A LEARNING OBJECT HARVESTING MODEL AND A SAMPLE APPLICATION

AHMET SOYLU

IŞIK UNIVERSITY

2008

A LEARNING OBJECT HARVESTING MODEL AND A SAMPLE APPLICATION

AHMET SOYLU

M.S., Computer Engineering, Işık University, 2008

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY
2008

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

A LEARNING OBJECT HARVESTING MODEL AND A SAMPLE APPLICATION

AHMET SOYLU

APPROVED BY:

Prof. Selahattin Kuru (Thesis Supervisor)	(Işık University)	_____
Assist. Prof. Taner Eskil	(Işık University)	_____
Assoc. Prof. Seyhun Altunbay	(Işık University)	_____

APPROVAL DATE : 03 / 06 / 2008

A LEARNING OBJECT HARVESTING MODEL AND A SAMPLE APPLICATION

Abstract

The aim of this thesis is to enable harvesting of Learning Objects which are embedded in web pages. Two main challenges have been identified which are interoperability and semantics. For this purpose, several learning object metadata standards such as XML, XSLT, RDF, SPARQL etc. and several Learning Object Metadata standards have been investigated for compatibility with this harvesting approach, and then a light weight Application Profile and Microformat for Learning Objects has been proposed. Additionally a web service has been created which uses XSLT/GRDDL to extract Learning Objects in different web pages, and uses SQI target for retrieval facility with a more complex query language called SPARQL. Final work is a sample application; a search client employing created SQI service for search and retrieval of Learning Objects.

ÖĞRENİM NESNESİ TOPLAMA MODELİ VE ÖRNEK UYGULAMA

Özet

Bu tezin amacı web sayfalarına gömülmüş öğrenim nesnelerinin ayıklanmasına imkan veren bir model yaratmaktır. Aşılması gereken iki önemli problem belirlenmiştir, bunlar birlikte çalışabilirlik ve anlamsallıktır. Bu amaç için birlikte çalışabilirlik ve anlamsallık çerçevesinde XML, XSLT, RDF, ve SPARQL gibi çeşitli teknolojiler ve öğrenim nesnesi üstveri standartları bu modele uygunluk açısından incelenmiştir. Daha sonra öğrenim nesneleri için dar kapsamlı bir Microformat önerisi getirilmiştir. Ek olarak XSLT/GRDDL teknolojileri kullanılarak farklı web sayfalarından öğrenme nesnelerini ayıklayabilen ve bunların SPARQL sorgulama dili ile sorgulanabilmesini sağlayan bir SQI web servisi oluşturulmuştur. Son olarak bu servis üzerinden çalışarak öğrenme nesnelerinin aranmasını ve aktarımını sağlayan bir arama motoru oluşturulmuştur.

Acknowledgements

I thank Prof. Dr. Selahattin Kuru, my major professor and dissertation supervisor. Having the opportunity to work with him over the years was intellectually rewarding and fulfilling. I also thank Mag. Fridolin Wild, Dr. Felix Mödritscher and Steinn E. Sigurdarson for their insightful suggestions and expertise, they contributed a lot to this thesis starting from the early stages of my work, and The Scientific and Technological Research Council of Turkey (TUBITAK) for their financial support during my studies.

My special thanks go to Uğur Kırmızıbekmez and Kadir Banicar whose friendship I deeply value.

The last words of thanks go to my family. I thank my parents Mustafa Soylu and Sakine Soylu and my sister Duygu Soylu for their patience and encouragement.

Table of Contents

Abstract ii
Özet iii
Acknowledgements iv
Table of Contents v
List of Tables vii
List of Figures viii
List of Abbreviations xii
1 Introduction 1
2 Ground Web Technologies and Specifications 7
2.1 XML 8
2.1.1 How to Create XML Documents 9
2.1.2 XML Schema Definition 11
2.2 XHTML 16
2.3 XSL 18
2.3.1 XPath 18
2.3.2 XSLT 20
2.4 Web Services 23
2.4.1 WSDL 24
2.4.2 SOAP 26
2.5 SQI 27
3 Semantic Web Technologies and Specifications 31
3.1 RDF 31
3.1.1 Model, Concepts and Syntax 32
3.1.2 RDF Schema 45
3.2 SPARQL 49
3.3 GRDDL 55
4 Semantic Web and Microformats 57
4.1 Semantic Web 57

4.2 A deeper look at Microformats 61
4.2.1 POSH 62
4.2.2 Basic Principles 63
4.2.3 Design Patterns 64
4.2.4 An example Microformat 66
5 Learning Object Metadata Standards 69
5.1 Learning Standards 69
5.2 Learning Object Metadata Standards 73
5.2.1 Application Profile 74
5.2.2 IEEE LOM 76
5.2.3 ADL-Scorm – Scorm Metadata 80
5.2.4 Other LO Metadata Standards and Application Profiles 81
5.2.5 Dublin Core 82
6 Implementation 85
6.1 Application Profile Proposal 85
6.1.1 Scope and Purpose 85
6.1.2 Use Cases 86
6.1.3 Metadata Elements 87
6.2 Microformat Proposal 94
6.3 Case Example: Sematic Search Engine for LOs 98
7 Evaluation of Model and Application 106
8 Conclusion and Recommendations for Future Work 111
8.1 Overview 111
8.2 Recommedations for Future Work 112
References 115
Appendix A: CD Containing Sample Application and Tools 120
Curriculum Vitae 121

List of Tables

Table 2.1 Example Expressions for XML File in Figure 2.5 19

Table 2.2 Example Axes for XML Code in Figure 2.5 [17] 20

Table 2.3 SQI Methods 29

Table 3.1 Result of Third Query on Figure 3.8 50

Table 3.2 Result of Second Query on Figure 3.13 51

Table 5.1 Snapshot of Standard Organizations [46] 72

Table 5.2 LOM v1.0 Elements 77

Table 5.3 A Sample from LOM v1.0 [10] 78

Table 5.4 LOM v1.0 Data Types 80

Table 5.5 DC Elements Mapping over IEEE LOM Elements [52] 83

Table 5.6 DCMI-EMS Metadata Schema [58, 59] 84

Table 7.1 LOM Element Use [60] 107

Table 7.2 Evaluation of Learning Object Metadata Proposal 108

Table 7.3 Evaluation of Microformats Proposal 109

List of Figures

Figure 1.1 Graphical Representation of the Proposed Work 6

Figure 2.1 Simple XML Code..... 9

Figure 2.2 Name Spaces..... 10

Figure 2.3 XML with Schema Reference 12

Figure 2.4 Schema Definition 12

Figure 2.5 XML in Details..... 13

Figure 2.6 Schema for Figure 2.1 14

Figure 2.7 Schema for Person Name Space..... 15

Figure 2.8 Schema for Product Name Space 16

Figure 2.9 Sample XHTML Document 17

Figure 2.10 How to Use XSLT 21

Figure 2.11 Style Sheet for Figure 2.1 22

Figure 2.12 Web Service Architecture..... 24

Figure 2.13 Sample WSDL Document 25

Figure 2.14 Sample SOAP Request Message 27

Figure 2.15 Communication Between Two Repositories [20] 27

Figure 2.16 UML Class Diagram of SQI [20] 29

Figure 3.1 Sample RDF Document..... 33

Figure 3.2 Graphical Representation of RDF 33

Figure 3.3 Sample Triple Format 34
Figure 3.4 Example of Prefix Use 34
Figure 3.5 Properties as Resources 34
Figure 3.6 Resources as Properties 35
Figure 3.7 Triple Representation of Resources as Properties 35
Figure 3.8 Demonstration of Structured Property 36
Figure 3.9 Structured Properties 36
Figure 3.10 Triple Demonstration of Structured Properties 37
Figure 3.11 Blank Nodes 37
Figure 3.12 Triple Demonstration of Blank Nodes 38
Figure 3.13 Demonstration of Blank Nodes 38
Figure 3.14 Typed Literals Sample 38
Figure 3.15 Triple Demonstration of Typed Literals 39
Figure 3.16 Demonstration of Blank Nodes 39
Figure 3.17 Demonstration of “rdf:ID” Attribute 40
Figure 3.18 Demonstration of “rdf:type” Property 40
Figure 3.19 Triple Demonstration of “rdf:type” 40
Figure 3.20 “rdf:type” Demonstration 41
Figure 3.21 Demonstration of RDF Container Elements 42
Figure 3.22 Triple Demonstration of RDF Container Elements 42
Figure 3.23 Graph Demonstration of “rdf:alt” Container 42
Figure 3.24 Demonstration of Collections 43
Figure 3.25 Graph Demonstration of Collections 44
Figure 3.26 Demonstration of Collections 44
Figure 3.27 Triple Demonstration of Collections 45

Figure 3.28 Demonstration of “Literal” and “Resource” Parse Type. 45
Figure 3.29 Demonstration of RDFS 46
Figure 3.30 Graph Demonstration of Collections 47
Figure 3.31 Triple Demonstration of RDFS Classes 47
Figure 3.32 Abbreviated Demonstration of RDFS 48
Figure 3.33 Property Declarations 48
Figure 3.34 Triple Demonstration of Property Declarations 49
Figure 3.35 Sample SPARQL Queries 50
Figure 3.36 Sample SPARQL Query with Prefix 50
Figure 3.37 Sample SPARQL Query Two Patterns 51
Figure 3.38 Sample SPARQL Query Using Filter 51
Figure 3.39 Sample SPARQL Query Using Regex 52
Figure 3.40 Sample SPARQL Query Using Optional 52
Figure 3.41 Sample SPARQL Query Using Union 53
Figure 3.42 Sample SPARQL Query Using From 53
Figure 3.43 Sample SPARQL Query Using From Named 54
Figure 3.44 Sample SPARQL Query Using Modifiers and Solution Sequences 54
Figure 3.45 Sample GRDDL Use [27] 55
Figure 3.46 GRDDL Transformation via Schema 56
Figure 3.47 GRDDL Transformation via Profiles 56
Figure 4.1 Example Use of Microformats 61
Figure 4.2 HTML, POSH and Microformats [39] 62
Figure 4.3 Foundation of Microformats [39] 64
Figure 4.4 Abbr Design Pattern Sample 65
Figure 4.5 Date Time Design Pattern Sample 65

Figure 4.6 Class Design Pattern Sample 65
Figure 4.7 Rel Design Pattern Sample 65
Figure 4.8 hCard Example 67
Figure 4.9 vCard Format after Exporting 68
Figure 5.1 How Standards are Formed [7] 70
Figure 5.2 Scorm Metadata – General Category XML Binding Example 81
Figure 6.1 Use Case I 86
Figure 6.2 Use Case II 87
Figure 6.3 Use Case III 87
Figure 6.4 Sample XML Binding of Application Profile 94
Figure 6.5 Sample Microformat of Proposed Application Profile (LO2) 95
Figure 6.6 Demonstration of Proposed Microformat 96
Figure 6.7 Sample CSS File 97
Figure 6.8 Demonstration of Proposed Microformat, CSS Applied 97
Figure 6.9 Sample RDF Document of LO2 99
Figure 6.10 Graph Representation of LO2 100
Figure 6.11 View from XSLT File for Microformat to RDF Conversion 100
Figure 6.12 Web Service Detected by Liquid Studio 101
Figure 6.13 Web Service Called via Browser 102
Figure 6.14 End-user Interface Displaying Results of a Query 103
Figure 6.15 End-user Interface Displaying Options Panel 104
Figure 8.1 Learning Object Metadata Lifecycle 112

List of Abbreviations

DC	Dublin Core
DOM	Document Object Model
DTD	Document Type Definition
GRDDL	Gleaning Resource Descriptions from Dialects of Languages
HTML	Hyper Text Mark-up Language
IEEE	Institute of Electrical and Electronic Engineers
IMS	Instructional Management Systems
LOM	Learning Object Metadata
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RSS	Really Simple Syndication
SCORM	Sharable Object Content Reference Model
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SPM	Smallest Permitted Maximum
SQI	Simple Query Interface
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WWW	World Wide Web
XHTML	Extensible Hyper Text Mark-up Language
XML	Extensible Mark-up Language
XML- RPC	XML Remote Procedure Call
XSD	XML Schema Definition
XSL	Extensible Style Sheet Language
XSLT	Extensible Style Sheet Language Transformations

Chapter 1
Introduction

E-learning refers learning which uses variety of technologies such as internet, television etc. to produce, deliver, manage, store, manipulate, and evaluate learning resources, activities and outcomes for better and efficient outputs. Another description of E-learning clarifies what it is meant by better and efficient output; "…e-enhancements of models of learning. That is to say that; using technology to achieve better learning outcomes, or a more effective assessment of these outcomes, or a more cost-efficient way of bringing learning environment to the learners." [1]. Online and hybrid nature of e-learning brings an important property together such as flexibility which makes learning progress independent of time and date, and enables features such as self-monitoring, self-determination of learning progress, always accessible information, and performance monitoring.

