Semantic Web Technologies.

by

Aurona Gerber
Dr Alta van der Merwe
Prof. Andries Barnard

Technical Report

UNISA-TR-2006-02
http://www.osprey.ac.za/TechnicalReports/

UNIVERSITY OF SOUTH AFRICA

November 2006
# Table of Contents

1 Introduction v

2 Semantic Web Technologies v

3 Layer 1: Unicode and URI vi

3.1 Unicode v

3.2 URI viii

4 Layer 2: Namespaces, XML and XML Schema x

4.1 Namespaces x

4.2 XML xii

4.2.1 The History of XML xiii

4.2.2 HTML and XML xiv

4.2.3 XML Document Structure xv

4.2.4 Valid XML Documents xvii

4.2.5 XML Language Specifications xvii

4.3 XML Schema xix

4.3.1 Document Type Declaration (DTD) xix

4.3.2 XML Schema xxxiii

4.3.3 XML Schema documents xxxiv

4.3.4 Namespaces and XML Schemas xxxii

5 Layer 3 / Layers 3a and 3b xxxiii

5.1 RDF xxxiv

5.1.1 RDF Model xxxv

5.1.2 Blank RDF nodes xxxviii

5.1.3 RDF Vocabularies xxxix
5.1.4 The Application of RDF, XML and XML Schema on the Semantic Web .............. xli
5.2 RDF Schema ........................................... xli
  5.2.1 Summary of RDF Schema Constructs .................. xliii

6 Layer 4 / Layers 4a and 4b ................................. xlv
  6.1 Ontology Vocabulary / Ontology ......................... xlv
    6.1.1 OWL ............................................. xlvii
    6.1.2 OWL, OIL and DAML+OIL ......................... xlvii
    6.1.3 OWL Specification ............................... xlix
    6.1.4 Structure of an OWL Ontology ..................... li
    6.1.5 Description Logics .............................. liii
  6.2 Rules ................................................ lv
    6.2.1 SWRL .......................................... lv

7 Layer 5 .................................................. lvi
  7.1 Logic / Logic framework ............................... lvi

8 Layer 6 .................................................. lvii
  8.1 Proof ............................................... lvii

9 Layer 7 ................................................ lvi
  9.1 Trust ............................................... lix

10 Vertical Layers ......................................... lx
    10.1 Digital Signatures ................................ lx
    10.2 Encryption ....................................... lxi

11 The Technologies of the Later Versions of the Layered Architecture ............. lxii
11.1 General Adaptations                      lxiii
11.2 Layer 4 Adaptations                      lxiv
   11.2.1 SPARQL                                lxv
   11.2.2 DLP bit of OWL/Rul Layer              lxv
   11.2.3 RIF                                   lxvii

12 Conclusion                                 lxvii

List of Figures

1 The V1 Semantic Web Architecture ([11])       vi
2 The Adapted Semantic Web Architecture (V2) ([14])    vii
3 An RDF graph for a basic statement: http://www.thesis.org/SWTechnologies.html has a creator whose value is Student AJG      xxxv
4 An RDF Graph describing Eric Miller [99]       xxxvii
5 An RDF graph with a blank node [99]            xxxviii
6 The V3 version of the Semantic Web Architecture [15].       lxii
7 The V4 version of the Semantic Web Architecture [16].    lxii

List of Tables

1 XML based languages                          xix
2 Comparing XML Schema and DTD                xxiii
3 RDF Classes                                 xliii
4 RDF Properties                              xliv
1 Introduction

The purpose of this report is to present an overview of the technologies envisioned as part of the Semantic Web by Berners-Lee [11, 12, 13].

The Semantic Web was introduced in 2001 by Tim Berners-Lee [18] in his vision of a new intelligent Web. Complementing this vision, he proposed several versions of the Semantic Web architecture.

With regards to the adoption of the versions of the architecture, versions V1 and V2 have been adopted more than once by authors who regard this model as the Semantic Web architecture of choice [2, 48, 55, 58, 65, 78, 79, 91]. In contrast, there was no significant adoption of versions V3 and V4 of the architecture in literature. The technologies of the Semantic Web are discussed in this report with reference to the adopted versions (V1 and V2) of the layered architecture. For each technology the appropriate terms together with its history and relation to other important concepts are discussed.

Sections 2 to 10, discuss the technologies according to their position in the different layers in the architecture. These technologies are investigated in relation to the underlying architecture that will facilitate the founder vision of the Semantic Web as proposed by Berners-Lee [14, 18] in section 2 (page v). The report is concluded in section 12.

2 Semantic Web Technologies

The Semantic Web is an information space used by *machines* rather than *humans*. Instead of processing and manipulating Web information, a user would have a personal *agent* on his/her computer that would solve problems related to information overload, acquisition and discrepancy resolution [41]. Once an agent has executed the first level of information management, a user would access or manipulate the results. In order to execute these tasks, the information the agents uses has to be presented in an increasing semantically enriched format by means of several technology layers. These technology layers are depicted in the different versions of the Semantic Web layered
architecture.

Versions V1 and V2 of the architecture are presented in Figures 1 and 2 with added *Layer* captions for reference purposes. In both these versions of the Semantic Web architecture, a higher level layer language use the syntax and semantics of its immediate lower level layer.

![Figure 1: The V1 Semantic Web Architecture ([11])](image)

In sections 3 through 9 the different layers of V1 and V2, as presented in Figures 1 and 2, are considered. The discussion of each layer comprises a description of the residing technologies. Sections 10.1 and 10.2 discusses the vertical layers, digital signatures and encryption, which serves as identification authentication as well as security mechanisms for layers three to six.

## 3 Layer 1: Unicode and URI

Layer 1 in both models comprises *Unicode* and *URI (Uniform Resource Identifier)* technologies.
3 Layer 1: Unicode and URI

3.1 Unicode

Unicode aims to uniquely identify the characters in all the written languages by assigning a unique number to each character. In order to uniquely identify each character, Unicode specifies the universal character encoding standard used for representation of text for computer processing. In general, character encoding standards define not only the identity of each character and its numeric value (code point), but also the representation of the value in bits. Unicode extends ASCII by assigning a unique numeric value and name for each character used in all the written languages of the world [37].

The Unicode Standard is specified by the Unicode Consortium. The Unicode Consortium is a non-profit Organisation founded to develop, extend and promote use of the Unicode Standard. Its membership represents a broad spectrum of corporations and organisations in the computer and information processing industry. The standard supports three encoding mechanisms, UTF-8, UTF-16 and UTF-32, allowing the same data to be encoded in a byte, word or double word format (i.e. in 8, 16 or 32-bits per code unit). All three encoding mechanisms encode the same common characters and can be transformed into one another. Any of these encoding methods is endorsed as a way to implement the Unicode Standard.
UTF-8 is generally used for HTML or similar protocols. UTF-8 transforms all Unicode characters into a variable length encoding of bytes. It has the advantages that the Unicode characters corresponding to the familiar ASCII set have the same byte values as ASCII, and that Unicode characters transformed into UTF-8 can be used with existing software without rewrites. UTF-16 is utilised in environments that need to balance efficient access to characters with economical use of storage. It is more compact than UTF-8 and the characters that are used in general fit into a single 16-bit code unit, whilst all other characters are accessible via pairs of 16-bit code units. UTF-32 is popular where memory space is no concern and where fixed width, single code unit access to characters is desired. When using UTF-32 each Unicode character is encoded as a single 32-bit code unit.

The three encoding forms of Unicode use a common collection of characters but also allows for the encoding of more characters as required. The standard makes provision for all known character encoding requirements, including full coverage of the historic scripts of the world, as well as punctuation marks, diacritics\(^1\), mathematical symbols, technical symbols, arrows, and even characters such as dingbats.

The Unicode Standard has been adopted by various industry leaders and is required by standards such as XML, Java, JavaScript, LDAP and CORBA 3.0. In general it is supported in modern operating systems and browsers.

### 3.2 URI

A URI (Uniform Resource Identifier) is defined as an extendable, compact string of characters that is used for the identification of a resource. Furthermore, a URI is used to identify either an abstract or a physical resource. The standard URI specification of the IETF (Internet Engineering Task Force) is RFC3986\(^1\). The IETF is an open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the operation of the Internet. The technical

---

\(^1\)Diacritics are modifying character marks such as the tilde ‘\(\tilde{}\)’, that are used in conjunction with base characters to represent accented letters such as \(\ddot{\text{u}}\).
work of the IETF is done in its working groups, which are organised by topic into several areas (e.g., routing, transport, security, etc.) [61, 62].

According to RFC3986, a resource is defined as anything that has identity. A resource is the conceptual mapping to an entity or set of entities and not necessarily the entity itself which corresponds to that mapping at any particular instance in time. A resource, therefore, remains constant even when its content changes over time, provided that the conceptual mapping is not changed in the process. This is an area of debate called the Semantic Web Identification Problem and the related issues have not been resolved yet [89].

An identifier is an object that can act as a reference to something that has identity. In the case of URI, the object is a sequence of characters, ideally Unicode, with a restricted syntax.

In addition, the URI specification proposes uniformity which enables the use of different types of resource identifiers in the same context, as well as the reuse of a defined URI in many different contexts. This implies that new applications or protocols can refer to an existing specified set of resource identifiers that are in use. The URI specification aims to assist in the uniform semantic interpretation of common syntactic conventions across different types of resource identifiers.

URLs (Uniform Resource Locators) are a subset of URI that specifically identify resources by using their network location rather than identifying the resource by name or by other attributes. More specifically, a URL is a compact string representation of the location for a resource that is available via the Internet [19]. It generally takes the form: http://www.w3c.org/Addressing/activity

URNs (Uniform Resource Names) refers to the subset of URI that is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable [38, 72]. A URN takes the form: urn:NID:NSS where NID is the namespace identifier and NSS is the namespace-specific string. An example, obtained from Carey [32], uniquely identify a book with an ISBN as follow: urn:isbn:0-619-01969-7
At present non-ASCII characters are not allowed in URIs [17]. To lift this restriction, however, IRIs (Internationalised Resource Identifiers) are being developed in the W3C Internationalisation Activity [64]. In contrast to an URI, an IRI is a sequence of Unicode characters². The present endeavours towards IRIs suggest the use of UTF-8 as the preferred character encoding for URIs, and it also supports IRI-to-URI conversion [107].

Any Web development is inextricably involved with URIs because of the necessity of global identification. In particular, data on the Semantic Web is ideally described using IRIs rather than URIs. This is supported at present by W3C activities. URIs ensure that concepts are not just text but are tied to a unique definition that can be found by any Web user. With IRIs, concepts are universally accessible across language boundaries.

4 Layer 2: Namespaces, XML and XML Schema

Layer 2 comprises of Namespaces, XML (Extensible Markup Language) as well as XML Schema technologies (V1 in Figure 1). In V2 as depicted in Figure 2, XML Schema was omitted, however, for this the purpose of this discussion it will be included as a Layer two technology.

4.1 Namespaces

Namespaces (NS) provides a simple method for qualifying element and attribute names used in XML documents. Namespaces are identified by URI references. The W3C Namespace Recommendation defines an XML namespace as a collection of names, identified by a URI reference [17, 24], which are used in XML documents as element types and attribute names. The Second Edition of Namespaces in XML 1.0 [25] was released as a W3C Recommendation on 16 August 2006 and supersedes the first Recommendation of 14 January 1999 [23].

This section discusses the concept of namespaces as a qualifier for a domain

²Refer to section 3.1 on page vii for Unicode
specifying a grammar or vocabulary on the Semantic Web in XML using XML Schema. This discussion must therefore be read in conjunction with the sections on XML (section 4.2 and XML Schema (section 4.3)). However, Namespaces is presented before these sections as it is an underlying concept to any grammar specification such as XML. A namespace represents a collection of element types and datatype names and is identified by a unique name [26, 88, 114]. Namespaces are used to manage naming conflicts that invariably arise when different authors create grammars or vocabularies for the Semantic Web.

At present the W3C Recommendation recommends the use of a URL to identify a namespace because URLs indicate domain names that are unique and are used throughout the Internet. For the purpose of a namespace declaration, this specific URL does not mean a Web address, but unique identifier. XML namespaces differ from the namespaces conventionally used in computing disciplines in that the XML version has internal structure and is not, mathematically speaking, a set [26].

