Knowledge Management
Chapter 10

Semantic Web
Explaining the Term

- Semantics: Meaning of linguistic expressions
- Semantic Web: Network of contents
- As opposed to the previous web: Network of links (URL, URI)
Semantic Web: Motivation & Vision:

• Current Web:
  – Launched in early 1990s, exponential growth in mid-1990s
  – Search engines as tool for finding resources
  – Self-publishing increased amount of low quality hits from search engines
  – Spammers attempt to fool search engines
  – Current information on the web is understandable by humans, but not by machines.
  – Difficulties in using the Web to find relevant resources
Semantic Web Idea

• The Vision:

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.”

Tim Berners-Lee, James Hendler and Ora Lassila,
The Semantic Web, Scientific American, May 2001

• “Facilities to put machine-understandable data on the Web are becoming a high priority for many communities. The Web can reach its full potential only if it becomes a place where data can be shared and processed by automated tools as well as by people. For the Web to scale, tomorrow's programs must be able to share and process data even when these programs have been designed totally independently. The Semantic Web is a vision: the idea of having data on the web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications.”

www.w3.org/2001/sw/
## Three Generations of Web Pages

<table>
<thead>
<tr>
<th>First Generation</th>
<th>Second Generation</th>
<th>Third Generation</th>
</tr>
</thead>
</table>
The Landscape of the Semantic Web (1)

A rough division, no sharp boundaries:

Both parts influence each other in many respects

- Law
- Economic Models
- Marketing
- Computer Technology
- Etc.
The Landscape of the Semantic Web (2)

Computers and the Web:

Intelligent Software

Basic technology:
- hardware
- networks
- system programming
- XML
- etc.

Again:
Both sides influence each other

Content-dependent

Content-independent
The Landscape of the Semantic Web (3)

Logic:
- Data bases
- Formal semantics
- Deduction
- etc

Approximation:
- Inexactness
- Subjectivity
- Informal notions

Two aspects of intelligent behavior
Example Application: Web-Based B2C or B2B Rule Exchange

- Merchant\textsubscript{1}
- Merchant\textsubscript{m}

\[ \ldots \]

- publish rulebase\textsubscript{1}
- publish rulebase\textsubscript{m}

- translate to standard format
- compare, instantiate, and run rulebases

Customer or Company
Semantic Web “Architecture“

- Self-desc. doc.
- Data
- Rules
- Logic
- Ontology vocabulary
- RDF + rdfschema
- XML + NS + xsmlschema
- Unicode
- URI

Digital Signature

Trust

Proof

(Dim Berners-Lee, 2000)
Representation of the Syntax of Predicate Logic

- Principle: Each expression is built up from primitive components by constructors. Constructors have a specific arity.

- The basic property of the construction:
  Each expression can only be constructed in one way, i.e. there is no ambiguity.

Examples:
XML

• XML (Extensible Markup Language) is a further development of the language HTML for the WWW. In XML semiformal and formal documents can be represented.
• XML is universal a exchange format for WWW documents.
• Here we will discuss some aspects of XML for declarative and logical representation languages.
• General structure of an HTML document:
  – <tag> Text </tag>
• Meaning:
  – <tag> : paranthesis „(“ ; </tag> : paranthesis „)“
  – Text: A string; can contain iterated pairs <tag> Text </tag>.
• It is a semiformal document:
  – Formal: The tag-structure (can be represented as a tree)
  – Informal: The texts, they have no formal interpretation
• HTML & XML:
  – HTML describes the presentation, uses fixed tags
  – XML represents the content, tags are “user-defined“ by the DTD
XML and HTML

tag-expression in HTML:
• interpreted as a command
• The different strings (ag symbol) represent different kinds of commands.
• In HTML these commands are only concerned with forms of representations of the included text.
Example:

The command <i> means italic text.
The text between <i> and </i> is represented as italic, depending on the font used and the size of the letters.