E-learning evolved a lot by the emergence of computers and later internet, and continues its raise with the advancements in network and mobile services and software market which offers variety of advanced learning environments, tools and adaptive multimedia technologies. As an example from one specific vertical market, that of education, a recent Datamonitor report suggested that the global learning market for higher education is set to grow with a healthy CAGR of 12%, to $1,891 million by 2008 [2]. By the support of technological advancements e-learning also faced with some important pedagogical movements such as learner centric, self directed approaches which are based on constructivist theories. These approaches consider learners as active participants of learning instead of passive consumers and change the role of teachers as facilitators who assist learners to clarify their goals and enable them to be capable of planning, executing and evaluating their learning

progress and outcomes collaboratively, without taking a particular position in the discussions, rather than being pure source of information [3, 4, 5]. The following statement describes the challenges of these approaches clearly; “Providing active, stimulating, authentic learning experiences that support learner collaboration, construction and reflection is major challenge for success of E-learning.” [6]. These approaches triggered the creation of learner-centric, social and collaborative learning environments. Moreover social software (blogs, wikis etc.) gained an important place for e-learning thus the mine of data, World Wide Web, because of Web 2.0’s great collaborative potential, Wisdom of Crowds, and simple find-remix and share rule. In today’s world we are much more connected and that fosters engagement of information and makes it necessary thus today embedding social networking and collaboration into learning progress is considered as driving force for learner’s motivation and activity. Social Web 2.0 tools coming into play also formed different type of e-learning forms such as blended learning which is usually combination of several web based tools and instructor-led learning. All in all, learners are not bound neither to individual learning environments as closed box of pure information nor to classical in-class learning environments anymore, instead by the guidance of these constructivist learning theories they are facing with many tools including their particular learning environments which enables them to collaborate, to reach endless amount of information of web, and to remix-share it, thus also to create social networks. Depending on the situation, these tools are being used individually by learners, or by means of mashups, or as heterogeneous systems which involve several tools and might be centered by a particular learning system. All these movements and approaches ended up in same technological challenge as many of other software related disciplines did; interoperability. E-Learning market has already been over populated with tools and platforms to support different types of learning communities with learning management, content management and communication tools [5]. Thus enabling all these tools to communicate and share data among each other became a key technological challenge to reach the objectives of driving pedagogic theories behind.

Interoperability ensures that applications can share data and communicate with each other in order to complete a particular task without any concern of each others internal structure or nature. This is completely desired but something is missing,

what about meanings, and relations between data? Isn't it also desirable that an application can understand collections of data for particular course and able to link and present them coherently? Here other global concepts appear which also reflect greatly over e-learning technologies, one is semantic which aims giving meaning to data, and ontology which means defining relations and rules between data pieces. Applications can exchange data with an agreement of the structure of data; however this seems impractical when the huge amount of information that is produced for human resides out there in World Wide Web.

Here, Standards come into scene as a solution for the goals specified above; interoperability and semantics. Standards help to ensure interoperability and five other important "abilities" which protect and even nurture e-Learning investments; re-usability, manageability, accessibility, durability, scalability and affordability [7]. Several bodies have developed standards, specifications, guidelines in order to enable interoperability among different tools, systems and applications and to achieve semantic web vision like IEEE Learning Standards (e.g. LOM), IMS Global Learning Consortium (content packaging, metadata etc.) which are education specific standards, and like W3C,The World Wide Web Consortium, (HTML, XHTML, XML, RSS, RDF, SOAP etc), Dublin Core, Microformats etc. which are globally for Web.

The work going to be done in this thesis bases on the key element called learning objects (also called as "learning resource"), for our case which might be anything digital (image, text etc) that can be used for educational purposes. A more down-to-earth approach is to think of a learning object as a digital part of a course ranging in size and complexity from a single graphic to an entire course itself [8]. Enabling learning objects to be re-usable and sharable by different tools, systems and applications is major for e-Learning interoperability challenge, furthermore enabling computers to understand and process information on the web will also move us to beginning of the semantic web vision. Here metadata plays a key role; metadata can be simply explained as data about data, for instance title information of an image or author information of a text is considered as a metadata element. The bodies noted previously also have standards and specifications published for metadata information. In the content of this thesis LOM (Learning Object Metadata)

standardized by IEEE holds an important place; however other bodies and standards will also be investigated. Most of these standards consist of many elements which might contain some irrelevant attributes depending on the application, for instance LOM currently has more than 71 elements. After investigation of LOM and other metadata standards, next step is going to be deriving an application profile from LOM, which is simply a sub-set of LOM elements. The application profile is going to be used as a basis for embedding learning resources into web pages in order to enable them to be harvested. Rest of the work can be tracked over Figure 1.1; however before going further it is better to explain some key technologies.

The terms XHTML, XSL, XSLT, SPARQL, RDF, GRDDL, SOAP, and WSDL are the keystones which constitute the core of enabling technologies of interoperability and semantic web together with other technologies and standards those are not listed here or not implemented yet but expressed as future possibilities. It is better to have brief descriptions of these technologies and standards here. All these technologies and standards going to be explained in details in coming chapters, however it is important to denote here that all of these technologies base, involve or relate with XML; Extensible Markup Language.

- **XML:** It is a markup language like HTML but differs from HTML because it's purpose and functionality; it is used to carry data not to display and employs own tags of user instead of predefined tags. (W3C)
- **XHTML:** is HTML defined as an XML application, and a strict version of HTML which is aimed to replace HTML. (W3C)
- **XSL:** Stands for Extensible Style sheet Language, describes how the XML document should be displayed, consist of three parts; XSLT, XPath and XSL-FO. (W3C)
- **XSLT:** Extensible Style sheet Language (XSL) Transformation is a language for transforming XML documents into another XML document or other formats such as XHTML or RDF. (W3C)
- **RDF:** The Resource Description Framework is a standard for describing resources on the Web. (W3C)

- **GRDDL:** is a mechanism for Gleaning Resource Descriptions from Dialects of Languages. GRDDL introduces markup based on existing standards for declaring that an XML document includes data compatible with the RDF and for linking to algorithms (typically represented in XSLT), for extracting this data from the document.
- **SPARQL:** is a standardized query language for RDF data which offers developers a way to write queries across the wide range of RDF information on the web. (W3C) [9,10]
- **SOAP:** is a simple XML-based protocol to let applications exchange information over HTTP. (W3C)
- **WSDL:** Web Service Definition Language is a XML document which describes Web Services, such as operations are available. (W3C)

After defining an application profile, next step will be proposing a light-weight Microformat which is a set of XHTML tags that is used to embed information into web pages which are understandable both by machines and humans while considering the human as first priority. Embedding Microformat structures into XHTML page might be simply manual which means via basic XHTML coding or it can be done via applying XSLT transformations over XML bindings of learning objects. After proposing a light-weight Microformat for learning objects, a web service (based on SOAP and WSDL) will be created which collects learning objects in XHTML pages and transforms them to RDF format via XSLT or GRDDL, which is also based on XSLT. XHTML file might be accompanied with an XSL file which tells how to translate embedded information into RDF, if so then GRDDL will be used, if this is not the case then a predefined XSL transformation will be applied for RDF transformation. This service will also enable these objects to be queried via an SQI target, which is an interoperability structure that enables heterogeneous systems to communicate for the purpose of learning object retrieval by using a common query langue in our case it is SPARQL [4]. The last step of the work will be setting up a search client which will use SQI target to query learning resources, this small application will serve as the proof of concept.

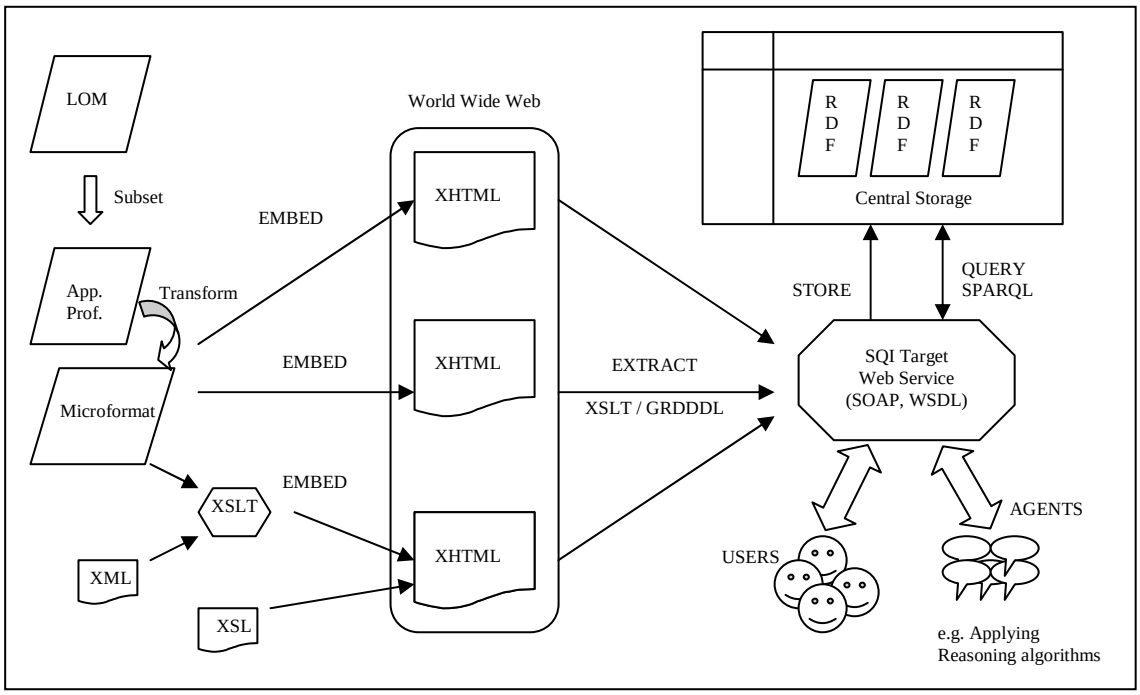


Figure 1.1 Graphical Representation of the Proposed Work

To sum up, the aim of this thesis is to enable harvesting of Learning Objects which are embedded in web pages by proposing a model. Therefore, several Learning Object Metadata standards will be investigated for compatibility with this harvesting approach, and then a light weight Microformat for learning objects will be proposed. Then a web service will be created which uses XSLT/GRDDL to extract learning objects in different web pages, and uses SQI target for retrieval facility with a more complex query language called SPARQL. Final work will be providing a search client employing created SQI service for search and retrieval of learning objects.

Rest of this document is structured as follows; *Chapter 2* involves the state of the art on basic related web technologies while *Chapter 3* provides the state of the art on related semantic web technologies. The details of Microformat approach will be included in *Chapter 4* and *Chapter 5* introduces and investigates learning object metadata standards and specifications. *Chapter 6* proposes an application profile and Microformat for learning object search and retrieval, and then provides an example implementation. *Chapter 7* evaluates the model and the application while *Chapter 8* discusses about possible future developments related with the work done and also involves a brief critique of the driven concepts, technologies and philosophies behind.

Chapter 2
Ground Web Technologies and Specifications

In this chapter some of W3C technologies and specifications that were referred in *Chapter 1* will be explained in details, also details of SQI will take part in this chapter; however these details will be limited by the needs of this thesis, wherever proper other relevant or more advanced technologies and specifications will be referenced and introduced briefly. These interoperable technologies which will be under interest of this chapter are XML, XHTML, XSL, XSLT, SOAP and WSDL and they will form the technological ground of the main work targeted by this thesis in the sense of interoperability, other technologies RDF, GRDDL, and SPARQL will be under consideration of *Chapter 3*, because these technologies differ from the each other in the sense of purpose and acceptance. First set of technologies has a wild acceptance and became wildly accepted standards, however second set of technologies are still have a lack of acceptance and they target semantic issues even though they are built over the first set of technologies.

There is several open-source, or freeware tools and services that can be used to test and create the examples given in this chapter. The tools and services which are used to test and validate examples and concepts in this chapter are as follows;

Liquid XML studio 2008, this is a freeware tool that can be used to create and test XML, XSLT, XSD, WSDL etc. documents and Web Services. Most of the examples in this chapter tested and validated with this tool.