An example illustrates the use of namespaces. Namespaces are usually declared as an attribute of the root element in an XML Schema in the following manner:

```xml
<aElement xmlns:abc="http://www.abc.com" />
```

In the declaration of the attribute xmlns:abc, xmlns is a reserved word used only to declare a namespace or to bind namespaces, and it is not itself bound to any namespace. In addition the prefix abc is bound to the namespace http://www.abc.com

It is convention to use the phrase XSD or XS as a prefix for the XML Schema Namespace. When defining dedicated application namespaces, the use of meaningful namespace prefixes is recommended since it assists with the clarity of XML documents. Prefixes are used as placeholders and are expanded by the namespace-aware XML parser to use the actual namespace bound to the prefix. In the next example the elements Title and Author are associated with the namespace http://www.literature.com:

```xml
<?xml version="1.0"?> <Book xmlns:lib="http://www.literature.com">
```
<lib:Title>Emma</lib:Title>

<lib:Author>Jane Austen</lib:Author>

</Book>

It is possible to declare a default namespace, meaning that any element within the scope of the default namespace declaration will be qualified implicitly if it is not already qualified explicitly using a prefix. As with prefixed namespaces, a default namespace can be overridden.

The namespaces concept is defined to manage naming conflicts in the information space of the Web where different vocabularies containing the same name might co-exist. A namespace defines an information space wherein all the declared names are unique. Thus, when vocabularies are combined, names are unique when referenced in association with their namespaces.

4.2 XML

XML (Extensible Markup Language) specifies a standard for the exchange of data over networks, notably the Web. XML is considered to be both a metalanguage and a markup language [32, 41, 69, 113]. The function of a markup language is to describe information, usually for storage, transmission or processing by an application. The function of a metalanguage is to formally describe another language. XML as metalanguage allows for the specification of the content of documents according to a predefined and specific structure. All documents conforming to this specification will have the same structure or represent data items in the specified structure [95, 113]. In addition, XML as markup language allows for the insertion of markup tags into text to define the logical structure of a document, or to add information regarding information contained in a document (meta-data) [95, 96].

An example adapted from McKinnon and McKinnon [69] assists with clarification of the dual role of XML. XML as metalanguage can be used to specify the document structure of documents that is used to store the contact information of customers in a specific application. For instance, it might specify the first item as a surname, followed only by initials and relevant telephone
number. In addition, all the documents that contain this contact information will use XML tags to indicate that the first field in the document is the surname and initials and in this way XML is used as a markup language.

An XML document is a text document which in itself does not have any functionality. It is used only to describe data, information or meta-data [32, 113]. Thus, XML is a means for defining common grammars to enable data exchange. XML does not specify semantics, all parties participating in the data exchange must agree on the data model and document structure for XML data exchange to be successful. If an XML grammar is accepted as a standard for data exchange, any XML parser can parse the XML data and access the content if it is a valid XML document. It is however difficult to re-engineer the data model from any given XML document if the document type specification is not available [41, 77].

4.2.1 The History of XML

XML was developed from SGML or GML, originally developed by IBM, who foresaw the need for a way to separate data and its display information [32, p.1.04]. IBM released GML or Generalised Markup Language in 1973. With GML there was a first attempt to separate the specific formatting instructions of a document from the content of the document. GML’s generic encoding approach made a document transportable, meaning that it could be displayed or rendered in different styles without any changes to the original document. In 1978, ANSI (the American National Standards Institute) initiated the development of a standard based on GML [69, 83]. The result was SGML, the Standardised General Markup Language, approved in 1986 [69, 83, 85].

SGML is an extensive, versatile standard using generic descriptive markup so that the content of a document is defined completely separate from its processing. SGML also formalises the concept of a document type associated with a document in another file called the DTD (Document Type Definition). DTD is discussed in section 4.3.1 on page xix. A DTD identifies all the elements and their structural relationships to be contained in a document.
of that type and a document of a specific type can be verified against the
document type definition to ensure that it conforms to its type declaration
[83, 85].

In spite of its advantages, the versatility of SGML made it too cumbersome
and resource intensive to be adapted or fully incorporated into applications.

4.2.2 HTML and XML

HTML (Hypertext Markup Language) was developed by CERN (The Euro-
pean Organisation for Nuclear Research) mainly because SGML was consid-
ered too cumbersome to develop documents for the Web. HTML is an ap-
lication profile of SGML making HTML a SGML document type [83]. A fully
compliant SGML system would be able to process HTML documents. Thus,
HTML documents are SGML documents with predefined markup tags that
are appropriate for the representation of information on the Web [69, 84].

By the 1990's the Internet was widely adopted as information exchange
medium and the simplified specification of HTML could not meet the func-
tionality demands of Web developers any more. The W3C (World Wide
Web Consortium) initiated an activity to simplify SGML for Web applica-
tion development in 1996, which had as result XML [83]. Like HTML, XML
is an application profile of SGML and any SGML system that fully conforms
will be able to process XML documents. However, XML does not require a
system that is capable of understanding full SGML. XML is not expected to
replace HTML, rather it is designed to deliver structured content over the
Web [32, 82].

XML was readily adopted by Internet users, and the first specification, the
Extensible Markup Language (XML) 1.0 (First Edition) specification, was
accepted by the W3C in 1998 whilst the third edition was accepted as a
W3C recommendation in February 2004 [26].

XML was never intended as replacement of HTML, XML was designed to
describe data that has to be exchanged or transported over networks, whilst
HTML was designed to display Web content and make it acceptable for
general users of the Web.

4.2.3 XML Document Structure

An XML document is a text document in that it does not have any functionality in itself. It is used only to describe data, information or meta-data according to the XML specification. In order to use XML, an XML processor or XML parser, or application is required. An XML processor or XML parser reads an XML document, performs validity checks and provides access to the content and structure of the XML document on behalf of users or software applications. Applications are software programs that use XML documents. Note that there is a distinction between application and XML application in that an XML application is an application of XML, or an XML language that has been developed according to the XML specification [32, 41, 69].

The W3C XML Recommendation [26] specifies that XML documents have a physical structure made up of storage units or entities. Entities are fragments of XML documents which range in type and scope from single characters to complete external documents. Entities can be parsed or unparsed. Parsed entities contain markup or content text and should be parsed by an XML processor. An unparsed entity may be text or any other format, is not parsed by the XML processor and is generally passed without changes to the application using the XML document.

Furthermore, XML documents have a logical structure according to the W3C Recommendation, which means that an XML document have a prolog and one or more elements or containers of information that can be nested [26, 32]. An XML document prolog is the first major component in any XML document and it can consist of up to five possible components:

- An XML declaration
- Processing instruction(s)
- A Document Type Declaration
- Comments(s)
- White Space.
All these components except for the first one (XML declaration), are optional. The first component, the XML declaration, is the first line in any XML document and nothing should precede it:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

There are three attributes defined in the XML declaration, namely the the XML version number, 1.0, the document language encoding designation, UTF-8, and the standalone specification yes. The version (1.0 in this case) corresponds to W3C XML Recommendation 1.0. The encoding attribute is optional, if nothing is specified, the default is UTF-8. Other character set options such as Unicode are available. The standalone attribute is also optional with the default being "yes". Yes means that the document exists alone and there is no need to refer to any external document [26, 32, 69].

Elements are the basic building blocks of an XML document, and each document must at least have one element called the Element, root or parent element. The elements in an XML document are indicated by start and end tags as indicated below:

```xml
<elementname>
  content
</elementname>
```

All other elements contained in an XML document must be nested inside the root element’s start and end tags. This forms the element hierarchy of an XML document. Elements may have attributes or attribute specifications. Element attributes are data that can be specified for elements and they appear in the form of name-value pairs inside the start tag of an element as indicated below [26, 32, 69]:

```xml
<elementname attribute="value">
  content
</elementname>
```
XML documents conforming to all the XML syntax rules such as that all start tags must have end tags etc. are called well-formed XML documents. A well-formed XML document consists of a balanced tree of nested pairs of open and close tags. Each pair can include several attribute-value pairs [26, 32, 69].

### 4.2.4 Valid XML Documents

Valid XML documents are well-formed XML documents that also conforms to their document type specification as contained in the document’s respective document type declaration (DTD) or schema. The role of DTDs and XML Schemas are essentially the same, they create a grammar for specific XML documents when they specify, for instance, allowable combinations and nestings of tag names, as well as attribute names. Both DTDs and XML Schemas specify only syntactic conventions, any intended semantics are outside the realm of an XML specification [26, 41].

Section 4.3 on page xix contains a discussion on XML Schema.

### 4.2.5 XML Language Specifications

There is no fixed vocabulary for XML documents. XML vocabularies are created for each specific application as required [26, 32, 69, 95].

Since the XML 1.0 Recommendation was endorsed by the W3C in 1998, numerous XML-based vocabularies or languages have been developed in academia and industry by organisations that have to share high volumes of information. Some of these are depicted in Table 1 (adapted from McKinnon and McKinnon [69, p.17-18]).

<table>
<thead>
<tr>
<th>Language Acronym</th>
<th>Description</th>
<th>URL / Reference</th>
</tr>
</thead>
</table>

---

3section 4.3 on page xix
<table>
<thead>
<tr>
<th>CDF</th>
<th>The CDF (Channel Definition Format) is an open specification that permits a web publisher to offer frequently updated collections of information, or channels, from any web server for automatic delivery to compatible receiver programs on PCs or other information appliances</th>
<th><a href="http://www.w3.org/TR/NOTE-CDFsubmit.html">http://www.w3.org/TR/NOTE-CDFsubmit.html</a> (accessed July 2006)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CML</td>
<td>Chemical Markup Language (CML) is an extensible base for chemically aware markup languages. The World Wide Molecular Matrix is a molecular repository and contains and manages chemical information and molecules entirely in XML and CML (chemical markup language) / CMLComp (computational chemical markup language).</td>
<td><a href="http://www.xml-cml.org/">http://www.xml-cml.org/</a> (accessed July 2006)</td>
</tr>
<tr>
<td>ETD-ML</td>
<td>The Electronic Thesis and Dissertation Markup Language (ETD-ML) allows semantic encoding of ETDs independent of visual appearance and allows simplified hypertext and multimedia. ETD-ML converts theses from documents generated in word processors, for example, to SGML/XML.</td>
<td><a href="http://etd.vt.edu/etd-ml/elements.html">http://etd.vt.edu/etd-ml/elements.html</a> (accessed July 2006)</td>
</tr>
<tr>
<td>SMIL</td>
<td>The Synchronised Multimedia Integration Language (SMIL) enables simple authoring of interactive audiovisual presentations. SMIL is typically used for &quot;rich media&quot;/multimedia presentations which integrate streaming audio and video with images, text or any other media type. SMIL is an easy-to-learn HTML-like language, and many SMIL presentations are written using a simple text-editor.</td>
<td><a href="http://www.w3.org/SMIL/">http://www.w3.org/SMIL/</a> (accessed July 2006)</td>
</tr>
</tbody>
</table>
MathML 2.0, a W3C Recommendation was released on 21 Feb 2001. A product of the W3C Math working group, MathML is a low-level specification for describing mathematics as a basis for machine to machine communication. It provides a much needed foundation for the inclusion of mathematical expressions in Web pages.

Table 1: XML based languages

4.3 XML Schema

An XML schema is an XML document defining the content and structure of one or more derived XML documents. Generally, a schema is a model for describing the structure and content of data. XML Schema is a content modelling language as well as an application of XML that applies only to XML-related languages and documents. In particular, an XML Schema describes a model for a whole class of XML documents. The model describes the possible arrangement of elements, their attributes and text that would be present in a schema-valid document.

As discussed in section 4.2.1, XML was developed from SGML or GML. SGML specifies the DTD (document type declaration) as part of a grammar specification [46]. However, because SGML with its associated DTDs are regarded as too cumbersome for the specification of data exchange vocabularies on the Web, XML Schema was defined by the W3C to replace DTDs [95]. For completeness the next section will briefly discuss DTDs.

4.3.1 Document Type Declaration (DTD)

SGML introduced the concept of the document type declaration or DTD as a mechanism to describe the structure and content for derived documents. Specifically, a DTD in an XML (or HTML) document provides a specification of the elements, attributes, comments, notes, and entities contained in the
document [83]. It also indicates the relationship between these elements within the document [69].

