Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Indicazioni per la definizione di una ontologia e sua espressione mediante DTD

Alfredo Garro
Massimo Ruffolo

ICAR-CNR - Istituto di Calcolo e Reti ad Alte Prestazioni
del Consiglio Nazionale delle Ricerche -
Via P. Bucci 41/c, c/o Università della Calabria - 87030 Rende (CS) -
Tel: 0984-831711, Fax: 0984-839054,
Email: garro@si.deis.unical.it
ruffolo@icar.cnr.it

Rapporto Tecnico ICAR/CS/2002/01
Novembre 2002

Indicazioni per la definizione di una ontologia e sua espressione mediante DTD

SOMMARIO

ICAR-CNR (*Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche*)
Via Pietro Bucci 41C – 87030 Rende (CS) – Italy - tel. +39 0984 831720 – fax. +39 0984 839054

1 Introduzione

Per i nostri scopi un'ontologia è un insieme "gerarchicamente strutturato" di termini che descrive un dominio e che può essere utilizzato come schema per una base di conoscenza. Un ontologia, oltre a definire lo "schema di conoscenza" per un dominio, fornisce un "vocabolario semantico" comune che può essere utilizzato per riutilizzare, condividere e scambiare conoscenza fra applicazioni (ad es. Agenti Intelligenti), utenti e gruppi di utenti. In definitiva, quindi, un ontologia fornisce un "vocabolario semantico" (basato su concetti) comune per uno specifico dominio (o porzione del mondo) e definisce, con diversi livelli di formalismo, il "significato" dei termini (o meglio dei concetti) del vocabolario e le relazioni intercorrenti fra i termini stessi.

2 La scelta della metodologia

Esiste una varietà di metodologie da seguire per la definizione di una ontologia "*from the scratch*" [Tabella 1].

Methodologies for building ontologies from the scratch
TOVE Methodology
Enterprise Methodology
Sensus methodology
Bernaras, Laresgoiti, Corera Methodology
Methontology
CyC methodology (www.cyc.com)
Uschold and King
Gruninger and Fox
Kactus methodology
Onto-To-Knowledge Methodology (http://www.ontoknowledge.org/)

Tabella 1.

Methodologies for cooperative construction of ontologies
CO4 methodology
(KA)2 methodology
Ontology learning methodology
Aussenac-Gille's and colleagues (http://www.biomath.jussieu.fr/TIA/)
Maedche and colleagues' methodology
Ontology merge methodologies
FCA-merge
PROMPT
Ontology evaluation methods
Guarino's group methodology
Gómez Pérez's evaluation methodology

Tabella 2.

Ne presenteremo una molto semplice e generale: **Enterprise Methodology**. In base all'esperienza maturata posso, comunque, affermare che la scelta della metodologia da seguire non

influenza se non in minima parte la "qualità" dell'ontologia. La definizione di una ontologia, proprio come la realizzazione di software, è un'attività difficilmente formalizzabile: in altre parole non ci sono formule matematiche o algoritmi che applicati forniscano in output una ontologia!!! Si può, quindi, indicare, ed è quello che fanno quasi tutte le metodologie, solo una sequenza di passi da seguire senza specificare nel dettaglio come eseguire ogni singolo passo. Quasi tutti i passi, infatti, sono difficilmente formalizzabili ed in ognuno di essi entra in gioco la creatività, l'esperienza e la conoscenza del dominio, dell'ingegnere che deve progettare l'ontologia.
In Tabella 2 sono, inoltre, elencate le metodologie sviluppate per la definizione "cooperativa" di ontologie (Methodologies for cooperative construction of ontologies), per estrarre uno "schema concettuale" dall'analisi di un insieme di risorse documentali (Ontology learning methodology), per costruire una nuova ontologia integrando delle ontologie esistenti (Ontology merge methodologies), per valutare la "qualità" di una ontologia (Ontology evaluation methods).Come accennato in precedenza in questo documento verrà presentata come metodologia "Enterprise Methodology". La metodologia Enterprise Methodology prevede 5 fasi:

1. Identify Purpose and Scope
2. Building the ontology
   - Ontology Capture
     i. Identify key concepts and relations
     ii. Produce unambiguous text definitions
     iii. Identify terms to refer to such concepts and relations
   - Ontology Coding
     i. Commit to a meta-ontology
     ii. Choose a representation language
     iii. Write the code
3. Evaluation
4. Documentation
5. Guidelines for each phase

Il primo passo (Identify Purpose and Scope) è di fondamentale importanza, il progettista deve dare risposte chiare e precise alle domande: "Per quale motivo e per raggiungere quale scopo sto definendo la mia ontologia? Quale sarà il suo utilizzo? Sarà una componente di un sistema più complesso? Se si quale sarà il suo ruolo?". Le risposte date influenzeranno la successiva fase di costruzione dell'ontologia. Ad esempio, supponiamo di dover definire un'ontologia per descrivere la costituzione della materia. Se essa deve essere utilizzata per classificare gli esperimenti ed i risultati scientifici ottenuti in un centro di ricerca internazionale, quale il CERN, probabilmente sarà più dettagliata (avrà più concetti, più relazione e tipi di relazioni fra concetti,…) di una ontologia per lo stesso dominio ma utilizzata per classificare dei contenuti didattici o per realizzare un tutorial di fisica nucleare per un corso universitario.
Per quanto concerne la fase 2, essa è suddivisa in due sottofasi: **Ontology Capture** ed **Ontology Coding**. La sottofase di **Ontology Capture** è quella in cui, in base agli obiettivi individuati in fase 1, si deve rappresentare il domino di interesse individuando i concetti principali, le relazione fra concetti, i termini che costituiranno il vocabolario dell'ontologia, gli assiomi di consistenza per far si che il modello del mondo definito dall'ontologia sia "congruente" con la realtà che deve descrivere. Come si evince, in questa sottofase possiamo utilizzare qualsiasi linguaggio: naturale, formale,

grafico o di nostra invenzione. Nella successiva fase di **Ontology Coding**, invece, il progettista deve formalizzare quanto definito in precedenza scegliendo un meta modello ontologico ed un opportuno linguaggio di codifica. La scelta di tale linguaggio deve essere condotta in modo da permettere il raggiungimento degli obiettivi individuati in fase 1 ed è, inoltre, condizionata da vari fattori: conoscenza del linguaggio stesso da parte del progettista, riutilizzo di ontologie definite in precedenza, potere espressivo, eventuale supporto da parte di tool per la definizione di ontologie, documentazione, "portabilità", diffusione,…. Terminata la fase di codifica si passa alla sperimentazione, conforme agli obiettivi individuati in fase 1, dell'ontologia definita, si rilascia una documentazione dettagliata dell'ontologia, ed, ovviamente, qualora si riscontrassero carenze in quanto prodotto, si individua la fase o sottofase in cui si è "commesso l'errore" e si interviene di conseguenza. Si deve tener presente che, in fase di correzione, più ci si sposta a ritroso attraverso le fasi più il "peso" ed il "costo" della correzione aumenta.

3 Dalla teoria alla pratica

In teoria non c'è differenza tra teoria e pratica, in pratica si.

Nell'esempio che segue definiremo un'ontologia per il dominio "ricercatori universitari e pubblicazioni" (*OntoRP*) con l'obiettivo di classificare le pubblicazioni scientifiche di un dipartimento universitario, codificheremo, quindi, l'ontologia definita utilizzando il linguaggio DTD. La scelta del DTD come linguaggio di codifica è dovuta al fatto che ciò ci permetterà di descrivere ogni risorsa del dominio (nel nostro caso ricercatori e pubblicazioni) mediante metadati contenuti in file XML validati dal DTD definito. In pratica, una volta codificata la nostra ontologia in formato DTD, ogni risorsa del dominio potrà essere descritta da un file XML conforme al DTD definito [Figura 1]. Ciò ci permetterà di affiancare al database delle risorse un database di descrittori in formato XML da utilizzare per operazioni di ricerca semantica, di condivisione e trasferimento di conoscenza beneficiando di tutti i vantaggi offerti dalle attuali tecnologie XML based.