• In XML such possibilities are extended by many more structural elements.
• These extensions are powerful enough in order to represent the syntax of predicate logic and fragments like horn logic.
• XML is not as powerful as KIF, e.g. most semantical aspects cannot be represented
XML Element

• An XML element is a fragment of the document encapsulated with named tags such as `<my_element>` and `</my_element>`
  – `<my_element>` marks the start of the element
  – `</my_element>` marks the end of the element

• Between the start and end tag we can have:
  – Arbitrary free text
  – Other elements (arbitrary nesting possible)
  – Nothing (empty element)

• Example:

  ```xml
  <hotel>
    <name>Meridien</name>
    <location>Hildesheim</location>
    <pricesgl>165 Euro</pricesgl>
    Very nice Hotel in the city center.
  </hotel>
  ```
Document Type Definitions (!ELEMENT)

• Simple context-free grammar for describing XML documents, which includes
  – the declaration of elements (tag names) and attributes
  – defines which elements can occur as sub-elements in other elements

• Syntax: <!ELEMENT name (definition)>  
  – name: name of the element (tag)
  – definition: allowed sub elements or free text

• Definition Syntax:
  – EMPTY: nothing is allowed between start end end tag.
  – #PCDATA (Parsed Character DATA): arbitrary free text
  – (a,b,c): List of sub-elements (a,b,c are the name of sub elements which must occur in this order)
  – (a|b|c): List of alternative sub-elements
  – Each sub-element or list can be followed by a cardinality specification:
    * element does not occur or occurs an arbitrary number of times
    + element does occurs at least once, or occurs an arbitrary number of times
    ? element does not occur or occurs once
    without cardinality specification: element occurs exactly once
Example

- DTD Fragment

```xml
<!ELEMENT hotel (name, location, pricesgl?, pricedbl?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT pricesgl (#PCDATA)>
<!ELEMENT pricedbl (#PCDATA)>
<hotel>
  <name>Meridien</name>
  <location>Hildesheim</location>
  <pricesgl>165 Euro</pricesgl>
  Very nice Hotel in the city center.
</hotel>
```

- XML Document
Document Type Definitions (!ELEMENT)

- Definition section in an !ELEMENT definition can be arbitrarily complex.
- Definitions may include the element that is defined, i.e. recursive definitions are allowed:
  Example:
  - `<!ELEMENT directory ( ( file | directory )* ) >`
- Definitions fix the order of the sub-elements, i.e. in `<!ELEMENT hotel (name, location, pricesgl?, pricedbl?)>` the name must be before the location before the optional pricesgl before the optional pricedbl.
  If this is not wanted, complex alternatives must be defined, e.g.
  `<!ELEMENT hotel ( (name, location, pricesgl?, pricedbl?) | (location, name, pricesgl?, pricedbl?) | (name, location, pricedbl?, pricesgl?) | (location, name, pricedbl?, pricesgl?) | >`
XML and Attributes (1)

- Tags can be augmented by attributes in the form $a_i = v_i$:
  - $<\text{tag } a_1 = v_1 \ldots a_n = v_n > \ldots </\text{tag}>$
- Structured tag iteration:

  
  
  
  
  
  
  

Document-Typ-Definitions (DTD‘s):
Allow the systematic introduction of (iterated) tags.
All DTD‘s could be given by context free grammars; this restricts the power of expressiveness but allows efficient handling.
XML Attributes (2)

• For every XML element we can have an arbitrary set of XML Attributes.
• An XML attribute is a attribute-name - attribute-value pair
• XML attributes are included in the elements start tag:
  <my_element  my_attr1="value1"  my_attr2="value2" >

• Example

three attributes

<hotel name="Meridien" location="Hildesheim" pricesgl="165 Euro"> Very nice Hotel in the city center. </hotel>
Document Type Definitions (!ATTLIST)

• Definition of attributes for an XML element through the following syntax:
  <!ATTLIST element attribute-name1 type1 specifier1 attribute-name2 …>

• Each attribute has one of the following types:
  – CDATA arbitrary text
  – ID an ID
  – IDREF a single reference to an ID
  – IDREFS a set of references to IDs’

• Each attribute has a specifier with the following meaning:
  – #REQUIRED attribute must occur in the element definition
  – #IMPLIED attribute may occur in the element definition
Example

• DTD Fragment

```xml
<!ELEMENT hotel (#PCDATA)>
<!ATTLIST hotel name CDATA #REQUIRED
location CDATA #REQUIRED
pricesgl CDATA #IMPLIED
pricedbl CDATA #IMPLIED>
```

• XML Document

```xml
<hotel name="Meridien" location="Hildesheim" pricesgl="165 Euro"> Very nice Hotel in the city center. </hotel>
```
Document Type Definitions (IDs)

- Attributes can contain IDs and references to IDs
- A value for an attribute with type ID defines an identifier for the element.
- An attribute with reference types to IDs can only hold IDs of other elements in the XML document.