DTDs, however, have several disadvantages, some of which are listed below [69, p.105]:

- DTDs have their own syntax (EBNF (Extended Backus Naur Form)) that differs from XML. This means that the same tools cannot be used to process documents and their document models.
- DTDs have limited ability to describe elements and their attributes, an example being that it is not possible to indicate whether character data should be numbers, date format or currency
- It is difficult to specify the cardinality of subelements in DTDs. It is possible to specify "one or more" of a subelement, but support for the specification of any other type of cardinality is limited
- Lastly but probably most significant, DTDs have limited support for namespaces meaning that they can’t define or restrict the content of elements based on context sensitivity.

It is possible to revise or extend the DTD specification as inherited from SGML to address the above issues, but this would require the revision of the SGML standard’s DTD language. The W3C opted for the development of XML Schema specifically for XML documents to overcome some of the shortcomings of DTD [71].

Using DTD

This section provides a short summary and some examples of DTD notation. The discussion is by no means exhaustive and the purpose is only to give an indication of the DTD syntax and how it is used.

A DTD specifies the grammar of an XML document and there can only be one DTD per XML document. A document type declaration is specified in the prolog of an XML document. The document type is specified either in the XML document itself (internal DTD) or referenced as an external document with the standalone attribute (external DTD) in the prolog of the
XML document [46, 69, p.117].

An example of the usage of DTDs are shown below. The XML file hallo.xml has a root element Greeting and this XML file can be validated against hallo.dtd:

```xml
<?xml version="1.0"?>
<!DOCTYPE Greeting SYSTEM "hallo.dtd">
  <Greeting>
    Hallo World!
  </Greeting>
```

The DTD file hallo.dtd simply specifies the element as:

```xml
<!ELEMENT Greeting (#PCDATA)>
```

For the hallo DTD, any XML content as shown below will be valid:

```xml
  <Greeting> any text but no markup </Greeting>
```

The following example would be invalid:

```xml
  <Greeting>
    <aTag>various text</aTag>
    <someEmptyTag/>
  </Greeting>
```

**DTD Elements**

DTDs define five different types of element content [32, p.3.08]:

- Any elements where there is no restriction on the element’s content,
- Empty elements where the element cannot store any content,
- Character Data where the element can only contain a text string,
- Elements where the element may only contain child elements and
Mixed element content where the element contains both a text string and child elements.

An example indicating that the Book element has two children namely Title and Author is shown below:

```xml
<!ELEMENT Book (Title, Author)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
```

#PCDATA stands for parsed character data which is any well-formed text string not containing special characters.

**DTD Element Attributes**

To enforce any attribute properties on an XML element, attribute-list declarations must be added to the document’s DTD [32, p.3.16-17]. Attribute-lists:

- lists the names of all attributes associated with a specific element;
- specifies the data type of an attribute;
- indicates whether an attribute is required or optional; and
- provides a default value for the attribute where necessary.

The syntax for declaring a list of attributes is

```xml
<!ATTLIST element attribute1 type1 default1
          attribute2 type2 default2
          attribute3 type3 default3 ... >
```

An example of a Customer element that must have a CustomerID attribute with a value ID is shown below:

```xml
<!ATTLIST Customer CustomerID ID #REQUIRED>
```

The purpose of DTD and XML Schema is the same, but their approach differs. Table 2 represents an adaption from Carey [32, p.4.14] and highlights some of the essential differences between XML Schema and DTD.
### Table 2: Comparing XML Schema and DTD

<table>
<thead>
<tr>
<th>XML Schema</th>
<th>DTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML based, written in XML</td>
<td>Written in different format (EBNF)</td>
</tr>
<tr>
<td>Schema can be validated by XML</td>
<td>Format not supported by XML parser</td>
</tr>
<tr>
<td>parser</td>
<td></td>
</tr>
<tr>
<td>Supports more than 40 data types</td>
<td>Supports ten data types</td>
</tr>
<tr>
<td>Supports the creation of customised</td>
<td>No customised data types</td>
</tr>
<tr>
<td>data types</td>
<td></td>
</tr>
<tr>
<td>Easily handles mixed element content</td>
<td>Difficult to specify mixed element content</td>
</tr>
<tr>
<td>Schemas can be attached to namespaces</td>
<td>DTDs cannot be associated with a namespace</td>
</tr>
<tr>
<td>No support for entities</td>
<td>Entity support</td>
</tr>
</tbody>
</table>

DTDs are still used with regards to the specification of XML documents. However, for the validation of XML documents on the Web, the W3C hopes to eventually completely replace DTDs with XML Schema declarations [95]. It is however noteworthy that the adherence of XML as a subset of the SGML standard was the reason for DTD validation. If any new grammar as a subset of SGML is defined or developed, it will again use DTD as a validation mechanism. **XML Schema** is a subset of XML and is only applicable to XML [46, 71].

### 4.3.2 XML Schema

When the XML 1.0 Recommendation was endorsed by the W3C in 1998 [113], the shortcomings of DTD were known. The XML Schema Working Group was specifically formed by the W3C to develop an XML Schema Language. As a result the W3C endorsed the XML Schema Part 1: Structures [96] and the XML Schema Part 2: Datatypes [97] in 2001.

A **schema** is a model for describing the structure and content of data, and XML Schema was developed as a content modelling language and an applica-
tion of XML, not SGML. XML Schema therefore applies only to XML-related languages and documents [32, 69].

An XML Schema describes a model for a whole class of XML documents [95]. The model describes the possible arrangement of elements, their attributes and text that would be present in a schema-valid document. The schema-models are described in terms of constraints where a constraint defines what can appear in a given document. Content model constraints define the elements that can appear, as well as the number and type of components, the order they appear in and whether they are required or optional. Datatype constraints describe the units of data that the schema considers valid [32, 69, 95].

A schema defines a class of XML documents, and therefore the term instance document is often used to describe an XML document that conforms to a particular schema. The instance document of the schema represents a specific instance of the structure that is specified in the schema document, and it contains relevant content in the structure. [32, 95].

4.3.3 XML Schema documents

This section discusses the format and structure of XML Schema documents, enabling the reader to form a concept of what is required when developing an XML Schema grammar.

An XML Schema document is in the first place a well-formed XML document. Therefore the first line is mandatory and is the XML declaration required in all XML documents [95].

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema example -->
<!-- Next we declare the root element = schema -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  document specifications
</xsd:schema>
```
The line beginning with `<xsd:schema ..>` is the start tag for the `schema element` which is the root element of the XML document. The `<schema>` element is therefore the parent element of all other elements or `subelements` in the schema. The start tag may also include attributes defining for instance the namespaces and unqualified local elements. The `document specifications` are schema components that declare the schema, in other words the properties and contents of the data that the instance documents should contain [32, 69, 95].

An XML Schema consists of components. A Schema component is the generic term for the building blocks that comprise the abstract data model of the schema [96]. An XML Schema is therefore a set of `schema components`, such as:

- Simple type definitions
- Complex type definitions
- Attribute declarations
- Element declarations

Each of the elements in a schema usually has a prefix `xsd:` which is associated with the XML Schema namespace through the declaration, `xmlns:xsd="http://www.w3.org/2001/XMLSchema"` that appears in the schema element. The prefix `xsd:` is used by convention to denote the XML Schema namespace, although any prefix can be used [32, 69, 95].

XML Schema differentiate between simple types (which cannot have element content and cannot carry attributes) and complex types (which allow elements in their content and may carry attributes). There is also a distinction between definitions which create new types (both simple and complex), and declarations for document instances which allow elements and attributes with specific names and types (both simple and complex) to appear [95].

**XML Schema simple types**

Any XML schema declares several elements and attributes that have simple
types. Some of these simple types, such as string and decimal, are built-in to XML Schema, while others are derived from the built-in’s. Both built-in simple types and their derivations can be used in all element and attribute declarations [69, 95–97].

There are 44 built-in data types that are part of the XML Schema specification. The built-in data types consist of either primitive also called base types, or derived types. The primitive types consist of 19 fundamental data types that are not defined in terms of other data types. They are also called atomic types, and the value of an atomic type is indivisible from XML Schema’s perspective [32, 97].

The derived data types are a collection of 25 data types that were created based on the 19 primitive types [32, 95]. An example of a built-in type declaration is presented below:

    <element name="ID-number" type="positiveInteger"/>

This example declares that the ID-number element is limited to only positive integers.

New simple types are defined by deriving them from existing simple types (built-in’s and derived). A new simple type can be defined by restricting an existing simple type (meaning the legal range of values for the new type are a subset of the existing type’s range of values). The simpleType element is used to define and name a new simple type. The restriction element is used to indicate the existing (base) type, and to identify the constraints or the range of values.

For example, defining a new type of integer called newInteger with a range of values is between 10000 and 99999 (inclusive) is declared by using the built-in simple type integer [96, 97]:

    <xsd:simpleType name="newInteger">
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="10000"/>
        </xsd:restriction>
    </xsd:simpleType>
User-derived data types are therefore specifically created by the schema author, and consist of the built-in types and/or other user-derived types.

XML Schema also defines other simple types such as union types and list types (for instance a list of Provinces of South Africa as shown below). This is in addition to the atomic types described above [96, 97].

Elements whose type is NineSAProvinces must have nine items, and each of the nine items must be one of the (atomic) values of the enumerated type SAProv.

Atomic types and list types enable an element or an attribute value to be one or more instances of one atomic type. In contrast, a union type enables an element or attribute value to be one or more instances of one type drawn from the union of multiple atomic and list types [96, 97].

XML Schema complex types

As discussed, XML Schema differentiates between complex types (which may have attributes and/or elements in their content), and simple types which cannot have element content and cannot carry attributes [95].

New complex types are defined using the complexType element and such
definitions typically contain a set of element declarations, element references, and attribute declarations. The next section discusses element declarations. The declarations are not themselves types, but rather an association between a name and the constraints which govern the appearance of that name in documents governed by the associated schema [96, 97].

Any addition of attributes or child elements to simple types will result in the specification of a complex type. If we want to specify a currency, for example, we have to declare a complex type. Decimal is a simple type, and the next example, obtained from W3C [95] defines a new complex type:

```xml
<xsd:element name="internationalPrice">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="currency" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

In the example the complexType element is used to start the definition of a new (anonymous) type. To indicate that the content model of the new type contains only character data and no elements, we use a simpleContent element. Finally, the new type is derived by extending the simple decimal type. The extension consists of adding a currency attribute using a standard attribute declaration. The internationalPrice element declared in this way can appear in an instance document as follows [96, 97]:

```xml
<internationalPrice currency="EUR">423.46</internationalPrice>
```

Elements that contain subelements or carry attributes are said to have complex types, whereas elements that contain numbers (and strings, and dates, etc.) but do not contain any subelements are said to have simple types. Some elements have attributes; attributes always have simple types [96, 97].
The next section focus on element declarations.

**XML Schema element declarations**

XML Schema allows for the specification of two types of elements: complex and simple (note: simple and complex *elements* vs. simple and complex *types*). A *simple type* element contains only character data. A *complex type* element is an element that has attributes or is a parent for child elements [96, 97].

A simple element is declared as follow:

```xml
<element name="name" type="type" />
```

Or if we use a namespace prefix (in this case *xsd*):

```xml
<xsd:element name="name" type="xsd:type" />
```

A complex element contains attributes and/or other elements, for example:

```xml
<element name="name">
  <complexType>
    compositors
    element declarations
    compositors
    attribute declarations
  </complexType>
</element>
```

In this example, *name* is the name of the element in the instance document, *element declarations* are simple type declarations for each child element, *compositors* define how the list of elements are organised (of value either sequence, choice or all), and *attribute declarations* are declarations that define the attributes of the element [82, 96, 97].
One advantage XML Schema has over DTDs is the ability to specify mixed content elements. If an element contains both text and child elements, the *mixed* attribute is added to the *complexType* tag [32]. For example, the following XML content:

```xml
<Description>
    Author <Name>Charles Darwin</Name> wrote
    <Book>On the Origin of Species</Book>
    in the 19th century.
</Description>
```

can be declared in XML Schema as

```xml
<element name="Description">
    <complexType mixed="true">
        <sequence>
            <element name="Name" type="string" />
            <element name="Book" type="string" />
        </sequence>
    </complexType>
</element>
```

XML Schema is very versatile and allows the content text to appear before, between and after child elements as in the example above [32, 69].