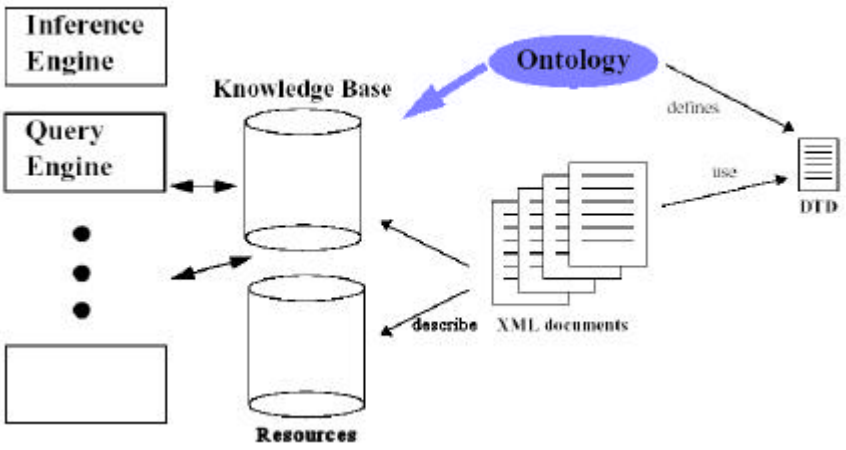

Figura 1.

3.1 Definizione di una semplice ontologia: OntoRP

Stabilito che l'obiettivo della nostra ontologia sarà quello di classificare le pubblicazioni scientifiche di un dipartimento universitario (FASE 1 dell'Enterprise Methodology) passiamo alla definizione vera e propria (FASE 2 sottofase "Ontology Capture") dell'ontologia ricordando che dobbiamo rappresentare il domino di interesse individuando i concetti principali, le relazione fra concetti, i termini che costituiranno il vocabolario dell'ontologia, gli assiomi di consistenza per far si che il modello del mondo definito dall'ontologia sia "congruente" con la realtà che deve rappresentare. Per descrivere quanto indicato, in questa sottofase, possiamo scegliere un qualunque linguaggio: naturale, formale, grafico o di nostra invenzione. Nell'esempio chi scrive ha scelto di utilizzare un linguaggio dalla semantica intuitiva che si ispira liberamente ai formalismi basati su logica descrittiva. Utilizzando tale linguaggio il dominio "ricercatori universitari e pubblicazioni" viene così modellato:

```
Object[].

Person :: Object.

Employee :: Person.

AcademicStaff :: Employee.

Researcher :: AcademicStaff.

PhDStudent :: Researcher.

Student :: Person.

PhDStudent :: Student.

Publication :: Object.

Book :: Publication.

Article :: Publication.

TechnicalReport :: Article.

JournalArticle :: Article.

Journal :: Publication.
```

```
Person[name =>> STRING; email =>> STRING; phone =>> STRING; publication
=>> Publication; editor =>> Book].

Employee[employeeNo =>> STRING].

AcademicStaff[supervises =>> PhDStudent].

Researcher[cooperatesWith =>> Researcher].

Student[studentID =>> NUMBER].

PhDStudent[supervisor =>> AcademicStaff].

Publication[author =>> Person; title =>> STRING; year =>> NUMBER;
abstract =>> STRING].

Book[editor =>> Person].

TechnicalReport[series =>> STRING; number =>> NUMBER].

JournalArticle[journal =>> Journal; firstPage =>> NUMBER; lastPage =>>
NUMBER].

Journal[editor =>> Person; volume =>> NUMBER; number =>> NUMBER;
containsArticle =>> JournalArticle].
```

```
FORALL Pers1, Pers2
Pers1:Researcher[cooperatesWith ->> Pers2] <->
Pers2:Researcher[cooperatesWith ->> Pers1].

FORALL Pers1, Publ1
Publ1:Publication[author ->> Pers1] <->
Pers1:Person[publication ->> Publ1].

FORALL Pers1, Publ1
Publ1:Book[editor ->> Pers1] <->
```

```
Pers1:Person[editor ->> Publ1].

FORALL Pers1, Pers2
Pers1:PhDStudent[supervisor ->> Pers2] <->
Pers2:AcademicStaff[supervises ->> Pers1].

FORALL Publ1, Publ2
Publ2:Journal[containsArticle ->> Publ1] <->
Publ1:JournalArticle[journal ->> Publ2].
```

L'ONTOLOGIA OntoRP.