- **Mark-up validator:** This service is used to validate Web Documents in the format of HTML, XHTML etc. It can be found in the following link: http://validator.w3.org

- **XML Schema validator:** This validator is used to test and validate XML Schema documents. It can be found in the following link http://www.w3.org/2001/03/webdata/xsv.
- **Online XSLT 2.0 Service:** This service is used to test and validate any given XSLT document over any given XML document. It can be found in the following link; http://www.w3.org/2005/08/online_xslt/

2.1 XML

Extensible Mark-up Language (XML) is a simple, very flexible text format. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere [11]. XML is recommendation of W3C, it is basically a mark up language like HTML.

It is noted previously that XML differs from HTML because of its purpose and functionality. XML allows user to use any tag she wants instead of using predefined tags. Therefore it is self descriptive and meanings of tags can only be understood by the writer if the tags are not clear enough. A XML document will not mean anything to a browser; it is not used for presentation purposes. XML does not do anything; XML was created to structure, store, and transport information [12]. XML files are plain text files therefore any program which can understand the tags in particular XML file can do special operations over this XML document.

XML is software and hardware independent by nature, so it really makes an efficient way of storing and transporting data among any kind of programs regardless of compatibility level among them. XML has been widely adopted both by Web Applications and by desktop applications. A XML file simply can be used as a response format for a Web service, or one can simply keep all product data of her web site in XML files to separate data from presentation, or a desktop application can hold all its configuration data in XML files. XML is also used for other internet technologies such as XHTML, RSS, WSDL, RDF, OWL etc. Therefore without clear understanding and competence of XML, it will not mean much in the kingdom of XML.

2.1.1 How to Create XML Documents

The XML file, Student.xml, in Figure 2.1 demonstrates simple XML file which holds basic data for a student.

```
1    <?xml version="1.0" encoding="utf-8"?>
2     <student recordid="1">
3            <name>Ahmet</name>
4            <surname>Soylu</surname>
5            <age>23</age>
6            <gender>Male</gender>
7     </student>
```

Figure 2.1 Simple XML Code

XML file consists of tags which are simply in “<tag>data</tag>” format. Each structure that has a starting tag and a closing tag called as element. Let’s move through the XML file in Figure 2.1, first line tells that this document is a XML document then it gives the version of XML and encoding of this XML file. XML files have a tree structure; therefore each XML file has a root element necessarily, in Figure 2.1 at line 2, the “<student>” tag is starting tag of our root element. Each element might have children, in this case “<name>”, “<surname>”, “<age>”, “<gender>” are children of “<student>” element, also each element might have siblings; all children of “<student>” are siblings of each other. Elements are case sensitive therefore “<name>” and “<Name>” are not equal to each other. Elements need to be nested properly, for instance “<a>” is nested wrongly, its correct nesting would be “<a>”. Elements might have attributes like the case of the tag at second line; “recordid” is the attribute of “<student>” element. All children of “<student>” element can be represented as attributes of this element. This is possible however; it would be inefficient and inflexible, only having data which is irrelevant to original data should be placed as attributes, which is actually metadata. All values of attributes should be quoted with a single quote or double quote. Between start and end tags of any element there might be other elements those elements having other elements as data are called complex elements, or there might be plain text as data those elements which hold plain text as data are called simple elements or leaves. Here <student> is a complex element, but “<name>” is a simple element. “<name>” contains text data. Using special characters inside data should be avoided, those are; ‘<’, ‘>’, ‘&’ etc. instead their HTML representations should be

used, because content of simple elements are also being parsed by parsers. Each simple element might have two types of data; first one is PCDATA which is parsed by parsers, or CDATA which is not parsed by parsers. Therefore these special characters can be used inside CDATA. All data inside simple tags are PCDATA by default, usage of CDATA is in following format <name> “<![CDATA["Ahmet"]]> </name>”. It is also important to note naming rules here; element names might include letters, numbers and other characters but can not include spaces, and can not start with numbers, punctuation characters and with “xml” keyword.

When one opens the XML document in Figure 2.1 via a browser she will just see a coloured version of this sample code, and she will see a “-” sign before <student> tag which allows her to expand or hide content of complex elements. This is so because, except the structure of a XML document there is nothing it can tell to the browser.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <basket xmlns:person="www.xxx.com/person" xmlns:product="www.xxx.com/product" >
3       <person:owner>
4           <person:name>Ahmet</person:name>
5           <person:surname>Soylu</person:surname>
6           <person:age>23</person:age>
7           <person:gender>Male</person:gender>
8       </person:owner>
9       <product:item id="134">
10          <product:name>TV</product:name>
11          <product:price>400</product:price>
12          <product:size>55</product:size>
13          <product:type>Flat</product:type>
14          <product:comment><![CDATA[This is nice. ]]></product:comment>
15      </product:item>
16      <date>2008-03-11</date>
17      <status>Draft</status>
18  </basket>
```

Figure 2.2 Name Spaces

Let’s move with another example, the XML code in Figure 2.2 includes data about a simple shopping basket. It introduces another concept, called “Name Spaces”. Line two introduces two name spaces which are “product” and “person”. “XMLNS” refers to XML Name Spaces, name spaces used to differentiate tags that belong to different XML documents. Actually each name space provide a set vocabulary in some respect, for instance to define a product, a product name space might provide “price”, “name”, “brand” etc. as terms of this vocabulary. Besides data on different XML documents can be mixed into single XML document, in order to prevent tag name

conflicts name spaces are used. In code in Figure 2.2 there are two tags called “<name>”, one is name for basket owner and other one is name for product. These two tags are differentiated by using different name spaces for each. Name space declaration is added to beginning of each tag via semi-colon. The URI that name space shows actually never used, generally it targets to a page where name space elements are defined. Only “date” and status “tags” belong to current XML file other tags belong either to person or product name space in this example.

So far it is really clear that anyone can store her data via well-formed XML, but what about restrictions and constrains, for instance what happens when one writes “Male” between “<age>” tags, or writes “<gender>” element two times. So parser needs to verify consistency of the data and structure of a XML file. Here DTD, Document Type Definition, and XSD, XML Schema Definition, plays important role. Those technologies used to define structure of a XML file and applying constraints on data. XSD is more flexible than DTD therefore DTD will not be under interest of this chapter.

2.1.2 XML Schema Definition

The purpose of a XML Schema is to define the legal building blocks of a XML document, just like a DTD. [13] It is used to define and validate structure, elements, attributes, data types of a XML document. It is written via XML syntax; it is much more powerful than DTD because it is extendable and supports name spaces and data types.

The XML code in Figure 2.3 is the same code in Figure 2.1, only difference is that it references a XML Schema Definition file which is Student.xsd. First the XML Schema name space instance is referenced, and then schema definition is referenced via “noNameSchemaLocation”. In examples of this chapter, all codes assumed to reside in same physical place however this location is usually a web address denoted by a URL.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="Student.xsd" recordid="1">
3      <name>Ahmet</name>
4      <surname>Soylu</surname>
5      <age>23</age>
6      <gender>Male</gender>
7  </student>
```

Figure 2.3 XML with Schema Reference

The code in Figure 2.4 is XML Schema definition of Student.xml. First line of the code is a usual XML document declaration. The second line first tells that any element and attribute used in this XML document must be name space qualified. Then it references XML Schema Name Space with “xs” which means any element that starts with “xs:” belongs to XML Schema Name Space. The rest of the document says that “<student>” element is root of this XML file and it is a complex element (“xs:complexType”), it consist of sequence (“xs:sequence”) of simple elements which are name, surname, age, and gender elements, and it has a required attribute “recordid” which must be positive integer. “Type” attribute in element declarations refers to data type of element, “maxOccur” means how many times this element can appear at most and “minOccur” attribute means how many times this element should appear at least. Variety of rules and constraints can be applied over a XML document, and much more complex situations exist; a XML document might include different name spaces like in the example Figure 2.2 and so on.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <xs:schema attributeFormDefault="qualified" elementFormDefault="qualified"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
3  <xs:element name="student">
4      <xs:complexType>
5          <xs:sequence>
6              <xs:element minOccurs="1" maxOccurs="1" name="name" type="xs:string" />
7              <xs:element minOccurs="1" maxOccurs="1" name="surname" type="xs:string" />
8          <xs:element minOccurs="1" maxOccurs="1" name="age" type="xs:positiveInteger" />
9              <xs:element minOccurs="1" maxOccurs="1" name="gender" type="xs:string" />
10         </xs:sequence>
11         <xs:attribute name="recordid" type="xs:positiveInteger" use="required" />
12     </xs:complexType>
13 </xs:element></xs:schema>
```

Figure 2.4 Schema Definition

A better example is demonstrated in Figure 2.5 to give deeper understanding and competence of XSD. Figure 2.5 contains a more complex example for a shopping

basket which includes basket owner details like name, surname, age and gender, and basic data about products like product name and price. There are three XSD files assigned for this XML file, one for “basket.xml” in general (“basket.xsd”), one for Person name space (“person.xsd”) and other one is for product name (“product.xsd”) space.

```
1   <?xml version="1.0" encoding="utf-8"?>
    <basket xmlns:person="person" xmlns:product="product"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="basket.xsd">
3       <person:owner>
4           <person:name>Ahmet</person:name>
5           <person:surname>Soylu</person:surname>
6           <person:age>23</person:age>
7           <person:gender>Male</person:gender>
8       </person:owner>
9       <product:item id="A13">
10          <product:name>TV</product:name>
11          <product:price>400</product:price>
12          <product:comment><![CDATA[This is nice. ]]></product:comment>
13      </product:item>
14      <product:item id="B42">
15          <product:name>Mouse</product:name>
16          <product:price>10</product:price>
17      </product:item>
18      <date>2008-03-11</date>
19      <status>Draft</status>
20  </basket>
```

Figure 2.5 XML in Details

In Figure 2.6 “Basket.xsd” file content is shown, first line is regular XML declaration that is used before. In second line name spaces that will be used in this document are declared which are person and product name spaces. In third and fourth lines the xsd files imports other xsd files (person and product). In import statement we do have name space attribute which tells for which name space the schema file is used for.

```
1   <?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:product="product" xmlns:person="person"
2   attributeFormDefault="qualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:import schemaLocation="person.xsd" namespace="person" />
4   <xs:import schemaLocation="product.xsd" namespace="product" />
5       <xs:element name="basket">
6           <xs:complexType>
7               <xs:sequence>
8                   <xs:element minOccurs="1" maxOccurs="1" ref="person:owner" />
9                   <xs:element minOccurs="0" maxOccurs="unbounded" ref="product:item" />
10                  <xs:element minOccurs="1" maxOccurs="1" name="date" type="xs:date" />
11                  <xs:element minOccurs="1" maxOccurs="1" default="Draft" name="status"
    type="xs:string" />
12              </xs:sequence>
13          </xs:complexType>
14      </xs:element>
15  </xs:schema>
```

Figure 2.6 Schema for Figure 2.1

Then it says there is a root element called “basket” which is complex type, and it has sequence, “<xs:sequence>” , of elements in given order. “<xs:all>” or “<xs:choice>” could also be used instead of “<xs:sequence>”. “<xs:all>” means all elements will be used without any order constraint, “<xs:choice>” means one of this elements can be used depending on the choice. First element is defined as owner element in person name space, and second one defines the product element which is in product name space defined in product.xsd. “minOccurs” defined as 0 and “maxOccur” defined as “unbounded” for “product” element which says product element might appear at least 0 times or at most infinite times. For status element there is a default value defined as “draft”. There is variety of types that can be used for “type” such as “integer”, “positiveInteger”, “string”, “date” etc.

Figure 2.7 shows the content of Person.xsd which defines schema for person name space. The lines from 3-14 introduces two simple element types that can be used by means of reference. Currently they are just definition (like custom data types). First simple element type is called “ageType” which will be used to denote age data, its restrictions (“xs:restriction”) says it needs to be positive integer and minimum value for this positive integer is 18 (“minInclusive”) and maximum value (“maxInclusive”) for this positive integer is 99. Other simple type definition is called as “genderType” which will be used to denote gender data. Its restrictions says it only can take “Male” and “Female” as value (“xs:enumeration”). It is important to note here, a complex

element can also be defined as element type. After element type definitions, XSD document starts to define XML file, it says root element is “owner” and this complex type element includes “name”, “surname”, “age”, and “gender” child elements. “Name” element is string type which has minimum length (“xs:minLenght”) of 2 and maximum length (“xs:maxLenght”) of 5.

