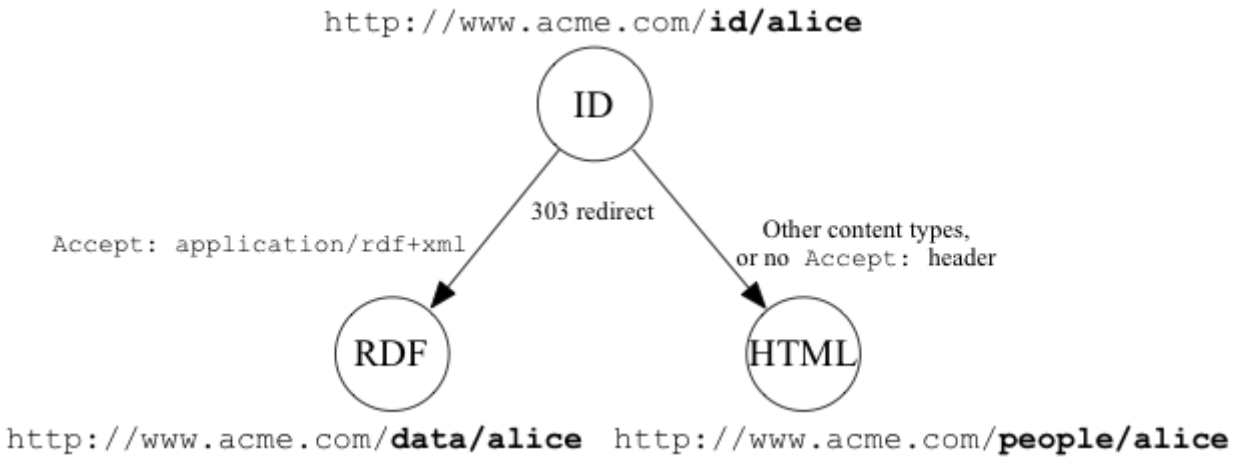


Figure 2: The 303 URI solution

The server could employ content negotiation (see Section 2.1) to send either the URL of an HTML description or RDF. Requests for HTML would be redirected to the Web page URLs we gave in Section 2. Requests for RDF data would be redirected to RDF documents, such as:

`http://www.acme.com/data/acme`
: RDF document describing ACME, the company

`http://www.acme.com/data/bob`

RDF document describing Bob, the person

http://www.acme.com/data/alice

RDF document describing Alice, the person

Each of the RDF documents would contain statements about the appropriate resource, using the original URI, e.g. `http://www.acme.com/id/alice`, to identify the described resource.

4.2 Hash URIs

The second solution is to use “hash URIs” for non-document resources. URIs can contain a *fragment*, a special part that is separated from the rest of the URI by a hash symbol (“#”).

When a client wants to retrieve a hash URI, then it is required to strip off the fragment part before requesting the URI from the server. This means a URI that includes a hash cannot be retrieved directly, and therefore cannot identify a Web document. We can use them to identify other, non-document resources, without creating ambiguity.

If ACME adopts this solution, then they could use these URIs to represent the company, Alice, and Bob:

http://www.acme.com/about#acme

ACME, the company

http://www.acme.com/about#bob

Bob, the person

http://www.acme.com/about#alice

Alice, the person

Clients will always strip off the fragment part before requesting any of these URIs, resulting in a request to this URI:

http://www.acme.com/about

RDF document describing ACME, Bob, and Alice

At this URI, ACME could serve an RDF document that contains descriptions of all three resources, using the original hash URIs to identify the resources.

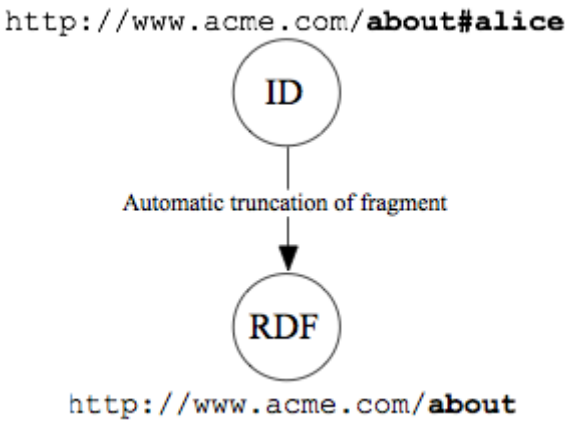


Figure 3: The hash URI solution without content negotiation

Alternatively, content negotiation (see Section 2.1) could be employed to redirect from the `about` URI to separate HTML and RDF documents. Again, the `303 See Other` status code must be used. (Otherwise, a client could conclude that the hash URI refers to a part of the HTML document.)