L'ontologia consta di una gerarchia di concetti (persona, studente, ricercatore, pubblicazione,…) e definisce diverse associazioni fra questi concetti. La notazione X::Y indica che X e un "sotto-concetto" di Y. Si evince, ad esempio, che `Person` è un sotto-concetto di `Object`, `Employee` e `Student` sono sotto-concetti di `Person` ecc. Il concetto `PhDStudent` è, invece, "figlio" sia del concetto `Student` che `Researcher` e quindi "eredita" proprietà ed attributi da entrambi i concetti "padre". Le proprietà sono definite nella seconda parte dell'ontologia e ci permettono di indicare associazioni fra i concetti. Le associazioni sono realizzate mediante attributi di tipo appropriato. Infatti, oltre ad avere "attributi puri" di tipo STRING o NUMBER, possiamo utilizzare come tipo di un attributo un concetto definito in precedenza. Ad esempio `PhDStudent` ha un attributo `supervisor` di tipo `AcademicStaff` e viceversa. La terza parte dell'ontologia contiene regole ed assiomi per far si che il modello del mondo definito dall'ontologia sia "congruente" con la realtà che deve descrivere. Queste regole potranno, inoltre, essere usate per derivare nuova conoscenza a partire da un insieme di "fatti". Ad esempio, se sappiamo che: Resercher A cooperatesWith B, noi possiamo inferire, in base alla regola: `FORALL Pers1, Pers2 Pers1:Researcher[cooperatesWith ->> Pers2] <-> Pers2:Researcher[cooperatesWith ->> Pers1]`, che B deve essere un Resercher e che Resercher B CooperatesWith A.

3.2 Espressione di una ontologia mediante DTD

Vedremo in questa sezione come data in input una ontologia, come quella presentata nel paragrafo precedente, si può derivare un DTD che la descriva. L'autore dà per scontata la conoscenza della sintassi e della semantica dei "linguaggi" DTD ed XML. Abbiamo visto che un ontologia è essenzialmente una gerarchia di concetti ma come rappresentare la relazione di "sotto-concetto" utilizzando i costrutti sintattici del DTD? Ciò che si può fare è utilizzare il costrutto DTD "*ENTITY*" . Un elemento di tipo *ENTITY* definisce, essenzialmente, un insieme di stringhe interscambiabili all'interno del documento DTD. Ogni volta che ci si riferisce ad un elemento di tipo *ENTITY* il riferimento viene rimpiazzato con una delle possibile stringhe associate. Ad esempio, **per ogni concetto C che ha come sotto-concetti C1,C2, C3, C4, viene definito un elemento *ENTITY*** nel modo seguente:

```
<!ENTITY % C "C | C1 | C2 | C3 | C4" >
```

Questa definizione ci dice che ogni riferimento all'entità %C nel DTD verrà rimpiazzato dalla "disgiunzione" "C | C1 | C2 | C3 | C4", cioè in un documento XML, conforme al DTD, dovunque può essere inserito l'elemento (concetto) C allora può essere inserito anche il concetto C1 oppure C2 oppure C3 oppure C4. L'ontologia **OntoRP** ci dice che le Persone potrebbero avere delle

pubblicazioni: **Person[**`name =>> STRING; email =>> STRING; phone =>> STRING;` **publication =>> Publication**; `editor =>> Book`**].** Grazie all'ereditarietà: `Article :: Publication`. è perfettamente valido avere un Articolo (`Article`) come valore per l'attributo `publication` di Persona. Affinché ciò possa essere espresso in un documento XML dobbiamo definire nel DTD:

```
<!ENTITY % Publication "Publication | Book | Article | Journal ...">
```