```
1   <?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:person="person" attributeFormDefault="qualified"
2   elementFormDefault="qualified" targetNamespace="person"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:simpleType name="ageType">
4       <xs:restriction base="xs:positiveInteger">
5           <xs:minInclusive value="18" />
6           <xs:maxInclusive value="99" />
7       </xs:restriction>
8   </xs:simpleType>
9   <xs:simpleType name="genderType">
10      <xs:restriction base="xs:string">
11          <xs:enumeration value="Male" />
12          <xs:enumeration value="Female" />
13      </xs:restriction>
14  </xs:simpleType>
15   <xs:element name="owner">
16      <xs:complexType>
17          <xs:sequence>
18              <xs:element minOccurs="1" maxOccurs="1" name="name">
19                  <xs:simpleType>
20                      <xs:restriction base="xs:string">
21                          <xs:minLength value="2" />
22                          <xs:maxLength value="10" />
23                      </xs:restriction>
24                  </xs:simpleType>
25              </xs:element>
26              <xs:element minOccurs="1" maxOccurs="1" name="surname" type="xs:string" />
27              <xs:element minOccurs="1" maxOccurs="1" name="age"
    type="person:ageType" />
28              <xs:element minOccurs="1" maxOccurs="1" name="gender"
    type="person:genderType" />
29          </xs:sequence>
30      </xs:complexType>
31  </xs:element>
32  </xs:schema>
```

Figure 2.7 Schema for Person Name Space

Figure 2.8 shows the content of Product.xsd which defines schema for product name space. It says root element is “item”. This element is complex type and it has sequence of elements which are name, price, and comment. The restriction for “comment” element says; it should collapse the content which means it will replace, file end signs, new line signs, and multiple lines with single space. In line 15

“xs:any” element means this complex element might be extended with new elements, another XSD file can be shown as reference in XML file for these new extended elements. It is also seen that “item” element has “id” attribute and a regular expression is applied as restriction to this attribute which enforces value of “id” to be consist of a capital character from A to Z and two digits from 1 to 9.

```
1   <?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:product="product" attributeFormDefault="unqualified"
2   elementFormDefault="qualified" targetNamespace="product"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="item">
4       <xs:complexType>
5           <xs:sequence>
6               <xs:element minOccurs="1" maxOccurs="1" name="name" type="xs:string" />
7               <xs:element minOccurs="1" maxOccurs="1" name="price"
    type="xs:positiveInteger" />
8               <xs:element minOccurs="0" maxOccurs="1" name="comment">
9                   <xs:simpleType>
10                      <xs:restriction base="xs:string">
11                          <xs:whiteSpace value="collapse" />
12                      </xs:restriction>
13                  </xs:simpleType>
14              </xs:element>
15              <xs:any minOccurs="0" maxOccurs="2" />
16          </xs:sequence>
17          <xs:attribute name="id">
18              <xs:simpleType>
19                  <xs:restriction base="xs:string">
20                      <xs:pattern value="[A-Z][0-9][0-9]" />
21                  </xs:restriction>
22              </xs:simpleType>
23          </xs:attribute>
24      </xs:complexType>
25  </xs:element>
```

Figure 2.8 Schema for Product Name Space

2.2 XHTML

XHTML 1.0, Extendable Hypertext Mark-up Language, is a reformulation of HTML 4 as a XML 1.0 application [14]. XHTML is aimed to replace HTML; it is actually stricter version of HTML and recommendation of W3C. Currently, Web browsers can render bad formed HTML (like having incorrectly nested tags) documents, and tries to fix and interpret document. This gives so many loads to browsers. Think about small PDAs or mobile devices, even the load will be much higher for this kind of devices. Therefore - by combining HTML and XML, and their

strengths, we got a mark-up language that is useful now and in the future – XHTML [15]. Today's browsers both support XHTML and HTML, however in the future they are just supposed to support XHTML, that will cause web to be consist of well-formed web pages.

There are important restrictions that XHTML enforces different than HTML, and actually they are the main difference between HTML and XHTML coming from XML. These restrictions are as follows, all elements must be properly nested, all elements need to have a closing tag or if it is en empty element like "<hr>" it should also must be used as "<hr />". All element and attribute names must be lower case, and each XHTML file need to have a root element which is "<html>". Attribute values need to be double or single quoted. "id" attribute should be used instead of "name" attribute. Also attribute minimization is not allowed, for instance this "<input checked>" should be written as "<input checked="checked">". All HTML documents have to have "DOCTYPE" declaration. "html", "head", "title" and "body" elements are mandatory. Inside the "DOCTYPE" a DTD for XHTML file is declared, this DTD is used to ensure syntax and grammar of the XHTML document. There are three types of DTD that can be used with an XHTML document; those are "strict", "transitional", and "frameset". Strict one does not allow any presentational features, a CSS (Cascading Style Sheet), need to be used together for presentational concerns. Transitional ones allows presentational features to be used and frameset one allows HTML frames to be used. The sample code in Figure 2.9 demonstrates a simple XHTML document.

```
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
1   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2   <html>
3       <head>
4           <title>Sample XHTML Document</title>
5       </head>
6       <body>
7           <hr />
8           <p id='p1'>This is a sample XHTML document</p>
9           <hr />
10      </body>
11  </html>
```

Figure 2.9 Sample XHTML Document

XHTML also defines standard attributes for tags and events (JavaScript events), Furthermore W3C created small set of XHTML elements as modules that can be used for different type of devices such as from a normal pc explorer to a mobile explorer, this is done because whole set might be so much for a device, and a small set might be insufficient for a pc browser. There are several modules created. Attribute and event sets and XHTML modules will not be detailed here, however W3C site might be checked for more details on these issues.

2.3 XSL

XSL is a family of W3C recommendations for defining XML document transformation and presentation. It consists of three parts [16]:

- **XSLT:** XSL Transformations, a language for transforming XML.
- **XPath:** XML Path Language, an expression language used by XSLT to access or refer to parts of a XML document.
- **XSL-FO:** XSL Formatting Objects, a XML vocabulary for specifying formatting semantics

XSLT and XML will be under focus of this chapter, those technologies will be useful to transform a XML document to another XML document or to a RDF document or to a XHTML page.

2.3.1 XPath

XPath is a language for finding information in a XML document. XPath is used to navigate through elements and attributes in a XML document [17].

XPath uses expression similar to traditional computer system to navigate in XML files. It has a library for standard functions which involves more than 100 built-in functions. XPath is a major element for XSLT therefore it is important to have a XPath knowledge to use XSLT.

XPath considers a XML file as a collection of nodes, and there seven types of XML items those are accepted as nodes; elements, attributes, text, name space, processing –instruction, comment, and root nodes [17]. Nodes might have, parent, children,

siblings, ancestor, and descendant relationship among each other. Parent, children, and sibling relationship already introduced at XML section, ancestors of an element are parent of this element, parent of its parent and so on. Descendants of an element are children of this element, children of its children and so on. Here on Table 2.1, important XPath expressions are introduced.

Table 2.1 Example Expressions for XML File in Figure 2.5

Expression	**Description**
/	Selects from the root node.
basket	Selects all the child nodes of basket.
/basket	Selects the root node of basket.
/basket/person:owner	Selects owner elements that are children of basket. If there were no name space in out XML document query would be /basket/owner.
//product:price	Selects all the price elements , no matter where they are in the document.
basket//product:price	Selects all price elements that are descendant of the basket element, no matter where they are under the bookstore element.
//@id	Selects all the attributes those are named id.
/basket/product:item[1]	Selects the first item element that is the child of the basket element.
/basket/product:item[last()]	Selects the last owner element that is the child of the basket element.
/basket/product:item[last()]	Selects the last but one book element that is the child of the bookstore element.
/basket/product:item[last()<3]	Selects the last two item elements that are children of the basket element.
//product:item[@id]	Selects all the item elements that have an attribute named id.
//product:item[@id=’A13’]	Selects all the item elements that have an attribute named id with a value of 'A13'
/basket/product:item[product:price>100]	Selects all the item elements of the basket element that have a price element with a value greater than 100.
/basket/product:item[product:price>100]/product:name	Selects all the name elements of the item elements of the basket element that have a price element with a value greater than 100
/basket/*	Selects all the child nodes of the basket element.
//*	Selects all elements in the document
//product:item[@*]	Selects all item elements which have any attribute.
//product:name | //product:price	Selects all the name AND price elements of all product elements.
//product:name/text()	Selects all the text values of name elements, result will be, TV and Mouse.

These expressions can be tested over any XML document via Liquid XML studio XPath facility, most of the web programming languages, and browsers (via JavaScript by using Document Object Model, DOM) allows XPath expressions to be used. The following expressions can also be tested via IE 7.0 DOM, however it is always advisable to use this expression on server side, as a programmer should never completely trust on client side processing.

There is also a concept called axes. An axis defines a node-set relative to the current node [17]. In Table 2.2, there are the examples of most important ones. Rest can be found on W3C site.

Table 2.2 Example Axes for XML Code in Figure 2.5 [17]

Expression	**Description**
child::product:item	Selects all item nodes that are children of the current node
attribute::id	Selects the lang attribute of the current node
child::*	Selects all children of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects all text child nodes of the current node
child::node()	Selects all child nodes of the current node
descendant::product:item	Selects all item descendants of the current node
ancestor::product:item	Selects all item ancestors of the current node
ancestor-or-self::product:item	Selects all item ancestors of the current node - and the current as well if it is a book node
child::*/child::price	Selects all price grandchildren of the current node

2.3.2 XSLT

XSLT stands for Extensible Style Sheet Transformation and it is a W3C recommendation XSLT is designed for use as part of XSL, which is a style sheet language for XML. In addition to XSLT, XSL includes a XML vocabulary for specifying formatting. XSL specifies the styling of a XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary [18].

HTML uses predefined tags that browsers know how to render, and via CSS it is much easier to assign custom presentation features to any element of HTML.

However a XML file does not include any predefined tags therefore, they are not understandable for browsers or by other programs. Thus, XSLT might be used to tell browser how a XML file should be displayed or can be used to transform a XML file into another XML file or a different format such as RDF. XSLT uses XPath to traverse on a XML file when a particular match found then XSLT transforms this matching part into desired result document.

In Figure 2.10, XSLT usage is demonstrated via example XML and XSLT documents. In XML file the line 2 makes the only difference from a regular XML file. This line references the XSLT file and says that this is a style sheet file. XSLT file starts with a regular XML line, line 1, since it uses XML syntax. The second line refers to XSL name space to use XSLT features. The rest of the document includes transformation rules and XPath expressions. After having an XSLT file and a XML file that references this XSLT file as a style sheet, when one opens this XML file via browser XML file is presented in a way that related XSLT file describes.

#	**XML File**	**XSLT File**
1	<?xml version="1.0" encoding="ISO-8859-1"?>	<?xml version="1.0" encoding="ISO-8859-1"?>
2	<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>	<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3	<basket>	<xsl:template match="/">
4	* * *	* * *
5		</xsl:template>
6	</basket>	</xsl:stylesheet>

Figure 2.10 How to Use XSLT

In Figure 2.11, a more detailed example has been given which covers most of the functionality of XSLT. The example XSLT file transforms basket.xml into a XHTML page. “<xsl:template>” element at line two selects the XML elements that the template will be applied, for given example it is set to whole document because of “/”. “<xsl:value-of select=’’>” at lines 5,6,7,8 is used to select value of a XML element and add it to result, here “select” allows applying an Xpath expression. “<xsl:for-each select=’’>” at line 15 is used to select elements those are belong to specific node set. “chose-when-otherwise” at lines 17-31 structure is used to apply a conditional test against XML file.