**XML Schema element occurrences**

To specify the number of times an element occur, one uses the *minOccurs* and *maxOccurs* attributes, for example [32]:

```xml
<element name="BookTitle" type="string" minOccurs="1" maxOccurs="unbounded" />
```
In this example there must be at least one BookTitle element, and there is not an upper limit to the number of occurrences. If minOccurs is set to 0, the declared item is optional, if minOccurs and maxOccurs are not specified, their value is assumed to be 1. The minOccurs and maxOccurs attributes can also be used in compositors to specify the repeat occurrence of entire sequences of items [97].

**XML Schema attributes**

In XML Schema, any element that contains a attribute, is also a complex type. The syntax for an attribute is:

```xml
<attribute name="name" type="type" use="use"
    default="default" fixed="fixed" />
```

where name is the name of the attribute, type is the data type, use states whether the attribute is required or not (possible values are required, optional and prohibited), default is the default value of the attribute and fixed is a fixed value for the attribute [32, 97].

An attribute must be declared with element it pertains to. If an element is empty, such as:

```xml
<Student No="123456" Gender="male" />
```

an attribute is declared using the same syntax as child elements:

```xml
<element name="Student">
    <complexType>
        <attribute name="No" type="string" />
        <attribute name="Gender" type="string" />
    </complexType>
</element>
```
If an element contains child elements in addition to attributes, the attributes 
are placed after the element declarations. For example:

```xml
<Student No="123456" Gender="male">
  <Name>John Smith</Name>
  <Age>18</Age>
</Student>
```

is specified

```xml
<element name="Student">
  <complexType>
    <sequence>
      <element name="Name" type="string"/>
      <element name="Age" type="positiveInteger"/>
    </sequence>
    <attribute name="No" type="string"/>
    <attribute name="Gender" type="string"/>
  </complexType>
</element>
```

### 4.3.4 Namespaces and XML Schemas

XML Schemas provide namespace support for the qualification of elements\(^4\). 
The target namespace declared in the prolog of an XML document provides 
the information for the processor to check any instance document to see if 
it validates against a schema. A namespace therefore indicates to the XML 
processor that the definitions of elements and other datatypes in the in the 
 schema are adopted from the declared namespace [24].

\(^4\)Namespace concepts are discussed in section 4.1 on page x
The namespace declaration can be done as an attribute of the schema element as shown [24, 69].

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
```

This code snippet declares the attribute xmlns:xsd with value "http://www.w3.org/2001/XMLSchema". Breaking the statement into parts:

- The xmlns name indicates that it is a namespace declaration
- The xsd portion is the abbreviation used to relate the respective elements and datatypes to the namespace
- The "http://www.w3.org/2001/XMLSchema" portion is the unique URI or URL identifying the namespace

In this case the W3C Namespace Recommendation is the namespace used by the XML processor for the definitions of elements and other datatypes. When the processor encounters any datatypes with the prefix xsd: (the prefix represents the URL) the meaning for those datatypes are identical to the definition found in the W3C Recommendation [24, 96].

After the namespace has been defined, the element used in conjunction with the abbreviation, becomes a unique element. An example is `<xsd:sequence>`, where the local part of the unique name is sequence [32].

Namespaces do not have to be declared explicitly with prefixes, as in:

```xml
<schema xmlns:xs="http://www.w3.org/2001/XMLSchema" .../>
```

Without a prefix, the shown URL is presumed to be the default namespace to which an element or attribute without a prefix in the document is associated [32].

## 5 Layer 3 / Layers 3a and 3b

RDF (Resource Descriptive Framework) and RDF Schema technologies reside on Layer 3 (refer to Figures 1 and 2). With the positioning of RDF Schema, Layer 3b, above RDF M&S (Model and Syntax), Layer 3a, in Figure 2, Berners-Lee [14] emphasises the importance of a vocabulary descrip-
tion mechanism on top of the RDF data model as part of the Semantic Web layered architecture.

5.1 RDF

RDF (Resource Descriptive Framework) is a W3C Recommendation designed to standardise the definition and usage of meta-data, or in the context of the Semantic Web, a mechanism to capture data about web resources. The W3C describes the Resource Description Framework (RDF) as a language for representing meta-data or information about resources on the Web [100]. RDF is intended for the exchange of meta-data about resources between applications, but without loss of meaning [43].

The purpose of RDF is to declare meta-data that is machine-processable. RDF provides a mechanism to declare statements that describe resources by means of a basic data model. A statement describes an entity (resource) in terms of properties, which have values. Furthermore, an RDF statement is a subject, predicate, object triple [42, 45, 100]. The subject is the resource of the statement. The predicate is the property or characteristic of the subject specified by the statement (examples include creator, creation-date, or language), and the value of the property is the object.

The W3C RDF specification was developed to provide a common framework so that application developers can use common RDF parsers and processing tools as is the case with XML using common XML parsers and tools. The design of RDF, according to the W3C RDF Concepts [105], is intended to meet the following goals:

- having a simple data model
- having formal semantics and provable inference
- using an extensible URI-based vocabulary
- using an XML-based syntax
- supporting use of XML schema datatypes
- allowing anyone to make statements about any resource

The next section discusses the RDF model in more detail. The reader not
interested in the detail of RDF could continue to RDF Schema, section 5.2 on page xli.

5.1.1 RDF Model

As stated previously, RDF is specifically intended for the representation of meta-data about Web resources. RDF describes the resources in terms of their properties and property values. More specifically, RDF makes statements about resources by using an subject, predicate, object triple [42, 100]. The statement http://www.thesis.org/SWTechnologies.html has a creator whose value is Student AJG is described in RDF terms as follows:

- the subject is the URL http://www.thesis.org/SWTechnologies.html
- the predicate is the word creator
- the object is the phrase Student AJG

The RDF statement above can be depicted in a graph as shown in figure 3.

![RDF Graph]

**Figure 3**: An RDF graph for a basic statement: http://www.thesis.org/SWTechnologies.html has a creator whose value is Student AJG
An RDF statement or subject-predicate-object-triple, (sometimes also called an object-attribute-value-triple) can also be indicated as $A(O,V)$, which means that an object $O$ has an attribute $A$ with value $V$. This relationship can also be modelled in a graph as a labelled edge $A$ between two nodes, $O$ and $V$: $[O] \rightarrow A \rightarrow [V]$ as shown by the arrow in the graph in Figure 3 [33, 100]. The underlying datamodel of RDF can be labelled a hypergraph, with each statement being a predicate-labelled link between object and subject. The graph is a hypergraph since each node can itself again contain an entire graph [29].

RDF uses URI\(^5\), or more specifically URIs\(^6\) (URI References) as its mechanism for identifying the subjects, predicates, and objects in statements. A URIref is a URI with an optional fragment identifier at the end. To clarify, the URI reference http://www.thesis.org/SWTechnologies.html#section2 consists of the URI http://www.thesis.org/SWTechnologies.html and the fragment identifier section2. RDF URIrefs are specified using Unicode\(^6\) characters. In general, RDF defines a resource as anything that is identifiable by a URI reference. RDF can therefore be used to represent information about anything that can be identified on the Web, even when it cannot be directly retrieved on the Web.

The RDF graph in figure 4 was obtained from the RDF Primer of the W3C [100] and it is used to further illustrate how RDF describes meta-data. This figure represents the group of statements there is a Person identified by http://www.w3.org/People/EM/contact\#me, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr.

In figure 4, URIrefs are used to identify:

> individuals, e.g., Eric Miller, identified by http://www.w3.org/People/EM/contact\#me
> kinds of things, e.g., Person, identified by http://www.w3.org/2000/10/swap/pim/contact\#Person
> properties of those things, e.g., mailbox, identified by http://www.

---

\(^5\)See section 3.2 on page viii for a discussion of URIs.
\(^6\)See section 3.1 on page vii for a discussion of Unicode
Figure 4: An RDF Graph describing Eric Miller [99]

w3.org/2000/10/swap/pim/contact#mailbox

values of those properties, e.g. mailto:em@w3.org as the value of the mailbox property (RDF also uses character strings such as "Eric Miller", and values from other datatypes such as integers and dates, as the values of properties)

The purpose of RDF is to declare meta-data that is machine-processable and RDF therefore uses XML as syntax. RDF defines a specific XML markup language, RDF/XML, to represent RDF information. The graph in figure 4 is represented in RDF/XML [9] as follows:

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:swap="http://www.w3.org/2000/10/swap/pim/contact#">
    <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
        <contact:fullName>Eric Miller</contact:fullName>
    </contact:Person>
</rdf:RDF>
```
RDF allows objects and values to be interchanged. This means that labelled edges in a graph can be chained. It is also possible to make an RDF statement or triple the object of another statement, allowing for the nesting of RDF statements [29, 42].

### 5.1.2 Blank RDF nodes

An RDF graph can also be used for expressions that are not necessarily triples. The representation of such a structure requires the use of a blank node. In the following example (obtained from the W3C RDF Primer [99]), RDF is used for the representation of John Smith’s address. John’s address needed to be recorded as a structure consisting of separate street, city, state, and postal code values.

![RDF graph with a blank node](image)

**Figure 5:** An RDF graph with a blank node [99]

Figure 5 is a valid RDF graph and uses a node without a URIref to stand
for the concept of "John Smith’s address". The node provides the necessary connectivity between the various other parts of the graph. However, some form of explicit identifier for that node is needed in order to represent this graph as triples.

One possibility of representation of the triples corresponding to Figure 5 is:

```
exstaff:85740  extems:address  _:johnaddress .
_:johnaddress  extems:street  "1501 Grant Avenue" .
_:johnaddress  extems:city  "Bedford" .
_:johnaddress  extems:state  "Massachusetts" .
_:johnaddress  extems:postalCode  "01730" .
```

where _:johnaddress (a blank node identifier) refers to the presence of the blank node.

There is often a need for statements about collections of resources. For this purpose RDF defines the concept of a container [100, 105]. RDF defines three types of containers that can represent collections of resources or literals:

- **Bags** are unordered lists. Bags don’t enforce set semantics, so a value can appear several times in a Bag.
- **Sequences** are ordered lists. Like Bags, Sequences permit duplicate values.
- **Alternatives** are lists from which the property can use only one value.

The previous sections gave a rudimentary description of the RDF model and some of the basic ways in which it represents meta-data. More detailed information can be obtained from the W3C RDF documentation set [99, 101–105].

### 5.1.3 RDF Vocabularies

RDF is used to define vocabularies. The term *vocabulary* is used to refer to a RDF graph depicting a set of meta-data or data definitions. A RDF vocabulary is therefore a set of URIrefs and its relationships defined for some specific purpose, such as a shared domain definition for participants.
of a domain, or even the set of URIs defined by RDF for its own use, for instance in RDF Schema [99, 101, 103].

RDF provides a way to make statements that RDF applications can process although the specific application does necessarily "understand" the statements contained in the definition. An example might assist in explaining the concepts. A user could search the Web for a review of all the Scuba diving sites in South Africa. This user might then create an average rating for each diving site and publish the information on the Web. Another Web site could take the list of diving sites and create a "Top Ten Tourist Adventure Sites" page. Because a shared vocabulary exists, different users might built a commonly-understood and increasingly-powerful (as additional contributions are made) information base about topics on the Web.

One of the most quoted examples of a RDF vocabulary is at the DCMI (Dublin Core Meta-data Initiative) [40]. The DCMI is an organisation dedicated to promoting the widespread adoption of interoperable meta-data standards and developing specialised meta-data vocabularies for describing resources that enable more intelligent information discovery systems. The mission of DCMI is to make it easier to find resources using the Internet through the following activities:

- Developing meta-data standards for discovery across domains,
- Defining frameworks for the interoperation of meta-data sets, and,
- Facilitating the development of community- or disciplinary-specific meta-data sets that are consistent with items 1 and 2.

In addition to the DMCI, the RDF Resources page compiled by Beckett [8] contains a list of several RDF applications and projects which is updated regularly.

section 5.2 discusses RDF Schema which is an example of a RDF vocabulary.
5.1.4 The Application of RDF, XML and XML Schema on the Semantic Web

RDF is preferred above XML for the description of meta-data on the Semantic Web because RDF specifies a data model and the information specified by RDF can be mapped directly and unambiguously to the model [41]. This model is part of an external specification and several generic, third-party parsers are already available to interpret such an RDF document [100]. An RDF model separates semantic and syntactic information. Because the RDF model is a W3C standard, all users that adopt RDF will be able to make this distinction [76].