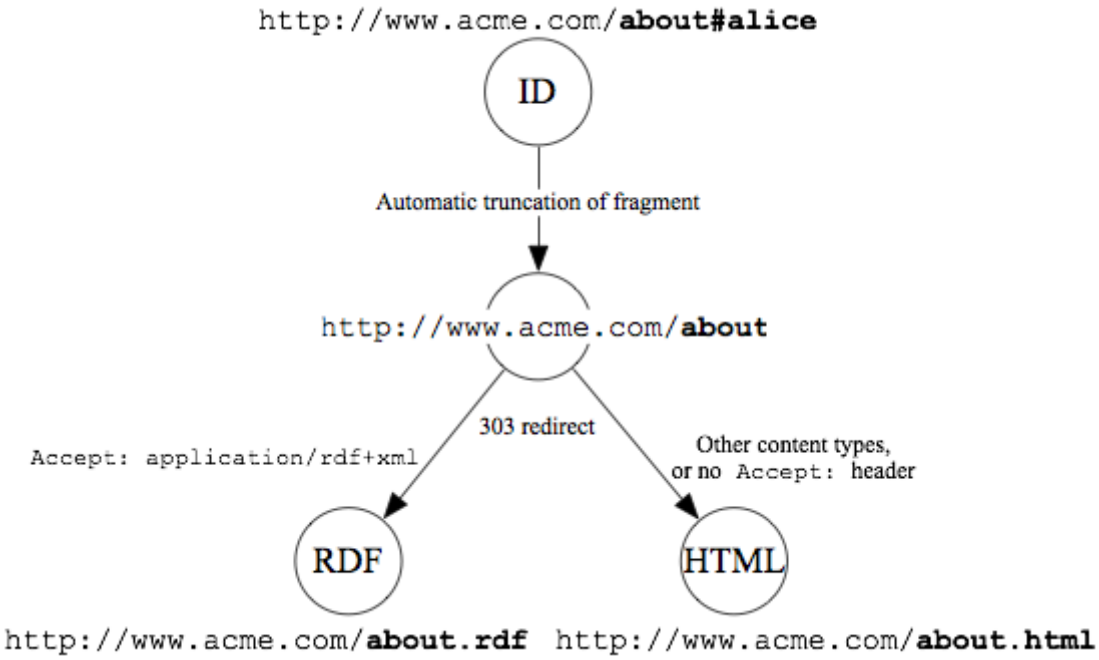


Figure 4: The hash URI solution with content negotiation

4.3 Choosing between 303 and Hash

Which approach is better? It depends. The hash URIs have the advantage of reducing the number of necessary HTTP round-trips, which in turn reduces access latency. A family of URIs can share the same non-hash part. The descriptions of `http://www.acme.com/about#acme`, `http://www.acme.com/about#product123`, and `http://www.acme.com/about# product456` are retrieved with a single request to `http://www.acme.com/about`. There is a counter-effect, too. A client interested only in `#product123` will inadvertently load the data for all other resources as well, because they are in the same file. 303 URIs, on the other hand, are very flexible because the redirection target can be configured separately for each resource. There could be one describing document for each resource, or one large document for all of them, or any combination in between. It is also possible to change the policy later on. But the large number of redirects may cause higher latency.

Conclusion.

Hash URIs should be preferred for rather small and stable sets of resources that evolve together. An ideal case are RDF Schema vocabularies and OWL ontologies, where the terms are often used together, and the number of terms is unlikely to grow much in the future.

Hash URIs without content negotiation can be implemented by simply uploading static RDF files to a Web server, without any special server configuration. This makes them popular for quick-and-dirty RDF publication.

303 URIs should be used for large sets of data that are, or may grow, beyond the point where it is practical to serve all related resources in a single document.

If in doubt, it's better to use the more flexible 303 URI approach.

4.4 Cool URIs

The best resource identifiers don't just provide descriptions for people and machines, but are also “cool”, a term Tim Berners-Lee uses for URIs designed with simplicity, stability and manageability in mind. More guidelines are in Sections 1 and 3 of “CHIPS”:

Simplicity.
: Short, mnemonic URIs will not break as easily when sent in emails and are in general easier to remember, e.g. when debugging your Semantic Web server.