Il passo successivo è quello di **definire per ogni concetto dell'ontologia un elemento di tipo *ELEMENT* nel DTD**. La struttura di un elemento così definito consiste di elementi che rappresentano gli attributi del concetto stesso. Per esempio, in **OntoRP** il concetto Libro (`Book`) ha 5 attributi, uno proprio (`editor`) e 4 ereditati dal suo "concetto-padre" `Publication` cioè `author`, `title`, `year`, `abstract`. Nel DTD ciò viene rappresentato come segue:

```
<!ELEMENT Book (#PCDATA| year | abstract | title | author | editor)*>
```

In un file XML, conforme al DTD che stiamo definendo, una particolare *risorsa libro* verrà descritta nel modo seguente:

<Book>
<title> The SGML Handbook. </title>
<author> Charles F. Goldfarb </author>
<year> 1990 </year>
</Book>

É bene ricordare che il DTD non indica (e non può indicare) l'ordine di comparizione degli attributi nel file XML, né che essi devono essere tutti presenti.

L'ultima cosa che rimane da fare è **specificare gli elementi "attributo" di ogni elemento** definito in precedenza (ad esempio `year`, `abstract`, `title`, `author`, `editor`). Ciò va fatto in accordo con il tipo dell'attributo definito nell'ontologia. Ad esempio, in **OntoRP**, l'attributo `title` è di tipo `STRING` e quindi non specifica una relazione con un altro concetto. Di conseguenza nel DTD l'elemento `title` sarà solo una sequenza di caratteri:

```
<!ELEMENT title (#PCDATA) >
```

L'attributo `author`, invece, è di tipo `Person`, definisce cioè una associazione fra il concetto `Publication` ed il concetto `Person`. Di conseguenza nel DTD avremo:

```
<!ELEMENT author (#PCDATA | %Person;)* >
```

Notiamo che un autore può essere anche un sotto-concetto di `Person` (`PhDStudent`, `Researcher`,...) e quindi `Person` è definita come elemento `ENTITY` (indicato con `%Person`):

```
<!ENTITY % Person "Person | Employee | AcademicStaff | Researcher |
PhDStudent | Student...">
```

In pratica è come se avessimo scritto:

```
<!ELEMENT author (#PCDATA | Person | Employee | AcademicStaff |
Researcher | PhDStudent | Student)* >
```

La cooperazione fra Ricercatori invece può essere espressa attraverso l'attributo `cooperatesWith`:

```
<!ELEMENT cooperatesWith (#PCDATA | %Researcher;)* >
```

Segue una parte del DTD che codifica l'ontologia **OntoRP** definita nel paragrafo 3.1:

```
<!-- entities for ontology concepts to realize is-a hierarchy -->
<!ENTITY % Person "Person | Employee | AcademicStaff |
 Researcher | PhDStudent | Student" >
<!ENTITY % Researcher "Researcher | PhDStudent" >
<!ENTITY % Publication "Publication | Book | Article |
 TechnicalReport | JournalArticle | Journal" >
<!ENTITY % Article "Article | TechnicalReport |
 JournalArticle" >
<!ENTITY % Book "Book" >

<!-- element declarations for ontology concepts -->
<!ELEMENT Person (#PCDATA | name | email | phone |
 publication | editor)*>
<!ELEMENT Researcher (#PCDATA | name | email | phone |
 publication | editor | employeeNo | supervises |
 cooperatesWith)*>
<!ELEMENT Publication (#PCDATA | author | title | year |
 abstract)*>
<!ELEMENT Article (#PCDATA | author | title | year |
 abstract)*>
<!ELEMENT Book (#PCDATA | author | title | year |
 abstract |editor)*>

<!-- element declarations for ontology attributes -->
<!ELEMENT author (#PCDATA | %Person;)* >
<!ELEMENT title (#PCDATA) >
<!ELEMENT year (#PCDATA) >
<!ELEMENT editor (#PCDATA | %Book; | %Person;)* >
<!ELEMENT cooperatesWith (#PCDATA | %Researcher;)* >
```

L'ONTOLOGIA OntoRP codificata mediante un DTD: OntoRP.DTD.