**DTD**

```xml
<!ELEMENT apartment      
   name CDATA #REQUIRED  
   location CDATA #REQUIRED  
   rooms IDREFs #IMPLIED>

<!ELEMENT room           
   bedtype CDATA #REQUIRED  
   roomsize CDATA #REQUIRED  
   id ID #REQUIRED>
```

**XML**

```xml
<apartment name="Playa Famara"  
    location="Lanzarote" rooms="r1 r2">  
    Basically furnished apartment close to the sea.  
</apartment>

<room id="r1"  
    bedtype="QueenSize" RoomSize= "8 qm" >  
    Small bed room for children.  
</room>

<room id="r2"  
    bedtype="KingSize" RoomSize= "12 qm" >  
    Large bed room for parents.  
</room>
```
Document Type Definitions (IDs)

- Problems with IDs:
  - IDs are global for XML documents, i.e., the visibility of an ID cannot be restricted
  - IDs are not typed, i.e., there is no way to restrict that in the previous example, the rooms attribute only contains IDs to room elements.
XMI

• XMI (XML Metadata Interchange) is a standard which is intended for the exchange of models built on UML or related standards.
• With XMI one can exchange UML-models between different tools (as Rational Rose, Visual Age for Java, Visual Warehouse). Each tool communicates with XMI only, needs therefore only one export/import format.
• As in XML there are XMI-DTD’s and XMI-documents. Both are described by rules.
• XMI-DTD’s can be generated from UML-models. Each class and each attribute corresponds to an element in the DTD.
In XMI there are in addition attributes; each attributes has as prefix the name of the class to which the element belongs. Instead of texts (denoted by #PCDATA, parsed character data) references to external documents are allowed.
Example (2)

The car DTD in XMI:

```xml
<ELEMENT car (car.brand car.model car.year car.color car.price,
  XML.extensions*)?>
<!ATTLIST car  %XML.element.att; %XML.link.att>
<ELEMENT car.brand (#PCDATA XML.reference)*)>
<ELEMENT car.model(#PCDATA XML.reference)*)>
<ELEMENT car.year (#PCDATA XML.reference)*)>
<ELEMENT car.color (#PCDATA XML.reference)*)>
<ELEMENT car.price (#PCDATA XML.reference)*)>
```

The dash | denotes as in BNF the “or” and * is repetition. Link attributes are pointers to other XMI documents or attributes. Extensions allow to add other attributes.
RDF (Resource Description Framework)

- RDF is an alternative to XML
- RDF has two parts:
  - The RDF syntax (different syntaxes from XML)
  - An RDF model described as a set of triples
- The purpose of RDF is to define vocabularies (as parts of terminologies, see chapter on General Properties)
- The purpose of RDF is to define vocabularies (simple terminologies)
- It is machine readable
- It is of interest for digital libraries and e-commerce
The RDF Data Model

• The data model has
  – Resources:
    • A resource is an object that can be referenced
    • Resources have an URI
    • RDF definitions are again resources
  – Properties
    • Slots define relations to other resources or atomic values
  – Statements
    • Values can be resources or atomic XML - data
RDF Data Model

• Resource:
  – A resource is an information object (most likely in the Web) that is described by meta data
  – A resource is referenced by a Universal Resource Identifier (URI)
    • E.g. a URL is a special kind of URI
    • URIs can point to parts of a document

• Properties:
  – A resource is described by a property with a value
  – A property is a binary relation between a resource and a value
  – A property is also a special kind of resource

• Literals:
  – Literals are strings that stand for themselves
  – Can be used as values for properties

• Statements:
  – One meta data description item for a resource
  – Statement is a triple: (Resource, Property, Value)
    • Value can be a Resource or a Literal
RDF / RDF Schema

RDF = Resource Description Framework (from W3C)
• Purpose: Description of Resources (e.g. Web documents) in a relational manner
• Consists of:
  – RDF Data Model (*Resources has Property with value*)
  – RDF Syntax: representing a RDF Data Model as XML document

RDF Schema:
• Purpose: Vocabulary Definition Language (e.g. for an ontology)
• Can itself be described in RDF
• Description like OO, but “property centric“ representation
RDF Schema

• Vocabulary definition language (for RDF) and in RDF
• Can be used to encode an ontology
• Main components: special kinds of resources and properties are introduced:
  – rdfs:Resource: everything described is a rdfs:resource
  – rdfs:Class: Meta Class, i.e. a class whose instances are classes themselves
  – rdf:Property: a special kind of resource to describe properties
  – rdf:type: a property that represents the instance-of relation
  – rdfs:subclassOf: a property that represents the is-a relation
  – rdfs:domain: domain (left side) of a property
  – rdfs:range: range (right side) of a property
Example using RDF Schema
RDF Data Model - Example