```
1   <?xml version="1.0" encoding="ISO-8859-1"?>
    <xsl:stylesheet version="1.0"
2   xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:person="person"
    xmlns:product="product">
    <xsl:template match="/">
     <html>
3      <body>
         <h2>My Shopping Basket</h2>
         <table border="1">
4          <tr> <th>Name</th>  <th>Surname</th> <th>Age</th>  <th>Gender</th> </tr>
           <tr>
5            <td> <xsl:value-of select="/basket/person:owner/person:name"/>  </td>
6            <td> <xsl:value-of select="/basket/person:owner/person:surname"/> </td>
7            <td> <xsl:value-of select="/basket/person:owner/person:gender"/> </td>
             <td> <xsl:value-of select="/basket/person:owner/person:age"/> </td>
8          </tr>
9        </table>
10       <table>
11         <caption><br/><b><i>Products in your basket</i></b></caption>
12         <tr bgcolor="silver">
13           <th>Name</th>      <th>Price</th>     <th>Comment</th>
14         </tr>
15         <xsl:for-each select="/basket/product:item">
16         <xsl:sort select="/basket/product:item/product:price"/>
17           <xsl:choose>
18             <xsl:when test="product:price &gt; 100">
19               <tr bgcolor="red">
20                 <td> <xsl:value-of select="product:name"/> </td>
21                 <td> <xsl:value-of select="product:price"/>  </td>
22                 <td> <xsl:value-of select="product:comment"/> </td>
23               </tr>
24             </xsl:when>
25             <xsl:otherwise>
26               <tr>
27                 <td> <xsl:value-of select="product:name"/> </td>
28                 <td> <xsl:value-of select="product:price"/> </td>
29                 <td> <xsl:value-of select="product:comment"/> </td>
30               </tr>
31             </xsl:otherwise>
32           </xsl:choose>
33         </xsl:for-each>
34       </table>
35       <br/>Date  :  <xsl:value-of select="/basket/date"/>   <br />
36             Status:  <xsl:value-of select="/basket/status"/>
37     </body>
38   </html>
39  </xsl:template>
40  </xsl:stylesheet>
```

Figure 2.11 Style Sheet for Figure 2.1

“<xsl:sort select=’’>” at line 16 is simply used to sort output. Another conditional test structure which is not used in this example is if clause. The format is as follows; “<xsl:if test=’expression’> some output </xsl:if>”.

2.4 Web Services

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network [19]. It is self descriptive and self-containing application component which uses open protocols and XML.

Web services provide application re-usability and data exchange between different types of applications and ensure loosely-coupling. There are many kind of applications needed very often, instead of writing same application again and again, a web service will be a good opportunity to use same application as a part of different applications. Moreover each one of these applications might be created via using different technologies, and each one has its own way to deal with data, in that sense a web application provides a common way of interaction and exchange of data between these applications.

The components which are the part of a web service can be counted as follows;

- Service Transport,
- XML messaging,
- Service Description,
- Service Discovery.

Service Transport is responsible for transportation of messages between applications, HTTP, FTP, SMTP etc. provides service transport. XML messaging provides means of encapsulating messages in a common XML form, so both applications can understand messages; examples are SOAP, XML-RPC (Remote Procedure Call), REST etc. Service Description defines an interface for service so other applications can communicate with this service with respect to this description; the service description language is WSDL which is also an XML document. Finally service discovery, gathers all services to a common registry so that other applications can find these services, this is handled via Universal Discovery Description and Integration, UDDI.

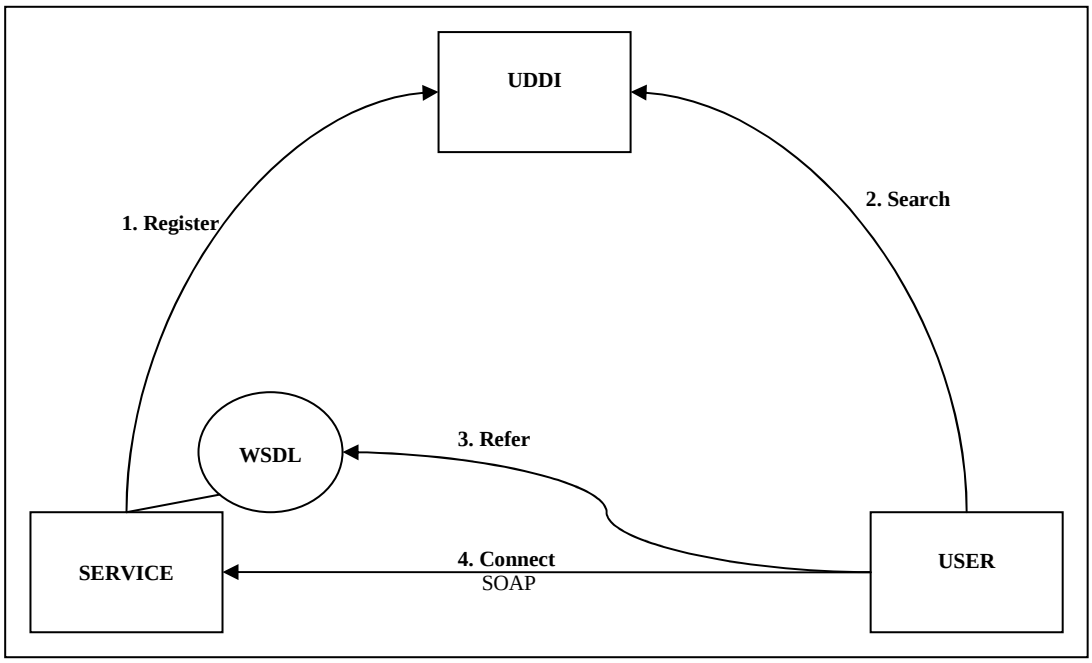


Figure 2.12 Web Service Architecture

The service transport technology that will be used is mainly based on the type of application; therefore the listed technologies for service transport are not complete alternatives to each other. There are no mature alternatives against WSDL and UDDI. However the XML messaging technology alternatives are competing, in that sense SOAP has a major place because of its advances. For instance XML-RPC is considered as too simple to be used in enterprise level where REST is not considered as well established yet. However SOAP is a W3C recommendation and it provides extensibility, better support and able to deal with complex data expressiveness. This brings complexity as a drawback.

2.4.1 WSDL

WSDL, Web Service Description Language, is a XML based language which describes Web services, their operations and how to access them. A WSDL document includes a set of definitions to describe service, besides within a single WDSL document several Web Services can be described.

The code piece in Figure 2.13 belongs to a sample WSDL document, main elements of a WSDL document are “<wsdl:portType>”, “<wsdl:message>”, “<wsdl:type>”, ”<wsdl:binding>”. “<wsdl:portType>” defines the operations performed by the Web

service, it can be thought as a function, “<wsdl:message>” defines messages used by the Web Service, they can be thought as parameters of functions. “<wsdl:binding>” defines the communication protocol used by the Web Service.

```
1   <?xml version="1.0" encoding="utf-8"?>
    <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                      xmlns:xs="http://www.w3.org/2001/XMLSchema"
2                     xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                      xmlns:tns="www.mysite.com/MyNamespace"
                      targetNamespace="www.mysite.com/MyNamespace">
3
4   <wsdl:message name="MyInput">
5       <wsdl:part name="parameters" element="xs:string" />
6   </wsdl:message>
7   <wsdl:message name="MyOutput">
8       <wsdl:part name="Body" element="xs:string" />
9   </wsdl:message>
10
11  <wsdl:portType name="ServiceSoap">
12      <wsdl:operation name="MyMWebMethod">
13          <wsdl:input message="tns:MyInput" />
14          <wsdl:output message="tns:MyOutput" />
15      </wsdl:operation>
16  </wsdl:portType>
17
18  <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
19      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
20      <wsdl:operation name="MyMWebMethod">
            <soap:operation
21  soapAction="http://www.example.com/SampleWebService/MyMWebMethod"
    style="document" />
22          <wsdl:input>
23              <soap:body use="literal" />
24          </wsdl:input>
25          <wsdl:output>
26              <soap:body use="literal" />
27          </wsdl:output>
28      </wsdl:operation>
29  </wsdl:binding>
30
31  <wsdl:service name="Service">
32      <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
            <soap:address location="http://www.example.com/SampleWebService/Service.asmx"
33  />
34      </wsdl:port>
35  </wsdl:service>
36
37  </wsdl:definitions>
```

Figure 2.13 Sample WSDL Document

The code piece in Figure 2.13 starts with common XML declaration after that it proceeds with the root element “<wsdl:definitions>” and makes declaration of

necessary Name Spaces. After root element messages are declared one as “MyInput” and other one as “MyOutput”, names are assigned arbitrarily. Code proceeds with “<portType>” declarations, port name “ServiceSoap” is an arbitrary name and can be thought as library name where “MyWebMethod” is can be thought as a function of this library. After operation declarations, document proceeds with bindings, and concludes with service definitions and location information. More details can be found in related W3C specifications.

2.4.2 SOAP

SOAP, Simple Object Access Protocol, is a platform and language independent and XML based protocol which enables applications to communicate and exchange information over HTTP. One of the most important pros of SOAP is its being based on HTTP, because HTTP is supported by all internet browsers and it provides a way for applications to communicate without any constraint based on operating systems, programming languages and technology.

A SOAP message is actually HTTP request/response based on XML, however it has to satisfy SOAP encoding rules. The SOAP message in Figure 2.14 is a simple SOAP request message encoded according to the sample WSDL document in Figure 2.13. Basic elements of a SOAP message are “envelope” (root element, mandatory), “header” (optional), “body” (mandatory) and “fault” (optional) element. The sample in Figure 2.14 starts with common XML declaration, then it proceeds with root element which is “<soap:envelope>” and it declares necessary Name Spaces. “<soap:Header>” comes after root element if it exists, and it contains application specific information like authentication. Document proceeds with “<soapBody>” where actually SOAP message takes place. A SOAP response follows the same convention. Usually SOAP users do not need to deal with SOAP messaging most of the application frameworks and tools already handles SOAP messaging and hides internal details from users.