Stability.
: Once you set up a URI to identify a certain resource, it should remain this way as long as possible. Think about the next ten years. Maybe twenty. Keep implementation-specific bits and pieces such as `.php` and `.asp` out of your URIs, you may want to change technologies later.

Manageability.
: Issue your URIs in a way that you can manage. One good practice is to include the current year in the URI path, so that you can change the URI-schema each year without breaking older URIs. Keeping all 303 URIs on a dedicated subdomain, e.g. `http://id.acme.com/alice`, eases later migration of the URI-handling server.

4.5 Linking

All the URIs related to a single real-world object—resource identifier, RDF document URL, HTML document URL—should be explicitly linked with each other to help information consumers understand their relation. For example, in the 303 URI solution for ACME, there are three URIs related to Alice:

`http://www.acme.com/id/alice`
: Identifier for Alice, the person

`http://www.acme.com/people/alice`
: Alice's homepage

`http://www.acme.com/data/alice`
: RDF document with description of Alice

Two of them are Web document URLs. The RDF document located at `http://www.acme.com/data/alice` might contain these statements (expressed in N3):

```
<http://www.acme.com/id/alice>
    foaf:page <http://www.acme.com/people/alice>;
    rdfs:isDefinedBy <http://www.acme.com/data/alice>;

    a foaf:Person;
    foaf:name "Alice";
    foaf:mbox <mailto:alice@acme.com>;
    ...
```

The document makes statements about Alice, the person, using the resource identifier. The first two properties relate the resource identifier to the two document URLs. The `foaf:page` statement links it to the HTML document. This allows RDF-aware clients to find a human-readable version of the resource, and at the same time, by linking the page to its topic, defines useful metadata about that HTML document. The `rdfs:isDefinedBy` statement links the person to the document containing its RDF description and allows RDF browsers to distinguish this main resource from other auxiliary resources that just happen to be mentioned in the document. We use `rdfs:isDefinedBy` instead of its weaker superproperty `rdfs:seeAlso` because the content at `/data/alice` is authoritative. The remaining statements are the actual white pages data.

The HTML document at `http://www.acme.com/people/alice` should contain in its header a `<link />` element that points to the corresponding RDF document:

```
<html lang="en">
  <head>
    <title>Alice's Homepage</title>
    <link rel="alternate" type="application/rdf+xml"
          title="RDF Version"
          href="http://www.acme.com/data/alice" />
  </head> ...
```

This allows RDF-aware Web clients to discover the RDF information. The approach is recommended in Section 9 of the RDF/XML specification. If the information on the Web page differs significantly from the RDF version, then we recommend using `rel="meta"` instead of `rel="alternate"`. The three desired links for each resource are shown here:

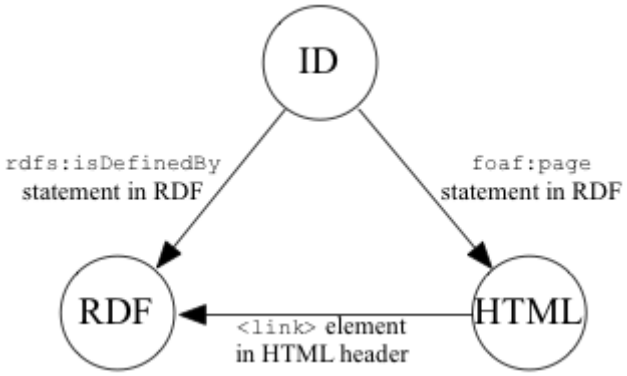


Figure 5: The RDF and HTML documents should relate the URIs to each other

4.6 Implementation

The W3C's Semantic Web Best Practices and Deployment Working Group has published a document
that describes how to implement the solutions presented here on the Apache Web server. The document mostly discusses the publication of *RDFS vocabularies*, but its ideas can also be applied to other kinds of small RDF datasets that are published from static files.

5. Examples from the Web

Not all projects that work with Semantic Web technologies make their data available on the Web. But a growing number of projects follow the practices described here. This section gives a few examples.

ECS Southampton. The School of Electronics and Computer Science at University of Southampton has a Semantic Web site that employs the 303 solution and is a great example of a carefully designed and well-documented Semantic Web engineering. Separate subdomains are used for HTML documents, RDF documents, and resource identifiers. Take these examples:

`http://id.ecs.soton.ac.uk/person/1650`
: URI for Wendy Hall, the person

`http://www.ecs.soton.ac.uk/people/wh`
: HTML page about Wendy Hall

`http://rdf.ecs.soton.ac.uk/person/1650`
: RDF about Wendy Hall

Entering the first URI into a normal Web browser redirects to an HTML page about Wendy Hall. It presents a Web view of all available data on her. The page also links to her URI and to her RDF document.