Il DTD presentato contiene tre sezioni. Nella prima sezione vengono definite le entità (`ENTITY`) che servono per simulare, attraverso il meccanismo di sostituzione illustrato, le relazioni IS-A. Nella seconda sezione vengono dichiarati gli elementi che rappresentano i concetti definiti nell'ontologia specificando per ogni elemento gli attributi che lo caratterizzano. Nella terza sezione, infine, vengono dichiarati gli elementi che definiscono gli attributi che compaiono nella seconda sezione specificandone il tipo che può essere o di base (`PCDATA`) oppure una entità o elemento derivante da un concetto dell'ontologia (`%Person, %Book, %Researcher`).

Notiamo che il DTD non specifica quale elemento sarà l'elemento "radice" di un documento XML da lui "valicato". In pratica differenti documenti XML possono avere differenti elementi "radice" pur essendo conformi allo stesso DTD. Ad esempio, un documento XML, che descrive una risorsa Ricercatore potrebbe essere il seguente::

<!DOCTYPE Researcher SYSTEM "OntoRP.dtd">

La radice sarà Researcher, ed il suo contenuto specificherà le proprietà del ricercatore in questione:

```
<Researcher>
      <name> Alfredo Garro </name>
      <address> Via P.Bucci 41C, 87036, Arcavacata di Rende (CS), Italy </address>
      <email> garro@si.deis.unical.it </email>
      <cooperatesWith> Giorgio Terracina </cooperatesWith>
      <publication>
        <Book>
              <title> ... </title>
              <year> ... </year>
              <editor> ... </editor>
        </Book>
        <Article>
              <title> ...</title>
              <author> Domenico Saccà </author>
              <abstract> ... </abstract>
        </Article>
      </publication>
</Researcher>
```

Un altro documento XML potrebbe descrivere una singola pubblicazione, ad esempio un libro. Anche in questo caso ci sarà una "document type declaration" (<!DOCTYPE Book SYSTEM " OntoRP.dtd">) seguita dalle proprietà specifiche di quel particolare libro.
Come si evince un documento XML che descrive una specifica risorsa non è che una specifica istanza di un concetto dell'ontologia (Person, Publication, Researcher, Book, …): una molteplicità di risorse possono essere descritte da altrettanti documenti XML usando un unico DTD ed una unica ontologia di riferimento. Tutti i documenti si riferiscono ad un'unica rappresentazione del mondo (ontologia) e possono essere "processati" da tutte le applicazioni in grado di "capire" il DTD a cui essi sono conformi; essi costituiscono una base di conoscenza coerente e consistente che è connessa sintatticamente dal DTD e semanticamente dall'ontologia che il DTD rappresenta.

4 Conclusioni

L'attività di definizione di una ontologia non è certo semplice, in parte perché difficilmente formalizzabile ed in parte perché implica sia una profonda conoscenza (diretta o indiretta) del dominio che si va a descrivere sia un bagaglio di conoscenze che spaziano dalla filosofia, alla logica, all'informatica teorica, all'ingegneria del software, ecc. "Seguire un esempio", per quanto buono e dettagliato esso possa essere, è dunque utile anche se del tutto insufficiente. Ciò premesso il documento ha cercato di affrontare l'argomento dal punto di vista "operativo". È stata presentata una semplice metodologia per la definizione di ontologie ed utilizzando un approccio "by examples" è stata definita una ontologia per un dominio molto ristretto. Sono state fornite, inoltre, delle dettagliate indicazioni su come descrivere una ontologia mediante la sintassi dei documenti DTD (Document Type Definition).