```
1   <?xml version="1.0" encoding="utf-8"?>
    <soap:Envelope
2   xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
3
4   <soap:Header>
    <m:Trans xmlns:m=" http://www.example.com/SampleWebService/"
5   soap:actor=" http://www.example.com/SampleWebService" soap:mustUnderstand=1> 234
6   </m:Trans>
7   </soap:Header>
8
9   <soap:Body>
10  <m:MyWebMethod xmlns:m="http://www.mysite.com/MyNameSpace">
11     <m:myInput>Hello</myInput>
12  </m:MyWebMethod>
13  </soap:Body>
14
15  </soap:Envelope>
```

Figure 2.14 Sample SOAP Request Message

2.5 SQI

SQI is an Application Program Interface, API, which defines set of methods for querying different learning object repositories in order to achieve interoperability among learning repositories [20, 21]. The set of methods provided by SQI are actually web service methods however SQI does not impose any constraint on query language or type of result format because each repository has its own nature.

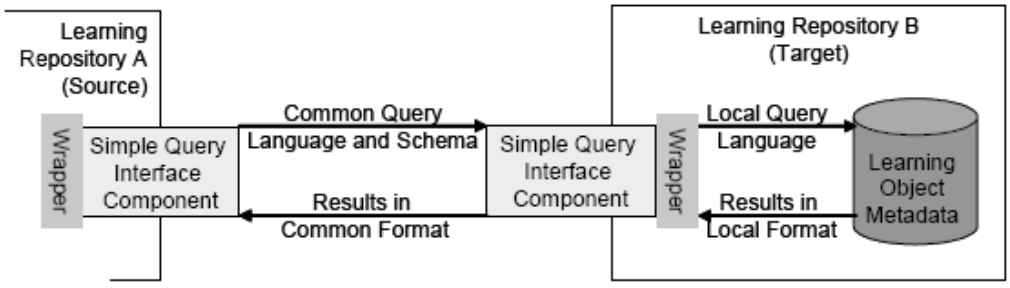


Figure 2.15 Communication Between Two Repositories [20]

The figure above, Figure 2.15, demonstrates the basic structure of communication for SQI. Learning repository A considered as source which submits a query to “target” which is learning repository B. It is the necessary for learning repositories to agree on a common query language and result format before submitting a query or for any further communication according to SQI specification. Wrappers might be needed

for any kind of mapping from common to local environment such as query or schema mapping.

SQI supports both synchronous and asynchronous queries; in synchronous querying target returns the result to the source as response to the query call and it is source initiated. In asynchronous querying, when target found enough amount of result for source's query, result is forwarded to source, and this is target initiated. Asynchronous querying is more difficult to handle, in SQI it is handled in following way when source calls query method it also sends a parameter which gives the address of source listener so target can forward result via this listener and another parameter which is an id the for the query submitted.

SQI also support stateless and stateful communication, in statelful communication target keeps track of previous actions with respect to their orders, and however in stateless communication target does not hold any information about previous actions. Moreover SQI enforces command-query separation principle which means that every method must either perform a command or process a query or send result back, a method performing both is not proper.

It is already noted before that it is assumed that before any further communication a session identifier assigned to the source. Therefore SQI methods could be separated as query management methods and session management methods. This says that SQI actually separates session management methods and query management methods. A session id must be assigned to source by target, this session does not necessarily need to be via password and user name, anonymous sessions can also be created and assigned, important use is identification of sources. Actually session management is not part of the SQI specification; however specification also refers some basic methods those might be needed. Below table, Table 2.3 gives the summary of SQI session management and query management methods.

Table 2.3 and Figure 2.16 give a brief overview of SQI API. More detailed specification of API involving input parameters, fault types etc. can be found at specification documentation.

Table 2.3 SQI Methods

#	Method	Implemented at
1	createSession	Target
2	createAnanoymousSession	Target
3	destroySession	Target
4	setResultsFormat	Target
5	setQueryLAnguage	Target
6	setMaxQueryResults	Target
7	setMaxDuration	Target
8	setResultSetSize	Target
9	syncronousQuery	Target
10	getTotalResultsCount	Target
11	asyncrounousQuery	Target
12	setSourceLocation	Target
13	queryResultListener	Source

“createSession” method creates and assigns a session identifier to source according to given defined credentials such as password. “createAnnonymousSession” method creates and assigns a session identifier without need of such credentials. “destroySession” method ends validity of a specified session identifier and destroys it. “setQueryLanguage” method defined the syntax of query language like SQL, SPARQL etc.

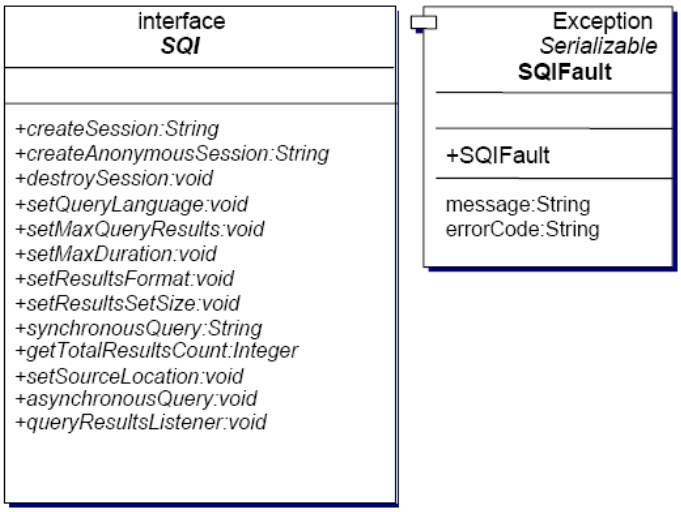


Figure 2.16 UML Class Diagram of SQI [20]

“setMaxQueryResults” method defines the maximum number of results that a query will produce.”setMAxDuration” method defines the time-out time for particularly asynchronous queries. “setResultsFormat” method defines the format of result that will be returned as a result of query. “setResultSetSize” defines the maximum number of results that will be returned by a single result set. ”synchronousQuery”

allows source to send a synchronous query to source. “getTotalResultCount” method returns the maximum number of results of query. “setSourceLocation” method defines the source’s result listener location so that target can forward results for asynchronous queries. “asynchronousQuery” method allows source to send an asynchronous query to target. “queryResutListener” method forwards the result set to source.

Chapter 3
Semantic Web Technologies and Specifications

This chapter introduces important Semantic Web technologies, RDF, SPARQL and GRDDL in details. RDF constitutes the basic building stone of Semantic Web Vision of W3C and inspires other works related with the Semantic Web, GRDDL and SPARQL enables harvest and use of RDF resources. These technologies are wildly considered as the future of Semantic Web where machines are capable of understanding and linking information on the web.

There are several tools and Web Services which are used to test and validate examples given in this chapter. These tools and services listed as follows;

- **Twinkle:** It is an open-source tool which provides a GUI interface in order to execute SPARQL queries. SPARQL queries in this chapter tested and validated via this tool. It can be found in the following link; http://www.ldodds.com/projects/twinkle/
- **W3C RDF validation Service:** This service is used to validate RDF documents and to transform them into triple and graph form. It can be found in the following link; http://www.w3.org/RDF/Validator/

3.1 RDF

RDF, Resource Description Framework, is a W3C recommendation; it provides a framework, syntax and schema, for describing web resources. RDF is based on XML, provides a XML syntax called RDF/XML, and inherits its tagging method and

name space facility. RDF supports the interoperability of metadata while it allows descriptions of Web resources - any object with a Uniform Resource Identifier (URI) as its address - to be made available in machine understandable form [22]. It is also one of the key stones of Semantic Web Vision. It is important emphasize that RDF is designed for machines not for human reading although there are some work (eRDF, RDFa, will be investigated later in *Chapter 4*) that also enables RDF to be human readable by embedding RDF into XHTML pages.

3.1.1 Model, Concepts and Syntax

Basically a resource described by RDF is collection of properties, and properties have values. An example would be as follows, “http://www.sales.com/tv.htm” is a resource, “price” is property and “300$” is property value. The combination of a “Resource”, a “Property”, and a “Property value” forms a “Statement” (known as the “subject”, “predicate” and “object” of a “Statement”) [23]. Subject must be presented by an URI, Unique Resource Identifier, which is usually URL, Universal Resource Locator, for web resources. URLs are particular kind of URI and used to locate objects that can be accessible via network. However objects those are not accessible via network or abstract concepts that actually do not exists physically can also be used as subject in RDF documents.

RDF file has been demonstrated at Figure 3.1. Main elements of an RDF file is “<rdf>” root element and “<rdf:description>” element. The first line of the code is usual XML declaration which says this is an XML document. At the second line RDF name space is referenced in order to use RDF syntax, also name space for product is referenced in order to use “product” name space elements. At line 3, it first identifies resource via using “<rdf:Description rdf:about=’resource URI’>”. At line 3 and 4, it defines the properties and their property values. Note that the product name space is out of RDF, RDF just provides the framework. Actually all properties of the resource could be written inside separate “description” elements, the example at Figure 3.1 is an abbreviation and also a common way of using. On the other hand all properties of resource could be used as attributes of “<rdf:description>” element. This might be used to embed RDF into a web page however this is not for presentation, in this way RDF document is embedded but not in a form that browser

can render and display it, only machines can detect and use it. However this limits the features of RDF.

```
1   <?xml version="1.0"?>
    <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:product="http://www.sales.com/product#">
3   <rdf:Description rdf:about="http://www.sales.com/tv">
4       <product:name>TV</product:name>
5       <product:price>300$</product:price>
6   </rdf:Description>
7   </rdf:RDF>
```

Figure 3.1 Sample RDF Document

RDF documents can be demonstrated via graphs; here the Figure 3.2 demonstrates the example given in a graph representation. The graph simply says that the TV resource has a price which is “300$” and a name which is “TV”. In RDF graphs nodes those are URI references shown as eclipse and the nodes those are literals shown as boxes.

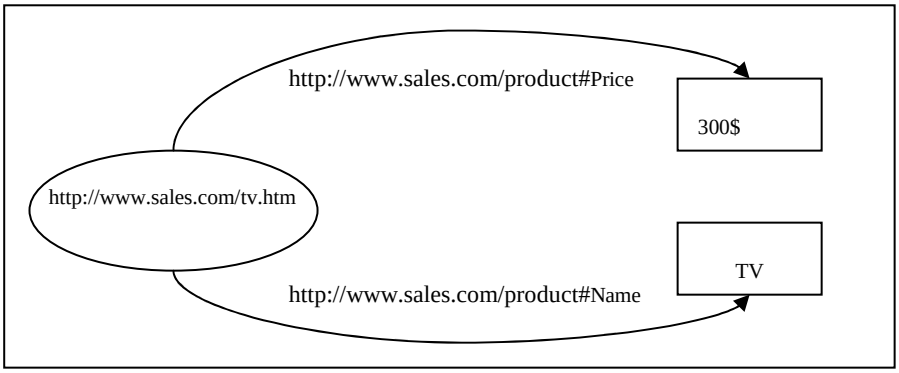


Figure 3.2 Graphical Representation of RDF

In plain English the graph in Figure 3.2 would be expressed as follows;

> http://www.sales.com/tv.htm has a price whose value is 300$.
> http://www.sales.com/tv.htm has a name whose value is TV.

Another way of representing a RDF document is using triples where each statement in the graph is written in a simple triple structure; subject, predicate, and object form in given order. The triple representation in Figure 3.3 belongs to the Figure 3.1. Note that “http://www.sales.com/product#” defines the name space, in other words it

provides a vocabulary. Any word after “#” character is an element of this vocabulary. It also important to note that as indicated in XML section before, this name spaces are not retrieved, this is just a convention. Also note that URIs must be written inside “<” and “>” characters as a convention.

```
1    <http://www.sales.com/tv.htm> <http://www.sales.com/product#Price> “300$”
2    <http://www.sales.com/tv.htm> <http://www.sales.com/product#Name> “TV”
```

Figure 3.3 Sample Triple Format

The way of representation in above example might cause long documents because of long URI names, therefore instead of repeating long name URIs each time, this URIs can be assigned to a “prefix” for one time, and then this prefix can be used instead of URIs. The example would be as follows;

```
1    PREFIX sales: http//www.sales.com
2    PREFIX product: http://www.sales.com/product#
3    sales:tv.htm          product:price          “300$”
4    sales:tv.htm          product:name           “TV”
```

Figure 3.4 Example of Prefix Use

Now it is time to traverse more on RDF syntax. RDF property elements can also be resources like the example given in Figure 3.5. In this example brand of the TV is another resource and “rdf:resource” attribute used to describe this case at line 7, and note that it is used within an empty tag, which means it has an “/” character at the end of tag.

```
1    <?xml version="1.0"?>
     <rdf:RDF
2    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:product="http://www.sales.com/product#">
3    <rdf:Description rdf:about="http://www.sales.com/tv">
4        <product:name>TV</product:name>
5        <product:price>300$</product:price>
6        <product:brand rdf:resource="http://www.xbrand.com/xbrand.htm" />
7    </rdf:Description>
8    </rdf:RDF>
```

Figure 3.5 Properties as Resources

The Figure 3.6 demonstrates the case when a property itself is also a resource. The demonstration only covers the brand property of Figure 3.5 since the rest of the

presentation is same as in Figure 3.2. As noted before resources are represented by an eclipse therefore eclipse is used instead of a rectangle for "http://www.xbrand.com/xbrand.htm" resource.

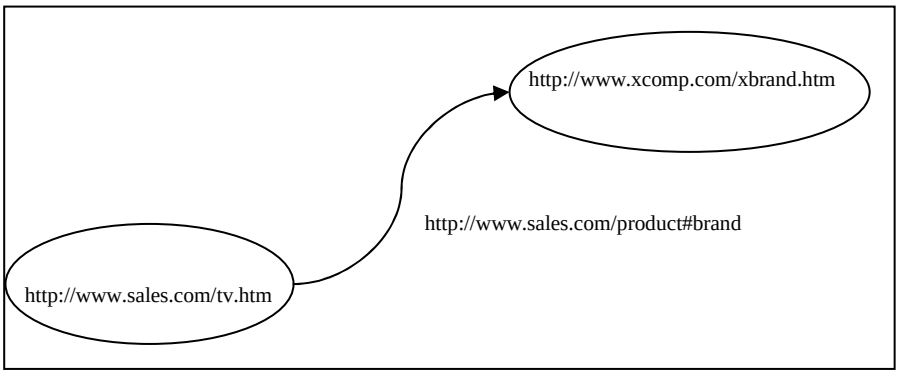


Figure 3.6 Resources as Properties

The triple demonstration of above graph is shown in Figure 3.7;

```
1   PREFIX sales: http://www.sales.com/
2   PREFIX product: http://www.sales.com/product#
3   PREFIX xcompany: http://www.xcomp.com
4   sales:tv.htm          product:brand          xcompany:xbrand.htm
```

Figure 3.7 Triple Representation of Resources as Properties

Untill now demonstrated example RDF resources have simple properties, but what about complex properties? Imagine a case that birth date information needs to be given, however instead of giving it as a single string; it needs to be given as separate pieces of information (day, month, and year). Then a structured property is needed which aggregates birth date information. Consider the following example RDF document in Figure 3.8. In this example the resource "http://www.sales.com/asoylu" has properties "name", "surname", and "age" whose vocabulary is defined in "http://www.sales.com/person#". The age property is another resource, "http://www.sales.com/asoylu/birthdate", which is actually a structured aggregate property. Birth date resources defined by "byear", "bmonth" and "bday" properties by using another "rdf:description" element.