• Statement:
  – M. M. Richter is the creator of the resource
    http://wwwagr.richter/Vorlesungen/KBS

• Structure:
  – Property (predicate): wwwagr.schema/#Creator
  – Value (object) „M.M. Richter“

• Represented as directed graph:
Example Data Model & Syntax (1)

- **Statement**: Ralph Bergmann is the creator of http://www.dwm.uni-hildesheim.de/~bergmann

- XML document syntax for this example

```xml
<rdf:RDF>
    <rdf:Description about="http://www.dwm.uni-hildesheim.de/~bergmann"/>
    <creator>Ralph Bergmann</creator>
</rdf:Description>
</rdf:RDF>
```
Example Data Model & Syntax (2)

• **Statements:**
  • Ralph Bergmann is the creator of http://www.dwm.uni-hildesheim.de/~bergmann
  • http://www.dwm.uni-hildesheim.de/~bergmann is part of http://www.uni-hildesheim.de

• XML document syntax for this example

```xml
<rdf:RDF>
  <rdf:Description about="http://www.dwm.uni-hildesheim.de/~bergmann">
    <creator>Ralph Bergmann</creator>
    <part_of rdf:resource="http://www.uni-hildesheim.de"/>
  </rdf:Description>
</rdf:RDF>
```

*Resources are allowed as values for properties!*
Example Data Model & Syntax (3)

• **Statement:**
  „Someone is the creator of http://www.dwm.uni-hildesheim.de/~bergmann and the name of that someone is Ralph Bergmann and the email address of that someone is bergmann@dwm.uni-hildesheim.de“

• XML document syntax for this example

```xml
<rdf:RDF>
  <rdf:Description about="http://www.dwm.uni-hildesheim.de/~bergmann">
    <creator>
      <rdf:Description ID="someone">
        <name> Ralph Bergman </name>
        <email rdf:resource="mailto:bergmann@dwm.uni-hildesheim.de">
      </rdf:Description>
    </creator>
  </rdf:Description>
</rdf:RDF>
```
Special Resources

- There are special resources:
  - Bag: unordered values (rdf:Bag)
  - Sequence: ordered values (rdf.Seq)
  - Alternative: single value (rdf.Alt)

- Example for Bag: The lecturers of course INTRO are Miller, Smith and Myers:
A Formal RDF - Model

• Basic definitions:
  – Resources
  – Properties \subseteq Resources
  – Literals
  – Statements = Properties \times Resources \times (Resources \cup Literals)

• Typing:
  – rdf:type \in properties
  – (rdf:type, sub, obj) \in statements \rightarrow obj \in Resources

• Collections:
• rdf:Bag, rdf:Seq, rdf:Alt \subseteq Properties
Document Meta Data in XML

```xml
<rdf:RDF>
  <rdf:Description about="http://.../Proposal/"
    <rdf:type rdf:resource="eg:Document" />
    <eg:author>
      <rdf:Description>
        <rdf:type rdf:resource="eg:Person" />
        <eg:name>Tim Berners-Lee</eg:name>
      </rdf:Description>
    </eg:author>
    <dc:title>Information Management: A Proposal</dc:title>
  </rdf:Description>
</rdf:RDF>
```
RDF Schema Complete

- RDF Schema contains even more modelling elements (not explained in detail here)
Examples of web-ontology languages

- **SHOE** (Simple HTML Ontology Extension, University of Maryland, Hendler): Uses in addition to HTML new tags for defining ontologies. Has an inheritance concept and allows rules of inference. Web pages can be annotated in order to extract information.

- **OML** (Ontology Mark-up Language, Washington State Univ., Kent): It is an extension of SHOE and is based on conceptual graphs.

- **XOL** (XML based Ontology Exchange Language, SRI, P. Karp): Allows to specify T-boxes and A-boxes. The semantics is based on
  Open Knowledge Base Connectivity (http://www.ai.sri.com/~okbc/)
HTML – XML – XML-based Ontologies (2)

DAML (Darpa Agent Mark-up Language): DAML-ONT is written in RDF (which is coded in XML and uses XML name spaces). Has many features like allowing synonyms, cardinalities, multiple inheritance.

OIL (Ontology Interface Language): Extends frame languages (OKBC) by terminological logics and web languages (XML, RDF). For OIL there is a DTD and an XML schema.

Annotated HTML Pages
Ontobroker Query
Summary

- Idea of the semantic web
- XML, XMI and RDF
- Ontologies in the semantic web