D2R Server
is an open-source application that can be used to publish data from relational databases on the Semantic Web in accordance with these guidelines. It employs the 303 solution and content negotiation. For example, the *D2R Server publishing the DBLP Bibliography Database*
publishes several 100k bibliographical records and information about their authors. Example URIs, again connected via 303 redirects:

`http://www4.wiwiss.fu-berlin.de/dblp/resource/person/315759`
: URI for Chris Bizer, the person

`http://www4.wiwiss.fu-berlin.de/dblp/page/person/315759`
: HTML page about Chris Bizer

The RDF document for Chris Bizer is a SPARQL query result from the server's SPARQL endpoint:

```
http://www4.wiwiss.fu-berlin.de/dblp/sparql?query=
DESCRIBE+\%3Chttp\%3A\%2F\%2Fwww4.wiwiss.fu-berlin.de
```

```
\%2Fdblp\%2Fresource\%2Fperson\%2F315759\%3E
```

The SPARQL query encoded in this URI is:

```
DESCRIBE <http://www4.wiwiss.fu-berlin.de/dblp/resource/person/315759>
```

This shows how a SPARQL endpoint can be used as a convenient method of serving resource descriptions.

Semantic MediaWiki
is an open-source Semantic Wiki engine. Authors can use special wiki syntax to put semantic attributes and relationships into wiki articles. For each article, the software generates a 303 URI that identifies the article's topic, and serves RDF descriptions generated from the attributes and relationships. Semantic MediaWiki drives the OntoWorld wiki. It has an article about the city of Karlsruhe:

`http://ontoworld.org/wiki/Karlsruhe`
: the article, an HTML document

`http://ontoworld.org/wiki/_Karlsruhe`
: the city of Karlsruhe

`http://ontoworld.org/index.php/Special:ExportRDF/Karlsruhe?xmlmime=rdf`
: RDF description of Karlsruhe

The URI of the RDF description is *not cool*, because it exposes the implementation (php) and refers redundantly to RDF in the path and in the query. A better URI would be for example `http://ontoworld.org/rdf/Karlsruhe`. There is an effort underway that calls for the adoption of Semantic MediaWiki as the software that runs Wikipedia. This would turn Wikipedia into a repository of identifiers with community-agreed descriptions.

6. Other Resource Naming Proposals

Many other approaches have been suggested over the years. While most of them are appropriate in special circumstances, we feel that they do not fit the criteria from Section 3, which are to *be on the Web* and *don't be ambiguous*. Therefore they are not adequate as general solutions for building a standards-based, non-fragmented, decentralized Semantic Web. We will discuss two of these approaches in some detail.

6.1 New URI schemes

HTTP URIs already identify Web resources and Web documents, not other kinds of resources. Shouldn't we create a new URI scheme to identify other resources? Then we could easily distinguish them from Web documents just by looking at the first characters of the URI. For example, the *info* scheme can be used to identify books based on a LCCN number: `info:lccn/2002022641`.

Here are examples of such new URI schemes. For a more complete list, see here.

- **Magnet**
  is an open URI scheme enabling seamless integration between Web sites and

locally-running utilities, such as file-management tools. It is based on hash-values, a URI looks like this:
`magnet:?xt=urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C`.

- The **`info:` URI scheme**
  is proposed to identify information assets that have identifiers in existing public namespaces. Examples are URIs for LCCN numbers (`info:lccn/2002022641`) and the Dewey decimal system (`info:ddc/22/eng//004.678`).
- The idea of **Tag URIs**
  is to generate collision-free URIs by using a domain name and the date when the URI was allocated. Even if the domain changes ownership at a later date, the URI remains unambiguous. Example: `tag:hawke.org,2001-06-05:Taiko`.
- **XRI**
  defines a scheme and resolution protocol for abstract identifiers. The idea is to use URIs that contain wildcards, to adapt to changes of organizations, servers, etc. Examples are `@Jones.and.Company/(+phone.number)` or `xri://northgate.library.example.com/(urn:isbn:0-395-36341-1)`.

To be truly useful, a new scheme must also define a protocol how to access more information about the identified resource. For example, the `ftp://` URI scheme identifies resourcse (files on an FTP server), and also comes with a protocol for accessing them (the FTP protocol).