```
1       <?xml version="1.0"?>
        <rdf:RDF
2       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:person="http://www.sales.com/person#">
3       <rdf:Description rdf:about="http://www.sales.com/asoylu">
4           <person:name>Ahmet</person:name>
5           <person:surname>Soylu</person:surname>
6           <person:age rdf:resource=" http://www.sales.com//birthdate/asoylu" />
7       </rdf:Description>
8       <rdf:Description rdf:about="http://www.sales.com/birthdate/asoylu">
9               <person:byear>1984</person:byear>
10              <person:bmonth>11</person:bmonth>
11              <person:bday>26</person:bday>
12      </rdf:Description></rdf:RDF>
```

Figure 3.8 Demonstration of Structured Property

Figure 3.9 demonstrates the graph representation of the RDF example in Figure 3.8. This one is more complicated than Figure 3.2 as each resource has also other simple properties.

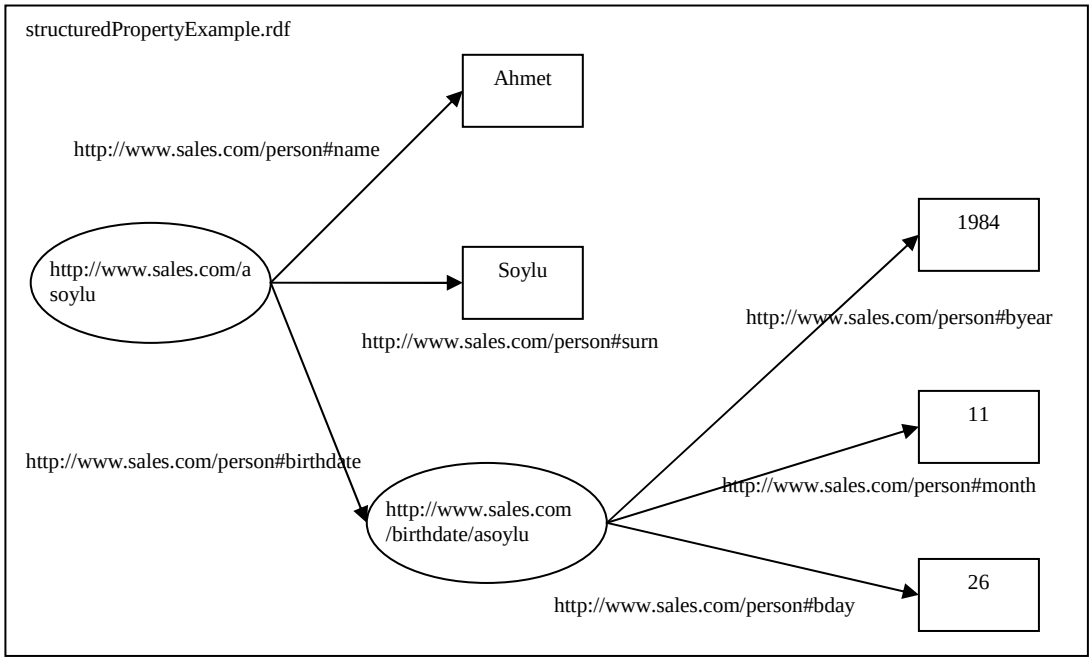


Figure 3.9 Structured Properties

Triple representation of the RDF example in Figure 3.8 is shown in Figure 3.10;

```
1  PREFIX  sales: http://www.sales.com/
2  PREFIX  pbirthdate: http://www.sales.com/birthdate/
3  PREFIX person: http://www.sales.com/person#
4  sales:asoylu                person:name                "Ahmet"
5  sales:asoylu                person:surname   "Soylu"
6  sales:asoylu                person:birthdate  pbirthdate:asoylu
7  pbirthdate:asoylu person:byear                "1984"
8  pbirhtdate:asoylu person:bmonth    "11"
9  pbirhtdate:asoylu person:bday                 "26"
```

Figure 3.10 Triple Demonstration of Structured Properties

How about a case when there is no actual URI for a resource which is actually a structured property? For instance we don't really need a universal identifier for birth date; therefore it is also possible and most common way to use blank nodes. Figure 3.11 demonstrates how it would be with using a blank node. It represents the same thing in Figure 3.9 however just uses a blank node for birth date resource.

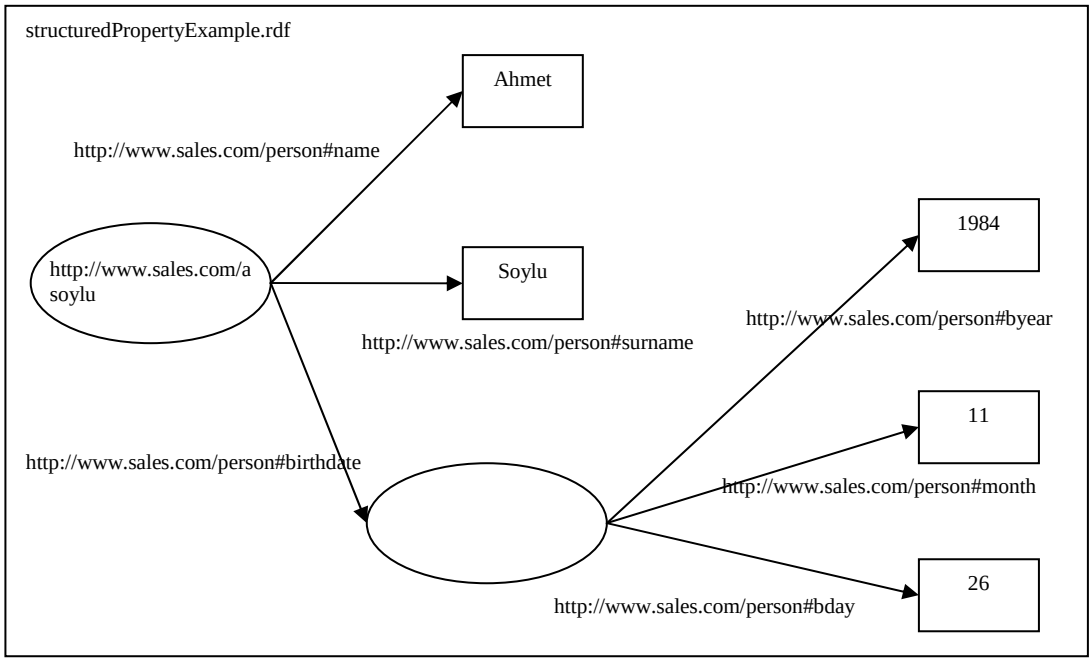


Figure 3.11 Blank Nodes

The below code, Figure.3.12, is triple representation of Figure 3.11, for blank node identifier the form "_:name" is used. In whole chapter the "name" variant will be selected randomly, any name might be given, however a processor application generates its own unique names for blank nodes.

```
1   PREFIX  sales: http://www.sales.com
2   PREFIX person: http://www.sales.com/person#
3   PREFIX person: http://www.sales.com/person#
4   sales:asoylu                person:name                 “Ahmet”
5   sales:asoylu                person:surname   “Soylu”
6   sales:asoylu                person:birthdate  _:asoylubirthdate
7   _:asoylubirthdate person:byear                “1984”
8   _:asoylubirthdate person:bmonth    “11”
9   _:asoylubirthdate person:bday                 “26”
```

Figure 3.12 Triple Demonstration of Blank Nodes

The Figure 3.13 demonstrates how a blank node is used in a RDF document by altering the example in Figure 3.8. In such a case this most common way is using a blank node identifier. A blank node identifier used to identify a blank node within a particular RDF/XML document but, unlike an URIref, it is unknown outside the document in which it is assigned [24]. At line 6, instead of “rdf:resource”, “rdf:nodeID” is used as blank node identifier, this is the same in line 8 instead of “rdf:about”, a blank node identifier is used.

```
1   <?xml version="1.0"?>
    <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:person="http://www.sales.com/person#">
3   <rdf:Description rdf:about="http://www.sales.com/asoylu">
4       <person:name>Ahmet</person:name>
5       <person:surname>Soylu</person:surname>
6       <person:age rdf:nodeID="asoylubirthdate" />
7   </rdf:Description>
8   <rdf:Description rdf:nodeID="asoylubirthdate">
9         <person:byear>1984</person:byear>
10        <person:bmonth>11</person:bmonth>
11        <person:bday>26</person:bday>
12  </rdf:Description>
13  </rdf:RDF>
```

Figure 3.13 Demonstration of Blank Nodes

Until now all the values we used for property values were plain literals. They were not declared as an integer, string or so on. Consider the below example triple;

```
1   _:asoylubirthdate person:bday                 “26”
```

Figure 3.14 Typed Literals Sample

It is not indicated above triple piece that the value 26 is actually an integer, it might be even considered as string that starts with ‘2’ continued with ‘6’. However the

below triple piece clearly indicates that this “bday” is an integer. This type of literals called Typed Literals.

1	_:asoylubirthdate	person:bday	“26”^^xsd:integer

Figure 3.15 Triple Demonstration of Typed Literals

Example RDF code in Figure 3.16 demonstrates how Typed Literals are used in RDF documents. “rdf:datatype” attribute at line 11 references an URI which defines the data type of this property value. The whole URI is not available in “rdf:datatype” attribute, instead it is abbreviated. An XML entity declaration is used to associate “xsd” with name space URI reference for XML Schema data types. The name space URI could also directly written into “rdf:datatype”attribute. “xsd:string”, “xsd:boolean” and “xsd:date” are some of other available data types.

```
1   <?xml version="1.0"?>
    <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:person="http://www.sales.com/person#">
3   <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
4                                                   * * *
5   <rdf:Description rdf:nodeID="asoylubirthdate">
6                                                   * * *
7         <person:bday rdf:datatype="&xsd;date">26</person:bday>
8   </rdf:Description>
9   </rdf:RDF>
```

Figure 3.16 Demonstration of Blank Nodes

Another type of abbreviation could be done via “rdf:ID”, the following example in Figure 3.17 demonstrates how it is used. At line two there is a new attribute used “xml:base”; it declares a base URI for this document. After this declaration, for any resource which resides in this base URI, it is not needed to given whole URI of resource; instead resource name will be enough like at line 4 and 8. This is done via using “rdf:ID” instead of using “rdf:about”. Actually defining a base URI might not be necessary depending on the situation. If the URI of the document is the same as the URI of resources then it is not needed to define a base, because if it is not defined the document URI will be accepted as a base URI. The concatenation is done via “#” element, considering the first resource it will be; “www.people.com/people#asoylu”.

```
1   <?xml version="1.0"?>
    <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:person=http://www.sales.com/person#
    xml:base="www.people.com/people">
3   <rdf:Description rdf:ID="asoylu">
4       <person:name>Ahmet</person:name>
5                                           * * *
6   </rdf:Description>
7   <rdf:Description rdf:ID="dsoylu">
8       <person:name>Ahmet</person:name>
9                                           * * *
10  </rdf:Description>
11                                          * * *
12  </rdf:RDF>
```

Figure 3.17 Demonstration of “rdf:ID” Attribute

RDF also enables assignment of a resource as an instance of specific types. This kind of resources called as typed nodes. For this purpose “rdf:type” property is used. Consider the example RDF code in Figure 3.18.