XML Schema is not necessarily used in conjunction with RDF since XML Schema is a language for restricting the syntax of XML applications [43]. However, XML Schema might be useful to specify certain datatypes or similar restrictions on the syntax. XML Schema is therefore not used to control the syntax of RDF, rather the syntax as contained in an XML document [76].

5.2 RDF Schema

RDF Schema specifies extensions to RDF that are used to define common vocabularies in RDF meta-data statements. RDF itself provides the data model and does not prescribe any application-specific classes and properties, this is accomplished by RDF Schema. RDF Schema specifies extensions to RDF, these are provided by the RDF Vocabulary Description Language 1.0: RDF Schema [103]. RDF Schema provides a predefined, basic type system for RDF models, thus extending RDF by assigning an externally specified semantics to specific resources. RDF Schema expressions are valid RDF expressions, and therefore RDF Schema is a semantic extension of RDF [28, 51]. Software that can interpret RDF can also be used to interpret an RDF Schema implementation although it will not attach the intended meaning to the built-in schema definitions [42].

The RDF vocabulary description strategy contained in RDF Schema ac-
knowledges that there are many techniques that enable description of meaning of classes and properties. To extend the description of meaning, ontology languages (such as DAML+OIL, OIL and OWL), inference rule languages and other formalisms are used [42].

In the definition of a particular vocabulary for RDF data, RDF Schema assists in a similar way to the function of XML schema that provides a vocabulary-definition facility for XML. RDF schema provides a predefined, basic type system for RDF models [41]. RDF Schema extends RDF by giving an externally specified semantics to specific resources, e.g., to rdfs:subclassOf or to rdfs:Class. It is because of this external semantics that RDF Schema is useful [29].

RDF Schema does not specify a vocabulary of application-specific classes, but it describes the mechanisms necessary to specify such a vocabulary [100]. It provides the facilities needed to describe application-specific classes and their associated properties. For example, the property ex:breedType should be used in describing an ex:Dog. RDF Schema can therefore also be described as providing a typing mechanism for RDF, similar to the type systems of object-oriented programming languages. An example (obtained from [103]), states that a class ex:Dog might be defined as a subclass of ex:Mammal which is a subclass of ex:Animal, meaning that any resource which is in class ex:Dog is also implicitly in class ex:Animal as well [27, 94].

As mentioned, RDF Schema facilities are themselves specified in the form of an RDF vocabulary; that is, as a specialised set of predefined RDF resources. Usually, resources in the RDF Schema vocabulary have URIrefs with the prefix http://www.w3.org/2000/01/rdf-schema# or rdfs:. Any vocabulary that is defined using RDF Schema is also a legal RDF graph and any software that can interpret RDF can therefore also be used to interpret an RDF Schema implementation although it will not attach the intended meaning to the built-in Schema definitions. For this an RDF application must be written to process an extended language that includes not only the rdf: vocabulary, but also the rdfs: vocabulary with the built-in meanings [28, 103].
5.2.1 Summary of RDF Schema Constructs

In order to illustrate the nature of RDF Schema as well as the way in which RDF is utilised, an RDF Schema summary obtained from the W3C Resource Description Framework (RDF) Model and Syntax Specification [103] is presented in tables 3 and 4. These tables presents an overview of the vocabulary of RDF, formed from the original vocabulary defined in the RDF Model and Syntax Specification [94] as well as the classes and properties that originated with RDF Schema [103].

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:Resource</td>
<td>The class resource, everything.</td>
</tr>
<tr>
<td>rdfs:Literal</td>
<td>The class of literal values, e.g. textual strings and integers.</td>
</tr>
<tr>
<td>rdf:XMLLiteral</td>
<td>The class of XML literals values.</td>
</tr>
<tr>
<td>rdfs:Class</td>
<td>The class of classes.</td>
</tr>
<tr>
<td>rdf:Property</td>
<td>The class of RDF properties.</td>
</tr>
<tr>
<td>rdfs:Datatype</td>
<td>The class of RDF datatypes.</td>
</tr>
<tr>
<td>rdf:Statement</td>
<td>The class of RDF statements.</td>
</tr>
<tr>
<td>rdf:Bag</td>
<td>The class of unordered containers.</td>
</tr>
<tr>
<td>rdf:Seq</td>
<td>The class of ordered containers.</td>
</tr>
<tr>
<td>rdf:Alt</td>
<td>The class of containers of alternatives.</td>
</tr>
<tr>
<td>rdfs:Container</td>
<td>The class of RDF containers.</td>
</tr>
<tr>
<td>rdfs:ContainerMembershipProperty</td>
<td>The class of container membership properties.</td>
</tr>
<tr>
<td>rdf:List</td>
<td>The class of RDF Lists.</td>
</tr>
</tbody>
</table>

Table 3: RDF Classes

<table>
<thead>
<tr>
<th>Property name</th>
<th>Comment</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdfs:type</td>
<td>The subject is an instance of a class.</td>
<td>rdfs:Resource</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:subClassOf</td>
<td>The subject is a subclass of a class.</td>
<td>rdfs:Class</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>The subject is a subproperty of a property.</td>
<td>rdf:Property</td>
<td>rdf:Property</td>
</tr>
</tbody>
</table>

xliii Semantic Web Technologies
<table>
<thead>
<tr>
<th>rdf:domain</th>
<th>A domain of the subject property.</th>
<th>rdf:Property</th>
<th>rdf:Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:range</td>
<td>A range of the subject property.</td>
<td>rdf:Property</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:label</td>
<td>A human-readable name for the subject.</td>
<td>rdf:Resource</td>
<td>rdf:Literal</td>
</tr>
<tr>
<td>rdf:comment</td>
<td>A description of the subject resource.</td>
<td>rdf:Resource</td>
<td>rdf:Literal</td>
</tr>
<tr>
<td>rdf:member</td>
<td>A member of the subject resource.</td>
<td>rdf:Resource</td>
<td>rdf:Resource</td>
</tr>
<tr>
<td>rdf:first</td>
<td>The first item in the subject RDF list.</td>
<td>rdf:List</td>
<td>rdf:Resource</td>
</tr>
<tr>
<td>rdf:rest</td>
<td>The rest of the subject RDF list after the first item.</td>
<td>rdf:List</td>
<td>rdf:List</td>
</tr>
<tr>
<td>rdf:seeAlso</td>
<td>Further information about the subject resource.</td>
<td>rdf:Resource</td>
<td>rdf:Resource</td>
</tr>
<tr>
<td>rdf:isDefinedBy</td>
<td>The definition of the subject resource.</td>
<td>rdf:Resource</td>
<td>rdf:Resource</td>
</tr>
<tr>
<td>rdf:value</td>
<td>Idiomatic property used for structured values.</td>
<td>rdf:Resource</td>
<td>rdf:Resource</td>
</tr>
<tr>
<td>rdf:subject</td>
<td>The subject of the subject RDF statement.</td>
<td>rdf:Statement</td>
<td>rdf:Resource</td>
</tr>
<tr>
<td>rdf:predicate</td>
<td>The predicate of the subject RDF statement.</td>
<td>rdf:Statement</td>
<td>rdf:Resource</td>
</tr>
<tr>
<td>rdf:object</td>
<td>The object of the subject RDF statement.</td>
<td>rdf:Statement</td>
<td>rdf:Resource</td>
</tr>
</tbody>
</table>

**Table 4: RDF Properties**

RDF is used to represent information in the Web using a simple data model using an subject, predicate, object triple. RDF Schema is a semantic extension of RDF [101] which means that RDF Schema expressions are valid RDF expressions in the form of the subject-predicate-object triple. In RDF Schema an agreement exists on the semantics of certain terms and thus on the interpretation of certain statements [28].

A benefit of this approach is that the description of existing resources can be extended [101]. The RDF vocabulary description strategy also acknowled...
edges that there are many techniques through which the meaning of classes and properties can be described. This is where ontology languages such as DAML+OIL\textsuperscript{7} and OWL\textsuperscript{8}, inference rule languages and other formalisms (for example temporal logics) will be used to extend the specification of meaning.

6 Layer 4 / Layers 4a and 4b

In V1 (Figure 1) Ontology vocabulary is depicted on Layer 4, whilst this layer is separated as Ontology, Layer 4a, and Rules, Layer 4b, in V2 (Figure 2). In Figure 2, Berners-Lee [14] acknowledges that an ontology is a knowledge representation language capturing the syntax (ontology) as well as semantics (rules) of a specific domain [67, 68]. OWL is the W3C technology representing an Ontology vocabulary or Ontology associated with this layer, whilst W3C research efforts aim to establish the technologies required for the implementation of the Rules to be contained in this layer [58]. It is noted that the terminology on this layer differs from the three preceding layers, because functionality rather than technology is mentioned.

6.1 Ontology Vocabulary / Ontology

An ontology specifies a machine readable vocabulary in computer systems technology descriptions. Generally it is defined as a shared, formal, explicit specification or conceptualisation of a particular domain [29, 41, 53]. An ontology typically describes a hierarchy of resource concepts within a domain and associates each concept’s crucial properties with it. Ontologies are used to define and manage concepts, attributes and relationships between concepts in a precise manner [30].

The concept of an ontology was inherited from philosophy and has only recently become commonplace in computer systems technology descriptions where an ontology specifies a machine readable vocabulary. Ontologies on the

\textsuperscript{7}See section 6.1.2 on page xlvii for a discussion of DAML + OIL.

\textsuperscript{8}See section 6.1.1 on page xlvii for a discussion of OWL.
Semantic Web and expert systems or AI (artificial intelligence) technologies of the 1980s are based on the same motivations but they emerged from different architectures which implies that the technologies are deployed or applied differently [76].

Ontologies assist in creating a common understanding for communication between people and computer applications. The Web Ontology Working Group at the W3C identified six main areas for Ontology use within the *OWL Use Cases and Requirements document* [52], namely:

- **Web Portals**
  - Categorisation rules used to enhance search

- **Multimedia Collections**
  - Content-based searches for non-text media

- **Corporate Web Site Management**
  - Automated Taxonomical Organisation of data and documents
  - Mapping Between Corporate Sectors (mergers!)

- **Design Documentation**
  - Explication of "derived" assemblies (e.g. the wing span of an aircraft)
  - Explicit Management of Constraints

- **Intelligent Agents**
  - Expressing User Preferences and/or Interests
  - Content Mapping between Web sites

- **Web Services and Ubiquitous Computing**
  - Web Service Discovery and Composition
  - Rights Management and Access Control

From the above we can deduce that ontologies will be used in applications that search across or merge information from diverse communities. XML, DTDs and XML Schemas are sufficient for exchanging data between parties who have agreed to common definitions beforehand, but the lack of semantics prevent this when new XML vocabularies emerge. RDF and RDF Schema specifies simple semantics associated with identifiers. With RDF Schema, one can define classes with multiple subclasses and super classes,
as well as properties with sub-properties, domains, and ranges. Therefore, RDF Schema is regarded as a simple ontology language. However, RDF Schema is limited, for instance, RDF Schema cannot specify that Person and Car classes are disjoint, or that a string quartet has exactly four musicians as members. In order to achieve interoperability between numerous, autonomously developed and managed schemas, richer semantics are needed [6, 60, 92].

It is foreseen that ontologies will play a crucial role in enabling Web-based knowledge processing, sharing, and reuse between applications in the future, and therefore the establishment of Semantic Web. The W3C developed OWL as a Web Ontology Language which satisfies most of the current foreseen requirements of an ontology specification for the Semantic Web [52, 54].

6.1.1 OWL

The acronym OWL was derived from Web Ontology Language. The W3C Working Group decided against the direct acronym WOL and decided on OWL instead. OWL as a Web Ontology Language means that it was designed to be compatible with the architecture of the Web, specifically the Semantic Web [68]. The next section provides some insight into the development of OWL and the relationship between known Ontology language efforts such as OIL and DAML+OIL.

6.1.2 OWL, OIL and DAML+OIL

Research to represent ontologies with languages dates back to the work on frame-languages in the early days of AI. With the emergence of the Web, efforts of designing ontology-representation languages that are Web enabled started [47]. Several initiatives were created around this research topic. Onto-Knowledge [75] is an European IST project that focused on the development of Semantic Web technology, specifically the development of ontology-based tool environments for knowledge management. These tools have to deal with large numbers of heterogeneous, distributed, and semi-structured
documents typically found in large company intrados and the Web. The project aimed to develop:

- a toolset for semantic information processing and user access;
- OIL, an ontology-based inference layer on top of the Web;
- an associated methodology and validation by industrial case studies.