Bibliografia

1. R. Benjamins and D. Fensel: The Ontological Engineering Initiative (KA)2. in: Proceedings of the International Conference on Formal Ontologies in Information Systems (FOIS-98), Trento, Italy, N. Guarino (eds.), Frontiers in Artificial Intelligence and Applications, IOS-Press, June 1998.
2. Chandrasekaran, B.; Johnson, T. R.; Benjamins, V. R. “Ontologies: what are they? why do we need them?”. IEEE Intelligent Systems and Their Applications. 14(1). Special Issue on Ontologies. Pages 20-26. 1999.
3. Guarino, N.; Giaretta, P. “Ontologies and knowledge bases. towards a terminological clarification”. Toward Very Large Knowledge Bases. Ed. IOS Press. 1995. Págs. 25-32.
4. Gruber, T. R. “A translation approach to portable ontology specifications”. Knowledge Acquisition. Vol. 5. 1993.
5. Uschold, M.; Grüninger, M. 1996. "Ontologies: Principles Methods and Applications" Knowledge Engineering Review. Vol. 2.
6. Studer, R., Benjamins, R., Fensel, D. Knowledge Engineering: Principles and Methods. DKE 25(1-2).pp:161-197. 1998
7. Lenat, D.B.; Guha, R.V. “Building large knowledge-based systems”. Addison-Wesley Publising Company, Inc. 1990.
8. T. Bray, J. Paoli, and C.M.Sperberg-McQueen (eds.): Extensible Markup Language (XML) 1.0. W3C Recommendation, February 10, 1998. http://www.w3.org/TR/1998/REC-xml-19980210
9. J. Clark, S. DeRose (eds.): XML Path Language (XPath) 1.0. W3C Working Draft, August 13, 1999. http://www.w3.org/1999/08/WD-xpath-19990813.html
11. S. Decker, D. Brickley, J. Saarela, and J. Angele: A Query and Inference Service for RDF. in: Proceedings of the W3C Query Language Workshop (QL-98), Boston, MA, December 3-4, 1998.
12. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu: XML-QL: A Query Language for XML. W3C Note, August 19, 1998. http://www.w3.org/TR/NOTE-xml-ql/
13. R. Goldman, J. McHugh, and J. Widom: From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. Stanford University 1999. ftp://db.stanford.edu/pub/papers/xml.ps
14. M. Kifer, G. Lausen, and J. Wu: Logical Foundations of Object-Oriented and Frame-Based Languages, Journal of the ACM, 42, 1995.
15. O. Lassila and R.R. Swick: Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, February 22, 1999. http://www.w3.org/TR/REC-rdf-syntax
16. E. Maler and S. DeRose (eds.): XML Linking Language (XLink). W3C Working Draft. March 3, 1998. http://www.w3.org/TR/1998/WD-xlink-19980303
17. A. Malhotra and M. Maloney (eds.): XML Schema Requirements. W3C Note. February 15, 1999. http://www.w3.org/TR/NOTE-xml-schema-req
18. David Megginson (ed.): SAX - The Simple API for XML. http://www.megginson.com/SAX/index.html
19. V. Apparao et al. (eds.): Document Object Model (DOM). W3C Recommendation, October 1, 1998. http://www.w3.org/DOM/
20. J. Robie, J. Lapp, and D. Schach: XML Query Language (XQL). in: Proceedings of the W3C Query Language Workshop (QL-98), Boston, MA, December 3-4, 1998. http://www.w3.org/TandS/QL/QL98/pp/xql.html

21. J. Robie (ed.): XQL (XML Query Language). Working draft. August 1999. http://metalab.unc.edu/xql/xql-proposal.html
22. C. Welty and N. Ide: Using the right tools: enhancing retrieval from markedup documents. in: Journal Computers and the Humanities. 33(10):59-84. April, 1999.
23. D. Brickley and R.V. Guha: Resource Description Framework (RDF) Schema Specification. W3C Proposed Recommendation, March 3, 1999. http://www.w3.org/TR/PR-rdf-schema
24. J. Clark (ed.): XSL Transformations (XSLT) Secification 1.0. W3C Working Draft, April 21, 1999. http://www.w3.org/TR/1999/WD-xslt-19990421.html