Some of the new URI schemes provide no such protocol at all. Others provide a Web service that allows retrieval of descriptions using the HTTP protocol. The identifier is passed to the service, which looks up the information in a central database or in a federated way. The problem here is that a failure in this service renders the system unusable.

Another drawback can be a dependence on a standardization body. To register new parts in the `info:`
space, a standardization body has to be contacted. This, or paying a license fee before creating a new URI, slows down adoption. In cases a standardization body is desirable to ensure that all URIs are unique (e.g. with ISBNs). But this can be achieved using HTTP URIs inside an HTTP namespace owned and managed by the standardization organization.

The problems with new URI schemes are discussed at length by Thompson and Orchard.

6.2 Reference by Description

This approach radically solves the URI problem by doing away with URIs altogether: Instead of *naming* resources with a URI, *anonymous nodes* are used, and are *described* with information that allows us to find the right one. A person, for example, could be described with her name, date of birth, and social security number. These pieces of information should be sufficient to uniquely identify a person.

A popular practice is the use of a person's email address as a uniquely identifying piece of information. The `foaf:mbox` property is used in FOAF profiles for this purpose. In OWL, this kind of property is known as an *Inverse Functional Property* (IFP). When an agent encounters two resources with the same email address, it can infer that both refer to the

same person and can treat them as one.

But how to *be on the Web*
with this approach? How to enable agents to download more data about resources we mention? There is a best practice to achieve this goal: Provide not only the IFP of the resource (e.g. the person's email address), but also an `rdfs:seeAlso` property that points to a Web address of an RDF document with further information about it. We see that HTTP URIs are still used to identify the location where to download more information.

Furthermore, we now need several pieces of information to refer to a resource, the IFP value and the RDF document location. The simple act of linking by using a URI has become a process involving several moving parts, and this increases the risk of broken links and makes implementation more cumbersome.

Regarding FOAF's practice of avoiding URIs for people, we agree with Tim Berners-Lee: “Go ahead and give yourself a URI. You deserve it!”

7. Conclusion

Resource names on the Semantic Web should fulfill two requirements: First, a description of the identified resource should be retrievable with standard Web technologies. Second, a naming scheme should not confuse documents and the things described by the documents.

We have described two approaches that fulfill these requirements, both based on the HTTP URI scheme and protocol. One is to use the 303 HTTP status code to redirect from the resource identifier to the describing document. One is to use “hash URIs” to identify resources, exploiting the fact that hash URIs are retrieved by dropping the part after the hash and retrieving the other part.

The requirement to distinguish between resources and their descriptions increases the need for coordination between multiple URIs. Some useful techniques are: embedding links to RDF data in HTML documents, using RDF statements to describe the relationship between the URIs, and using content negotiation to redirect to an appropriate description of a resource.

8. Acknowledgements

Many thanks to Tim Berners-Lee who helped us understanding the TAG solution by answering chat requests. Special thanks go to Stuart Williams (HP Labs, member of TAG), who reviewed
this document thouroughly and provided essential feedback about many sentences that were (accidentially) contrary to the TAG's view. We wish to thank everyone who has reviewed drafts of this document, especially Chris Bizer and Gunnar AAstrand Grimnes.

This work was supported by the German Federal Ministry of Education, Science, Research and Technology (bmb+f), (Grants 01 IW C01, Project EPOS: Evolving Personal to Organizational Memories; and 01 AK 702B, Project InterVal: Internet and Value Chains) and by the European Union IST fund (Grant FP6-027705, Project Nepomuk).

X. References

@@@ Work in progress ...

AWWW
http://www.w3.org/TR/webarch/

Booth
http://www.w3.org/2002/11/dbooth-names/dbooth-names_clean.htm

HTTP-URI2
http://www.w3.org/DesignIssues/HTTP-URI2.html

httpRange
http://lists.w3.org/Archives/Public/www-tag/2005Jun/0039.html

N3
http://www.w3.org/DesignIssues/Notation3

RDFPrimer
http://www.w3.org/TR/rdf-primer/

RFC2616
http://www.ietf.org/rfc/rfc2616.txt

RFC3986
http://www.ietf.org/rfc/rfc3986.txt

TAG-Alt
http://www.w3.org/2001/tag/doc/alternatives-discovery.html

TriX
http://www.mulberrytech.com/Extreme/Proceedings/html/2004/Stickler01/EML2004Stickler01.html

WP-HTTP
http://en.wikipedia.org/wiki/HTTP

X. Change log

- 1.0 Initial Version **29.11.2006**.
- 1.1 Revised Version **9.8.2007**. Changes based on TAG review.