OIL (Ontology Inference Layer) as well as several tools and architectures providing a Web-based representation and inference layer for ontologies were results of this work. A complete RDF Schema implementation for OIL was developed [29, 47, 48, 75].

OIL stands in a similar relationship to RDF Schema than RDF Schema to RDF in that it defines semantics to extend RDF Schema. It is possible to capture meaning in OIL that cannot be captured in RDF Schema, but this extension is done in such a way the the RDF Schema document is still valid [29, 41].

OIL adopted the essential modelling primitives of frame-based systems into its language, basing its formal semantics on DL (Description Logics). DL models knowledge in terms of concepts, role restrictions and derived classification taxonomies, and provide a principled method to reason with the presented knowledge. In addition, OIL was developed with support for the XML and RDF syntaxes. OIL has a well-defined syntax in XML based on a DTD and a XML schema definition, and is also defined as an extension of RDF.

Concurrently with the On-to-Knowledge initiative, DARPA (the Defense Advanced Research Projects Agency), developed DAML (the DARPA Agent Markup Language) by extending RDF with more expressive constructs aimed at facilitating agent interaction on the Web. Because of similar work these organisations decided to co-operate [39]. DAML was a significantly funded project for the development of semantic web technology. As a result of this co-operative research, a new language called DAML+OIL was defined. To continue work in this area, the Joint EU/US ad hoc Agent Markup Language Committee was formed and some of the OIL standards were adjusted to conform with an international standard backed up by the US defence de-
partment [39]. Fensel [47] describes DAML+OIL as an expressive description logic supported with web syntax. It does not provide layered sub languages with different complexity nor language primitives that are defined around a modelling paradigm.

Because of its inheritance from DAML and OIL, DAML+OIL builds on existing Web technologies such as XML, URI and RDF. DAML+OIL markup is a specific kind of RDF markup. RDF, in turn, can be serialised to XML with its associated Namespaces and URIs [67, 93].

It was inevitable that the W3C, because of its mandate, set up a Semantic Web Activity initiative. The Activity was initiated early 2001 with a Web-Ontology (WebOnt) Working Group tasked to investigate all the efforts and consolidated them into a Web Ontology Language [68, 98]. OWL is the result of the activities of this group and the OWL Recommendation consisting of six documents was issued by the W3C in February 2004 [106]. Some of the results of this work are discussed in the next section, section 6.1.3.

6.1.3 OWL Specification

OWL includes concepts and design aspects of DAML+OIL. OWL extends RDF Schema in order to express complex relationships between different classes specified in RDF Schema. In addition OWL enhances the specification of constraints applicable to classes and properties [52, 68].

The W3C OWL document set consists of six documents aimed at different audiences or users, namely (1) a presentation of the OWL use cases and requirements [52], (2) an overview document which briefly explains the features of OWL [68], (3) a comprehensive OWL guide that provides a walk-through of the features of OWL [87], (4) a reference document that provides the details of every OWL feature [7], (5) a test case document [34], as well as (6) a document presenting the semantics of OWL and details of the mapping from OWL to RDF [80].

Within this document set, OWL Specifies three sublanguages. These languages were specified to be increasingly expressive and ontology designers
should choose the most appropriate version:

- **OWL Lite** supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. Owl Lite also has a lower formal complexity than OWL DL, see the section on OWL Lite in the OWL Reference for further details.

- **OWL DL** supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.

- **OWL Full** is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full. Complete OWL Full implementations do not exist with the writing of this document.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not [68, 87].

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
Every valid OWL Lite conclusion is a valid OWL DL conclusion.
Every valid OWL DL conclusion is a valid OWL Full conclusion.

6.1.4 Structure of an OWL Ontology

A typical OWL ontology begins with a namespace declaration. Namespaces provide a means to unambiguously interpret identifiers and make the rest of the ontology presentation more readable indicating precisely what vocabularies are being used. Therefore a standard initial component of an ontology includes a set of XML Namespace declarations enclosed in an opening rdf:RDF tag [7, 52, 68].

Once namespaces are established, a collection of assertions about the ontology grouped under an owl:Ontology tag are normally included. These tags support for instance a name or reference for the ontology, comments, version control and might also specify the inclusion of other ontologies [7]. At this stage in the ontology document it is appropriate to specify the basic elements of the ontology. Elements of an OWL ontology basically consists of classes, properties, instances of classes, and relationships between these instances. OWL provides the language components essential to define these elements [7, 52, 68].

OWL support the definition of simple named classes with the Class and rdfs:subClassOf constructs. An ontology needs to specify the basic concepts in the domain. These concepts should correspond to classes that are the roots of various taxonomic trees. Every class is a member of the class owl:Thing. Thus each user-defined class is implicitly a subclass of owl:Thing. Domain specific root classes are defined by declaring a named class. In addition, OWL also defines the empty class, owl:Nothing.

A class definition has two parts: a name introduction or reference and a list of restrictions. Each of the immediate contained expressions in the class definition further restricts the instances of the defined class [7, 52, 68, 87].

In addition to classes OWL allows for the description of members of classes.

---

9 Section 4.1 on page x discusses Namespaces.
These can be described as individuals in the universe of things described by the ontology. An individual is minimally introduced by declaring it to be a member of a class. It is important to note the difference between a class and an individual in OWL. A class is simply a name and collection of properties that describe a set of individuals. Individuals are the members of those sets. Thus classes should correspond to naturally occurring sets of things in a domain of discourse, and individuals should correspond to actual entities that can be grouped into these classes [87].

OWL properties are used to assert general facts about the members of classes and specific facts about individuals. A property is a binary relation. Two types of properties are distinguished [87]:

- datatype properties that are used to describe relations between instances of classes and RDF literals and XML Schema datatypes
- object properties that are used to describe relations between instances of two classes. Note that the name object property is not intended to reflect a connection with the RDF term rdf:object.

OWL includes a number of mechanisms that are used to further specify properties. It is possible to specify property characteristics which provides a powerful mechanism for enhanced reasoning about a property. In addition to designating property characteristics, it is possible to further constrain the range of a property in specific contexts in a variety of ways. This is done with property restrictions. For the detail of OWL, a reader is referred to the W3C OWL documentation set [7, 34, 52, 68, 80, 87].

At present a number of OWL ontologies are available on the Web, including an ontology library at DAML [39], which contains about more than 300 examples written in OWL or DAML+OIL. In addition, several ontologies described as large ontologies have been released in OWL. These include a cancer ontology in OWL developed by the US NCI’s (National Cancer Institute) Center for Bio-informatics, which contains about 17,000 cancer related terms and their definitions [73]. The NCI Thesaurus is a public domain description logic-based terminology produced by the National Cancer Institute. It is deep and complex compared to most broad clinical vocabularies, imple-
menting rich semantic interrelationships between the nodes of its taxonomies. The semantic relationships in the Thesaurus are intended to facilitate translational research and to support the Bio-Informatics infrastructure of the Institute.

To realise the vision of the Semantic Web, the ontologies of the Semantic Web need to be widely shared and re-used. An example of such re-use as portrayed from Smith et al. [87] states that a user might adopt a date ontology from one source and a physical location ontology from another and then extend the notion of location to include the time period during which it holds. Much of the effort of developing an ontology is devoted to hooking together classes and properties in ways that maximise implications. Simple assertions about class membership should have broad and useful implications. This is a difficult part of ontology development. If an existing ontology that has already undergone extensive use and refinement can be found, it makes sense to adopt it [68, 87].

6.1.5 Description Logics

Any discussion of ontologies or OWL would be incomplete without a section on DL (Description Logics). DL is the logical foundation of all three the OWL sublanguages [87].

To discuss the relationship between OWL and DL, it is necessary to investigate knowledge representation languages. A knowledge representation language is specified when both the syntax and the semantics of the language is described [67]. In the syntax definition the legal statements in the language are defined, and the semantic description specifies each legal statement’s intended meaning. The semantics can be formally specified in multiple ways within a logical framework such as with FOL (First Order Logic) or the various Description Logics [4, 41].

The OWL language provides a specific subset in the form of OWL DL to provide a language subset that has the computational properties necessary for reasoning systems [7, 68, 87]. DL offers a formal foundation for frame-
based systems where meaning is provided by interpretations that define the formal semantics of the logic [49] as well as support for some automated reasoning (e.g. class consistency checking) [29, 67].

Where the central modelling primitives of predicate logic are predicates, in frame-based and object-oriented approaches the central modelling primitives are classes (or frames) with certain properties, also referred to as attributes, that do not have a global scope. Description Logics describe knowledge in terms of concepts and role restrictions that are used to automatically derive classification taxonomies. In spite of some of the discouraging theoretical complexity of results with regards to expressing structured knowledge and accessing and reasoning with it in a principled way, there are now efficient implementations for DL languages [47].

OWL incorporates the essential modelling primitives of frame-based systems, and therefore the formal semantics of Description Logics.

Grau [49] summarizes Description Logics as follow:

*Description logics (DL) are a set of knowledge representation formalisms, whose semantic characterisation is based on standard first-order logics. Meaning is provided by interpretations, which define the formal semantics of the logic. An interpretation in DL is a mathematical structure $I = \{I^1, I\}$ consisting of:

- A nonempty set $I^1$, called the domain of the interpretation.
  The domain is divided into two disjoint sets:
  - The abstract domain is the set of all the individuals
  - The concrete domain is composed of data values and is used to integrate datatypes in description logics

- An interpretation function $I$ that maps:
  - Every concept (class) name to a subset of $I^1$
  - Every role (property) name to a subset of $I^1 \times I^1$
  - Every individual to an element of $I^1$

The interpretation function can be extended to complex concepts*
and roles and can be used to provide meaning to axioms in the knowledge base.

As the purpose of this section is to provide an overview of the Semantic Web technologies, the reader is referred to one of the most substantial books on the subject, *The Description Logic Handbook, Theory, implementations and application* by Baader et al. [4] for an in depth discussion of DL.

6.2 Rules

*Rules* is not depicted in V1 (Figure 1). In contrast, Layer 4 is separated as *Ontology* (Layer 4a) and *Rules* (Layer 4b) in V2 (Figure 2). With the issue of the *OWL Recommendation* by the W3C in 2004, *Rules* was stated as a requirement, and it is plausible to speculate that this is the reason for its inclusion into V2. Rules are defined as executable pieces of declarative knowledge, important in managing complex and dynamic operations [56].

6.2.1 SWRL

At present there is a W3C proposal for a Semantic Web Rule Language (SWRL) based on a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Data-Log RuleML sublanguages of the Rule Markup Language [59]. The proposal extends the set of OWL axioms to include Horn-like rules. It thus enables Horn-like rules to be combined with an OWL knowledge base. A high-level abstract syntax is provided that extends the OWL abstract syntax described in the *OWL Semantics and Abstract Syntax* document [80]. An extension of the OWL model-theoretic semantics is also given to provide a formal meaning for OWL ontologies including rules written in this abstract syntax [59, 111].

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold [59]. An OWL ontology in the abstract syntax contains a sequence of axioms and facts. Axioms may be of various
kinds, e.g., subClass axioms and equivalentClass axioms. It is proposed to extend this with rule axioms: axiom ::= rule

At present SWRL is a W3C member submission. The W3C Team evaluates this submission in the context of W3C activities and the work is therefore in process [59, 111].

A W3C Workshop on Rule Languages for Interoperability was held during April 2005 to gather data and explore options for establishing a standard web-based language for expressing rules. More than a dozen use cases were presented for rule language standardisation, and about a half-dozen candidate technologies were presented and discussed. The workshop gave many indications that a W3C Recommendation here would be useful. As a result of the workshop discussions, the Semantic Web Activity group of the W3C established a RIF (Rule Interchange Format) Working Group at the end of 2005 to assist with the establishment of a core rule language plus extensions which together allow rules to be translated between rule languages and thus transferred between rule systems [111].

7 Layer 5

The Semantic Web architectures in Figures 1 and 2 depicts Logic or Logic framework residing above the ontology layer.

7.1 Logic / Logic framework

Logic is regarded as the foundation of knowledge representation languages, and it is required to provide the highly expressive language constructs in which knowledge can be captured in a transparent way. A logic framework provides a well-established formal semantics which assigns unambiguous meaning to logical statements. Without a logic framework, inferencing on the Semantic Web will not be possible.

McGuinness et al. [67] defines a knowledge representation language as a language that specifies both the syntax and the semantics of the language. In
the syntax definition the legal statements in the language are defined, and the semantic description specifies each legal statement’s intended meaning. The semantics can be formally specified in multiple ways with a logical framework such as FOL (First Order Logic) or the various Description Logics [4, 41]. The V1 and V2 Semantic Web architectures in Figures 1 and 2 depict Logic or Logic framework residing above the ontology layer even through logic is an aspect included in the OWL Ontology Language through DL formalisms [79]. Therefore, the positioning of this layer represents the notion that a richer logical language should be provided on top of the Ontology language, which provides additional mechanisms for the capturing of reasoning formalisms. Proposals for web logic languages may therefore employ a special semantics, such as minimal model semantics, to make inference more amenable to computer implementation. The technologies that might implement Logic or Logic Framework are at present largely research efforts by institutions such as the W3C.

8 Layer 6

In the V1 and V2 Semantic Web architectures of Berners-Lee (Figures 1 and 2), Proof resides on Layer 6.

8.1 Proof

Proof as concept exists within the theorem proving domain, for instance as applied in artificial intelligence [81]. To support Semantic Web proof scenarios, proof languages were developed. A proof language determines the validity of a specific statement. An instance thereof generally consists of a list of inference items used to derive the information in question, as well as the associated trust information of each item [3, 76]. A Semantic Web will probably not require proof generation and in general proof validation will be adequate. The search for and generation of a proof for an arbitrary question, is typically an intractable process for many real world problems, and the Semantic Web does not require this to be solved
[79]. For perceived Semantic Web applications construction of a proof is performed according to constrained rules, and only the validation thereof is required from other parties. For example, when a user is granted access to a web site, an accompanying document explains to the web server why the user in question should be granted access. Such proof for example, could be a chain of assertions and reasoning rules with pointers to all supporting material [10].

Even though the trust and proof aspects have not really been explored and is considered beyond current research [79], it is a crucial aspect of the Semantic Web. To illustrate the concept with a use case, we adopt an example of Palmer [76]:

*If one person says that x is blue, and another says that x is not blue, doesn’t the whole Semantic Web fall apart? Of course not, because*

a) applications on the Semantic Web at the moment generally depend upon context, and

b) because applications in the future will generally contain proof checking mechanisms, and digital signatures.

To support scenarios such as the one above, the notion of proof languages has to be developed. A proof language is simply a language that is used to prove whether or not a specific statement is true. An instance of a proof language will generally consist of a list of inference items used to derive the information in question, and the trust information for each of those items that can then be checked [76].

9 Layer 7

*Trust resides on Layer 7 in Figures 1 and 2.*
9.1 Trust

Semantic Web interaction requires different collaborators to communicate, implying that they have to determine how to trust one another, as well as how to establish the trust levels of acquired information [91]. The trust levels of information depend on (1) the source of the information, (2) whether the source can be trusted, as well as (3) whether the source is who it claims to be, in other words, the authenticity and trustworthiness of the source. When dealing with user interactions on the Web, McKnight, Choudhury and Kaemar [70] define the term trust as the belief that another entity is benevolent, competent, honest, or predictable in a given situation. Trust also includes the participants’ willingness to depend on one another in a specific interaction.

Within the Semantic Web the concepts trust and proof are dependent on the interaction context. However, an all-encompassing definition of context is problematic [20]. An appropriate meaning of context is therefore explicated by means of the following example:

A user on the Semantic Web receives data from a friend regarding the best music performances. The data can be trusted as it originates from a known (implying verified) friend, whose musical interests are familiar. It is thus possible to use the data because the user either shares or disagrees with the musical tastes of the friend.

Within the domain of the Semantic Web, context therefore assists applications or users regarding the trustworthiness and usefulness of information [20]. Context will enable applications or users to decide whether or not received data can be trusted and therefore how to handle information [90]. Trust levels of information might include user groups and shared context. If a group or organisation develop a Semantic Web application, then any user’s trust of that application, amongst other things, depends upon how much the entire group can be trusted. The information context would therefore create an conceptual environment where the Semantic Web operate and interact
intuitively, without having to rely on complex authentication and checking [76].

10 Vertical Layers

In versions V1 and V2 of the Semantic Web layered architecture of Berners-Lee Digital Signature is associated with layers three to six (see Figures 1 and 2).

10.1 Digital Signatures

The DSS (Digital Signature Standard) is a cryptographic standard or a particular application of public key cryptography promulgated by NIST (National Institute of Standards and Technology) [35]. A digital signature is an electronic signature that can be used to authenticate identity. Digital signatures are easily transportable, cannot be imitated, and can be automatically time-stamped. A digital signature can be used with any kind of message, whether it is encrypted or not [10, 21].

XMLDSig (XML Digital Signatures), also called XML Signatures, is a joint IETF/W3C standard that specifies how to digitally sign and verify a signature of a XML data object [3]. XMLDSig enables digital signatures on arbitrary digital content (XML or non-XML) [66]. XML Signatures are digital signatures designed for use in XML transactions [86].

For the Semantic Web a digital signature is a mechanism used to unambiguously verify an identity such as the author of a document [76]. The implementation of digital signatures on the Semantic Web could result in a system which can express and reason about relationships across the whole range of public-key based security and trust systems. It is foreseen that XMLDSig will be used in many phases of semantic knowledge management systems, such as the authenticity verification for retrieved/updated knowledge and involved intermediaries, among others [5].
10.2 Encryption

The V1 version of the Semantic Web layered architecture of Berners-Lee (Figure 1) does not depict Encryption. It was however added in version V2 depicted in Figure 2 where it is associated with layers three to six, along with Signature.

Encryption is an effective way to achieve data security. XMLEnc (XML Encryption) is a W3C standard that specifies how to encrypt/decrypt an XML-formatted data object. XMLEnc supports end-to-end (as opposed to point-to-point) encryption of an XML object, which can be the whole or a part of an XML document. The document can be transmitted in XML or non-XML syntax [63]. indexXMLEnc

On the Semantic Web it is foreseen that encryption would be used in knowledge storage, internal/external knowledge transfer as well as authentication [14, 66]. At present, the implementation of the Encryption functionality within the Semantic Web remains largely a research effort [79].

11 The Technologies of the Later Versions of the Layered Architecture

Tim Berners-Lee published additional versions of the architecture in 2005 and 2006, referred to in this thesis as V3 and V4 respectively [15, 16]. A third version (V3) of the Semantic Web architecture was introduced in his keynote presentation at the 2005 World Wide Web Conference [15], and the latest version (V4) of the architecture was presented in a AAAI2006 keynote presentation [16]. These versions are reproduced as Figures 6 and 7 with 'Layer’ captions corresponding to the first two versions as depicted in Figures 1 and 1.

In this section the technologies of the last versions (V3 and V4) of the architecture is discussed by relating it to the previous versions V1 and V2 in Figures 1 and 2 on pages vii. In section 11.1 the focus is on the general adaptations that appear in versions V3 and V4 when referring to
Figure 6: The V3 version of the Semantic Web Architecture [15].

Figure 7: The V4 version of the Semantic Web Architecture [16].
versions V1 and V2 discussed in section 2. Section 11.2 discusses the specific adaptations to Layer 4 in versions V3 and V4 with the addition of Rules and Rules:RIF and SPARQL.

### 11.1 General Adaptations

Versions V3 and V4 of the Semantic Web layered architecture proposed by Berners-Lee [15, 16] introduce some noteworthy adaptations, which are discussed in this section. These versions were introduced as part of keynote presentations and thus the specific meaning of the architecture adaptations (as in the case with the previous versions of the architecture) has not been discussed in literature.

Layer 1 in versions V3 and V4 still depicts URI and Unicode. Layer 2 in V3 in Figure 6 is similar to previous versions, however, Namespaces is omitted from the V4 architecture in Figure 7. Generally, Namespaces are regarded as included in the XML specifications\(^{10}\) and it is proposed that the omission is regarded as superficial.

On Layer 3a in both V3 and V4 additional descriptions of RDF are included, namely either RDF Core in V3 or Data interchange: RDF in V4. However, the specific meaning thereof is unclear. RDF provides a meta-data data model for data description. The Data interchange: RDF caption in V4 support the functionality description of RDF as mechanism for data interchange on the Semantic Web.

On Layer 3b in both V3 and V4 RDF Schema or RDF-S still resides above RDF, however SPARQL is added. SPARQL provides a mechanism to query RDF data. SPARQL is discussed in more detail in section 11.2.1. In V3 Figure 6 DLP bit of OWL/Rdf is added as a layer above RDF Schema but the intended meaning of this addition is unclear. DLP bit of OWL/Rdf is discussed in more detail in section 11.2.2.

In V3 and V4 on Layer 4, OWL and ontology: OWL replace Ontology and Rules or Rules:RIF is introduced. Rules are therefore moved down from

\(^{10}\)Refer to section 4 on page x for a discussion of XML, Namespaces and XML Schema.
Layer 4b in V2 of Figure 2 to Layer 4 on the same level as OWL in V3 and V4. OWL is the W3C recommendation that are used for the creation of ontologies and it is discussed in section 6.1 on page xlv. RIF is an acronym for Rule Interchange Format that is at present a Working Group of the W3C. RIF is discussed further in section 11.2.3.

On Layer 5 of both V3 and V4 Proof is extended down to above Rules, thus residing both above both the Rules: RIF layer as well as above the Unifying Logic layer. The purpose of this layer seem to be to assist with the integration of different formalisms. In V4 the caption Unifying Logic replaces Logic Framework but it is plausible to speculate that the intention of the layer remain the same and this adoption is a matter of semantics. Trust still resides on Layer 7 in both V3 and V4. In addition, a layer (Layer 8) is added above Layer 7 in V4 depicting User Interface and applications that seems to represent the notion that all applications and user interfaces of the Semantic Web will reside above Layer 7. Layer 8 is problematic since it neither depicts technologies nor functionalities required for the realisation of the Semantic Web, but rather where Semantic Web applications would reside, and thus the terminology of this layer deviates from all previous versions.

The vertical layers in Figure 2 (Digital Signatures and Encryption) are still depicted in V3 in Figure 6. However, in V4 (Figure 7) they have been replaced with a single vertical layer called Crypto. What is significant is that the Crypto vertical layer does not start on top of Layer 2 (XML) as in previous versions such as V3, but reside alongside the whole stack. It is however not clear what is the meaning of Crypto and it is possible to speculate that it is a combined layer reflecting the security needs of the Semantic Web architecture.

11.2 Layer 4 Adaptations

Layer 4 in V3 (Figure 6) and V4 (Figure 7) depicts the addition of new technologies to the architecture, namely SPARQL, RIF and DLP bit of OWL/Rul. There is however a deviation from the previous versions V1 and
V2 of the Semantic Web layered architecture in the introduction of these technologies since none of them are W3C recommendations whilst previously only W3C recommendations were depicted. The remainder of this section discusses SPARQL, RIF and DLP bit of OWL/Rul, as well as highlight the present status of these technologies.

11.2.1 SPARQL

As mentioned in section 5.1, RDF is a data model and is considered to be a flexible and extensible way to represent information about Web resources. It is used to represent diverse information such as personal information or meta-data about any digital artifacts. RDF provides a mechanism to integrate diverse sources of information.

In order to use RDF, a query language is considered to be an urgent requirement [108, 110]. Without such a language it will not be possible to efficiently extract information from an RDF data store. SPARQL has as goal a standardised query language for RDF data. SPARQL Protocol And RDF Query Language is the result of the W3C endeavour to develop a query language and it offers developers and end users a way to write and to consume the results of queries across this wide range of information. Used in conjunction with a common protocol, applications can access and combine information from across the Web [110]. SPARQL is at present a W3C candidate recommendation.

11.2.2 DLP bit of OWL/Rul Layer

DLP bit of OWL/Rul is added as a layer above RDF Schema but the intended meaning of this addition is unclear.

DLP (Description Logic Programs) is described as a mechanism to transform any OWL ontology into a (disjunctive) logic program [44]. However, Hitzler, Haase, Krotzsch, Sere and Studer [57] states that an entirely satisfactory definition of DLP is not straightforward. According to them, DLP was originally conceived in [50] as a fragment of OWL DL, but subsequently it has
been a source of confusion and controversy.

The addition of the DLP bit of OWL/Rul layer into V3 seems to be to incorporate the work of Grosof, Horrocks, Volz and Decker [50] who argue for a mechanism to interoperate, semantically and inferentially, between rules (as RuleML Logic Programs) and ontologies (as OWL/DAML+OIL Description Logic). They define DLP as an intermediate knowledge representation contained within this intersection between rules and ontologies. The intention is for DLP to extend Datalog to include knowledge representation. In this case it is plausible to speculate that, in order for Rules to reside above RDF Schema, it was considered to necessary to extract the DLP bit of OWL/Rul from OWL and insert this into a layer above RDF Schema.

However, this was since refuted by Horrocks, Parsia, Patel-Schneider and Hendler [58] who argue that (1) Datalog cannot be layered on DLP due to semantic differences, unless a different semantics for DLP (DLP-Datalog) is adopted, (2) DLP-Datalog cannot be layered on RDF Schema and is not compatible with RDF semantics, and (3) OWL cannot be layered on top of DLP-Datalog.

Datalog is a (function-free) variant of Horn predicate logic which is widely used for deductive databases [74]. It is also described as a query and rule language for deductive databases that, syntactically, is a subset of Prolog [36]. Datalog was popular in academic database research but never succeeded in becoming part of a commercial database system. Advantages of Datalog over SQL such as the clean semantics or recursive queries were not sufficient. Datalog is however used in knowledge representation applications, such as KAON-2, a Datalog system that represents an approximation of OWL. In addition, Ontoprise implemented OntoBroker, which is based on a Datalog reasoner [1].

In summary, the DLP bit of OWL/Rul layer in the V3 version (Figure 6) of the architecture introduced Datalog extensions to the architecture underlying OWL [31]. However, Horrocks et al. [58] pointed out that OWL and Datalog are not semantically compatible. This is probably the reason for the removal of this layer in the subsequent V4 version (Figure 7).
11.2.3 RIF

The Semantic Web Activity group of the W3C established a RIF (Rule Interchange Format) Working Group at the end of 2005 in order to specify a core rule language with extensions, which together, allow rules to be translated between rule languages and thus transferred between rule systems [112].

Rule-languages and rule-based systems have been used in computer science and information technology in several types of applications such as expert systems or deductive databases. Automated inferencing based on symbolic representations has a rich history and continues to be a key technology driver [22, 56]. Due to the innovations prevalent in the Semantic Web domain, there is now a need for research in this technology area [112]. Some issues with regards to automated inferencing will require further research, but others can be addressed by enabling existing rule-based technologies to interoperate according to standards-based methodologies and processes. This is the basic goal of RIF (the Rule Interchange Format) [109]. The RIF Working Group attempts to devise the required standards that are useful in the present situation as well as being easily extensible in order to deal with the evolution of rule technology and other enabling technologies [112].

The RIF Working Group published a new Public Working Draft of RIF called RIF Use Cases and Requirements [109]. This version includes requirements in addition to use cases, as well as goals and a rough cut of the space of rule systems likely to be addressed in Phase 1. RIF is at present a W3C member submission as working draft and the W3C Team is evaluating this submission within the context of W3C activities. The W3C has not endorsed the submission of RIF yet.

12 Conclusion

At present, the information overload experienced by information technology users, specifically on the Web, necessitates the introduction of automated information management functionality, one realisation of which constitutes
the envisioned Semantic Web. In addition, the Semantic Web and its associated technologies are permeating various fields and domains within the ICT (Information and Communications Technology) domain. In order to understand the nature and impact of these technologies, specifically with regards to the envisioned Semantic Web by Berners-Lee and the relation with the proposed versions of his architecture, it is necessary to discuss these technologies and concepts. In this report, the research provided a starting point to assimilate Semantic Web terminology and associated concepts. In addition, the information about the technologies presented in this report serves as a foundation to any subsequent discussions of the different versions of the Semantic Web layered architecture.
References


# Index

ANSI, xv  
Application of XML, xvi  
CDF, xix  
CERN, xv  
Character Encoding, vii  
CML, xix  
Context, lxii  
DAML, l  
DAML+OIL, xlix, l  
DARPA, l  
Datalog, lxvii  
Description Logics, lv, lviii  
Diacritics, ix  
Digital Signatures, lxi  
DL, lv, lviii  
DLP, lxv-lxvii  
Document Type, xv  
Document Type Declaration, xxi  
Document type declaration, xviii  
DSS, lxii  
DTD, xv, xvii, xxi, xxii  
DTD Element Attributes, xxiii  
DTD Elements, xxiii  
Element, xxx  
Element attributes, xviii  
Element hierarchy, xviii  
Element, parent, xvii  
Encryption, lxii  
ETD-ML, xx  

<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensible Markup Language</td>
<td>xiii</td>
</tr>
<tr>
<td>First Order Logic</td>
<td>lv, lviii</td>
</tr>
<tr>
<td>FOL</td>
<td>lv, lviii</td>
</tr>
<tr>
<td>Frame based models</td>
<td>lv</td>
</tr>
<tr>
<td>GML</td>
<td>xv</td>
</tr>
<tr>
<td>HTML</td>
<td>xv, xvi</td>
</tr>
<tr>
<td>Hypertext Markup Language</td>
<td>xv, xvi</td>
</tr>
<tr>
<td>Identifier</td>
<td>x</td>
</tr>
<tr>
<td>IETF</td>
<td>x</td>
</tr>
<tr>
<td>Internationalised Resource Identifiers</td>
<td>xi</td>
</tr>
<tr>
<td>Internet Engineering Task Force</td>
<td>x</td>
</tr>
<tr>
<td>IRI</td>
<td>xi</td>
</tr>
<tr>
<td>Logic</td>
<td>lviii</td>
</tr>
<tr>
<td>Logic Framework</td>
<td>lviii</td>
</tr>
<tr>
<td>Markup Language</td>
<td>xiv</td>
</tr>
<tr>
<td>MathML</td>
<td>xx</td>
</tr>
<tr>
<td>Metalanguage</td>
<td>xiv</td>
</tr>
<tr>
<td>Namespace</td>
<td>xii</td>
</tr>
<tr>
<td>Namespaces</td>
<td>xii, xxvi, xxxiv</td>
</tr>
<tr>
<td>Namespaces and XML Schema</td>
<td>xiii</td>
</tr>
<tr>
<td>Naming conflicts</td>
<td>xii</td>
</tr>
<tr>
<td>OIL</td>
<td>xlix</td>
</tr>
<tr>
<td>On-to-Knowledge</td>
<td>xlix</td>
</tr>
<tr>
<td>Ontologies</td>
<td>definition, xlvi</td>
</tr>
<tr>
<td>Ontologies, usage of</td>
<td>xlvi</td>
</tr>
<tr>
<td>Ontology</td>
<td>xlvi</td>
</tr>
</tbody>
</table>

lxxxiii
<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ontology Vocabulary, xlvi</td>
<td></td>
</tr>
<tr>
<td>OWL, xlviii, xlix</td>
<td></td>
</tr>
<tr>
<td>OWL and RDF Schema, li</td>
<td></td>
</tr>
<tr>
<td>OWL classes, lii</td>
<td></td>
</tr>
<tr>
<td>OWL DL, lii</td>
<td></td>
</tr>
<tr>
<td>OWL Full, lii</td>
<td></td>
</tr>
<tr>
<td>OWL Lite, lii</td>
<td></td>
</tr>
<tr>
<td>OWL ontologies on the Web, liv</td>
<td></td>
</tr>
<tr>
<td>OWL properties, liv</td>
<td></td>
</tr>
<tr>
<td>OWL Specification, li</td>
<td></td>
</tr>
<tr>
<td>OWL sublanguages, lii</td>
<td></td>
</tr>
<tr>
<td>OWL, Ontology, lii</td>
<td>v</td>
</tr>
<tr>
<td>Proof, lix</td>
<td></td>
</tr>
<tr>
<td>Proof languages, lx</td>
<td></td>
</tr>
<tr>
<td>RDF, xxxv, xlii, lxv</td>
<td></td>
</tr>
<tr>
<td>RDF data model, xxxvi</td>
<td></td>
</tr>
<tr>
<td>RDF graph, xxxviii</td>
<td></td>
</tr>
<tr>
<td>RDF Schema, xliii, xliiv</td>
<td></td>
</tr>
<tr>
<td>RDF Schema Classes, xlv</td>
<td></td>
</tr>
<tr>
<td>RDF Schema Constructs, xliv</td>
<td></td>
</tr>
<tr>
<td>RDF Schema Properties, xlvi</td>
<td></td>
</tr>
<tr>
<td>RDF triple, xxxviii</td>
<td></td>
</tr>
<tr>
<td>RDF Vocabularies, xli</td>
<td></td>
</tr>
<tr>
<td>Resource, x</td>
<td></td>
</tr>
<tr>
<td>Resource Descriptive Framework, xxxv</td>
<td></td>
</tr>
<tr>
<td>RFC3986, x</td>
<td></td>
</tr>
<tr>
<td>RIF, lxviii</td>
<td></td>
</tr>
<tr>
<td>Root Element, xviii</td>
<td></td>
</tr>
<tr>
<td>Rules, lvi</td>
<td></td>
</tr>
<tr>
<td>Semantic Web Technologies, vii</td>
<td></td>
</tr>
<tr>
<td>Semantic Web V1 Architecture, vii</td>
<td></td>
</tr>
<tr>
<td>Semantic Web V2 Architecture, vii</td>
<td></td>
</tr>
<tr>
<td>Semantic Web V3 Architecture, liii</td>
<td></td>
</tr>
<tr>
<td>Semantic Web V4 Architecture, liii</td>
<td></td>
</tr>
<tr>
<td>SGML, xv, xxi, xxii</td>
<td></td>
</tr>
<tr>
<td>SMIL, xx</td>
<td></td>
</tr>
<tr>
<td>SPARQL, lxvi</td>
<td></td>
</tr>
<tr>
<td>Standardised General Markup Language, xv</td>
<td></td>
</tr>
<tr>
<td>SWRL, lvii</td>
<td></td>
</tr>
<tr>
<td>Trust, lx</td>
<td></td>
</tr>
<tr>
<td>Unicode, vii, xi, lxiv</td>
<td></td>
</tr>
<tr>
<td>Unicode Consortium, vii</td>
<td></td>
</tr>
<tr>
<td>Unicode Encoding Mechanisms, ix</td>
<td></td>
</tr>
<tr>
<td>Unicode Standard, vii</td>
<td></td>
</tr>
<tr>
<td>Uniform Resource Identifier, xi</td>
<td></td>
</tr>
<tr>
<td>Uniform Resource Locator, xi</td>
<td></td>
</tr>
<tr>
<td>Uniform Resource Name, xi</td>
<td></td>
</tr>
<tr>
<td>URI, x, xi, lxiv</td>
<td></td>
</tr>
<tr>
<td>URL, xi</td>
<td></td>
</tr>
<tr>
<td>URN, xi</td>
<td></td>
</tr>
<tr>
<td>UTF-16, ix</td>
<td></td>
</tr>
<tr>
<td>UTF-32, ix</td>
<td></td>
</tr>
<tr>
<td>UTF-8, ix, xi</td>
<td></td>
</tr>
<tr>
<td>XML, xiii, iv, xvi, xlii</td>
<td></td>
</tr>
<tr>
<td>XML Document, xvii</td>
<td></td>
</tr>
<tr>
<td>XML Document Structure, xvi</td>
<td></td>
</tr>
<tr>
<td>XML Document: Elements, xvii</td>
<td></td>
</tr>
<tr>
<td>XML Document: Valid, xviii</td>
<td></td>
</tr>
<tr>
<td>XML Entity, xvii</td>
<td></td>
</tr>
<tr>
<td>XML entity, parsed, xvii</td>
<td></td>
</tr>
</tbody>
</table>
XML entity, unparsed, xvii
XML History, xiv
XML Languages, xix
XML Parser, xvi
XML Processor, xvi
XML Schema, xiii, xx, xxv, xlii
XML Schema attributes, xxxii
XML Schema complex types, xxix
XML Schema documents, xxvi
XML Schema element declarations,
 xxx
XML Schema element occurrences,
 xxxii
XML Schema simple types, xxvii
XML Schemas and Namespaces,
 xxxiv
XML, Element Attributes, xvii
XML, Element Hierarchy, xviii
XML, well-formed, xviii
XMLDSig, lxii