```
1   <?xml version="1.0"?>
    <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:person=http://www.sales.com/person#
    xml:base="www.people.com/people">
3   <rdf:Description rdf:ID="asoylu">* * *
4       <rdf:type rdf:resource="http://www.sales.com/terms/Person"/>
5       <person:name>Ahmet</person:name>
6    * *  *
7   </rdf:Description>
8   </rdf:RDF>
```

Figure 3.18 Demonstration of “rdf:type” Property

Below triple, Figure 3.20, representation belongs to example RDF document in Table 3.18

```
1   PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
2   PREFIX  people: http://www.people.com/people
3   PREFIX  person: http://www.sales.com/person#
4   PREFIX  terms :  http://www.sales.com/terms/
5   people:asoylu           person:name             “Ahmet”
6   people:asoylu           person:surname   “Soylu”
7   people:asoylu           person:birthdate  _:asoylubirthdate
8   people:asoylu     rdf:type                 sales:person
9   _:asoylubirthdate person:byear             “1984”
10  _:asoylubirthdate person:bmonth   “11”
11  _:asoylubirthdate person:bday              “26”
```

Figure 3.19 Triple Demonstration of “rdf:type”

Figure 3.19 demonstrates how an “rdf:type” property is shown in graph representation.

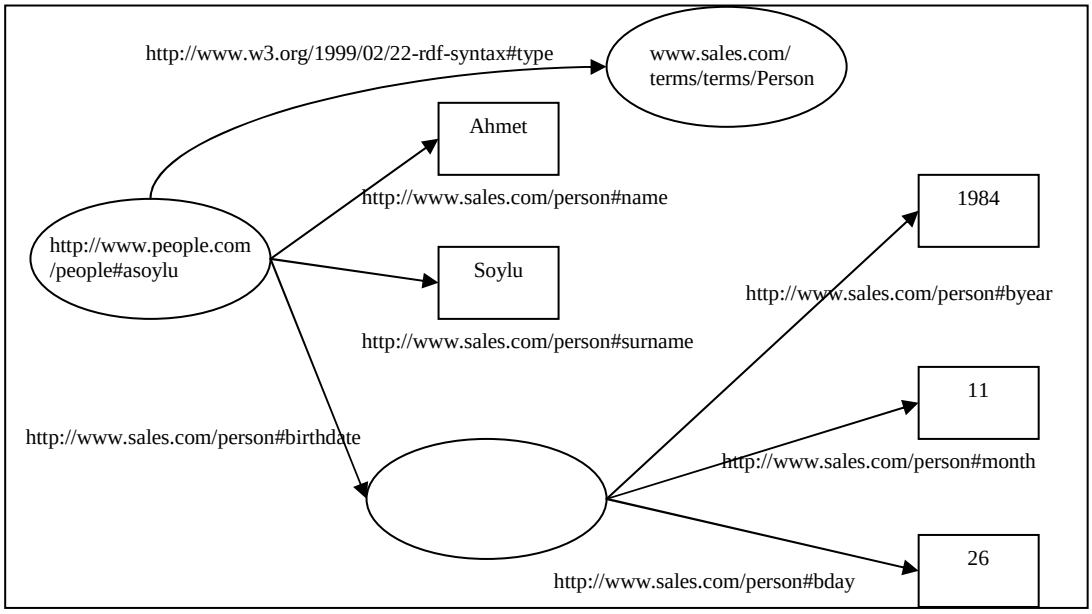


Figure 3.20 “rdf:type” Demonstration

RDF container elements are another type of RDF elements which are used describe group of things such as authors of a paper. The contained things called as members and they can be resources or literals. “<rdf:Bag>”, “<rdf:Seq>”, “<rdf:Alt>” elements are RDF container elements. “<rdf:Bag>” element used to describe an unordered list of values whereas “<rdf:Seq>” element is used to describe ordered list of values. “<rdf:Alt>” element is used to describe list of alternative values. The example below is demonstration of “<rdf:alt>” element which describes a tool and gives alternatives for its version as “full” or “trial”. Use of other RDF container elements is the same way as shown on Figure 3.21.

Each member have the name in the form of rdf:_n, n is a decimal number with no leading number and starts from one, “rdf:li” is used to avoid explicitly numbering each member. However triple and graph representation will use “rdf:_n” form, this transformation is done automatically by processing program.

```
1   <?xml version="1.0"?>
    <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:software="http://www.sales.com/software#">
3   <rdf:Description rdf:about="http://www.sales.com/xtool">
4       <software:version>
5           <rdf:Alt>
6               <rdf:li>Full</rdf:li> <rdf:li>Trial</rdf:li>
7           </rdf:Alt>
8       </software:version>
9   </rdf:Description> </rdf:RDF>
```

Figure 3.21 Demonstration of RDF Container Elements

A blank node is created in the type of “alt”, “seq”, or “bag” and members assigned to this node as properties. The below triple demonstration belongs to Figure 3.21;

```
1   PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
2   PREFIX software: http://www.sales.com/software#
3   PREFIX sales: http://www.sales.com
4   sales:xtool          software:version      _:z
5   _:z                  rdf:type              rdf:Alt
6   _:z                  rdf:_1                      “Full”
7   _:z                  rdf:_2                      “Trial”
```

Figure 3.22 Triple Demonstration of RDF Container Elements

The Figure 3.23 below is graph demonstration of example RDF document in Figure 3.21.

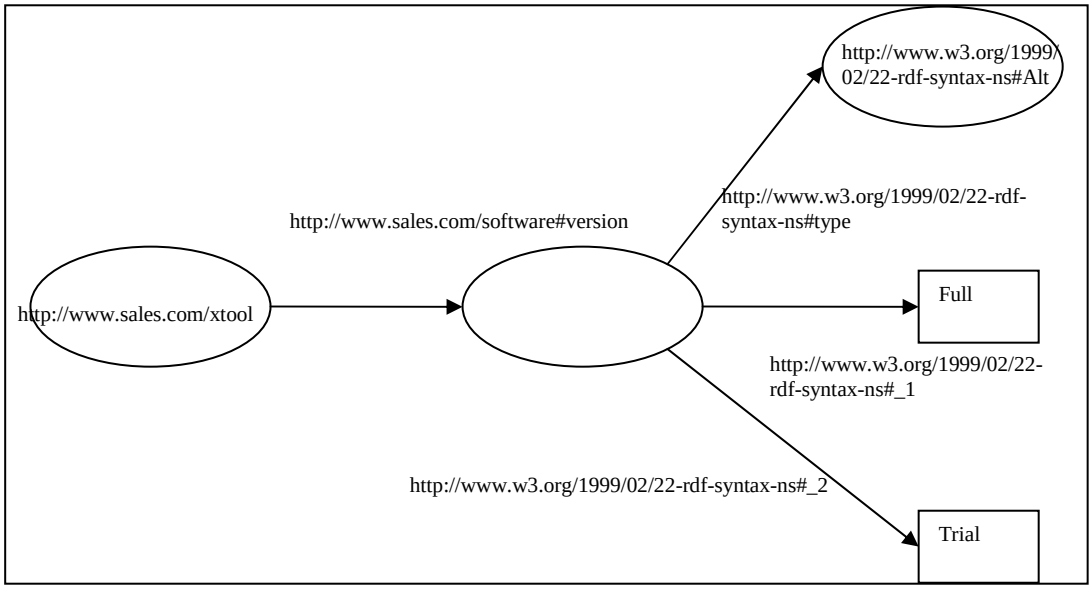


Figure 3.23 Graph Demonstration of “rdf:alt” Container

Members do not necessarily need to be plain literals, they can also be resources, in this case “rdf:resource” attribute is used inside the “rdf:li” element. Also members can be duplicate elements such as “software:author” as there might be several authors of a software. Members also might be different elements such as “software:author” and “software:expire_time”. It is important to note that the members of a container are not expected to be the full list, there might be other elements but not included in the list, the element in the list is not supposed to be complete. “rdf:parseType=’Collection’“ is used for such cases.

The RDF example in Figure 3.24 demonstrates use of “rdf:parseType=’Collection’“. A collection is assumed to be a complete list where a container is not. At line four of below example parse type defined as to be a collection and some resources listed as members of this collection.

```
1   <?xml version="1.0"?>
    <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:team="http://www.xteam.com/vocab#">
3   <rdf:Description rdf:about="http://www.xteam.com/ourteam">
4       <team:players rdf:parsetype= "collection">
5         <rdf:Description rdf:about="http://www.xteam.com/players/ahmet"/>
6         <rdf:Description rdf:about="http://www.xteam.com/players/hasan"/>
7       </team:players>
8   </rdf:Description> </rdf:RDF>
```

Figure 3.24 Demonstration of Collections

The graph in Figure 3.25 belongs to RDF document in Figure 3.24, each member of collection is a blank node which consists of original member itself referred with “rdf:first” element in this case a resource and a blank node pointing out next member with a “rdf:rest” element. Then the next member also consists of original member itself and a blank node pointing out the next member of collection. The last member of collection references to “rdf:nil” element to close collection.

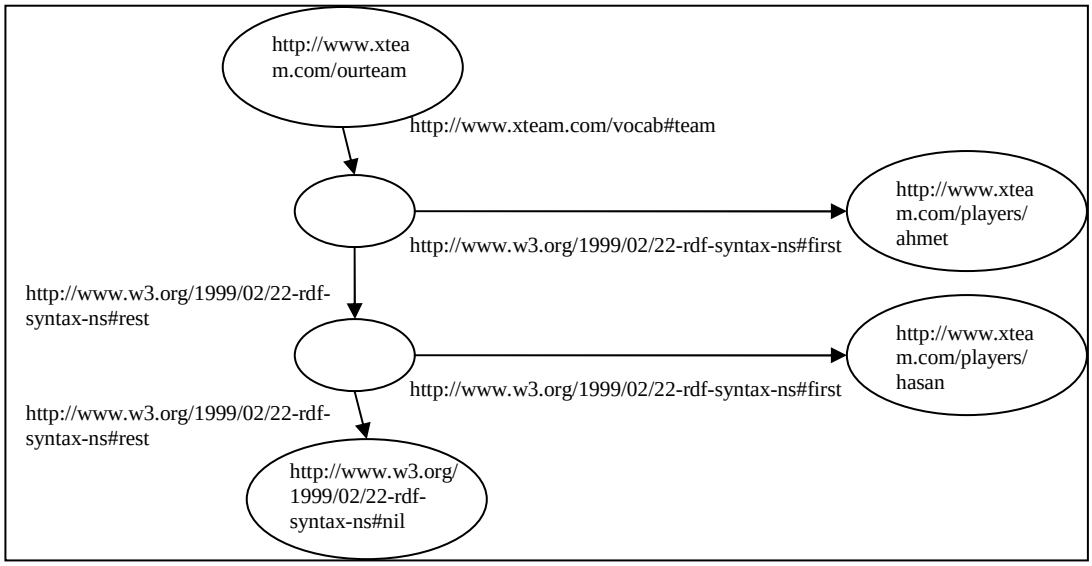


Figure 3.25 Graph Demonstration of Collections

Therefore according to explanation of above graph, the same rdf document can be also created by using “rdf:first”, “rdf:rest” and “rdf:nil” elements like in Figure 3.25. The RDF document itself is quite self descriptive.

```
        <?xml version="1.0"?><rdf:RDF
1       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:team="http://www.xteam.com/vocab#">
2       <rdf:Description rdf:about="http://www.xteam.com/ourteam">
3            <team:players rdf:nodeID="s1" />
4       </rdf:Description>
5       <rdf:Description rdf:nodeID="s1">
6           <rdf:first rdf:resource="http://www.xteam.com/players/ahmet"/>
7           <rdf:rest  rdf:nodeID="s2" /></rdf:Description>
8       <rdf:Description rdf:nodeID="s2">
9           <rdf:first rdf:resource="http://www.xteam.com/players/hasan"/>
10          <rdf:rest  rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil" />
11      </rdf:Description> </rdf:RDF>
```

Figure 3.26 Demonstration of Collections

The code piece in Figure 3.27 is triple representation of RDF documents in Figure 3.24 and Figure 3.26. They both represent the same thing therefore it is normal that they have same triple representation.

```
1   PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
2   PREFIX team: http://www.xteam.com/vocab#
3   PREFIX xteam: http://www.xteam.com/
4   PREFIX xplayers: http://www.xteam.com/players/
5   xteam:ourteam        team:players        _:s1
6   _:s1                 rdf:first           xplayers:ahmet
7   _:s1                 rdf:rest            _:s2
8   _:s2                 rdf:first           xplayers:hasan
9   _:s2                 rdf:rest            rdf:nil
```

Figure 3.27 Triple Demonstration of Collections

Other important parse type attribute values are “Literal” and “Resource”. “rdf:parsetype=’Resource’” attribute says that the content of an element is to be interpreted as the description of a new (blank node) resource. “rdf:parsetype=’Literal’” allows users to use XML fragments as a value of a property.

```
1   <?xml version="1.0"?>
2   <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
    <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:team="http://www.xteam.com/vocab#">
4   <rdf:Description rdf:about="http://www.xteam.com/ourteam/asoylu">
5       <team:player rdf:parseType="Literal">
6           <team:pname><em>Ahmet</em></team:pname>
7       </team:player>
8       <team:salary rdf:parsetype= "Resource">
9           <rdf:value rdf:datatype="&xsd;decimal">240000</rdf:value>
10          <team:units rdf:resource="http://www.example.org/units/euro"/>
11      </team:salary>
12  </rdf:Description></rdf:RDF>
```

Figure 3.28 Demonstration of “Literal” and “Resource” Parse Type.

The RDF document in Figure 3.28 demonstrates how “Resource” and “Literal” parse type attribute values are used. At line 5 “rdf:parsetype=’Literal’” is employed, it does not create any blank node in the graph where “rdf:parsetype=’Resource’” which is employed at line 8 creates a blank node.

3.1.2 RDF Schema

RDF also enables users to define their application specific classes and relations similar to the way object oriented languages provide in some respects. It does not provide application specific classes, but instead it enables users to describe custom

classes by providing a type system. Actually it is done via an extension of RDF called RDF Schema. Users actually create their own RDF vocabulary by defining classes and relations among them.

```
1   <?xml version="1.0"?>
2   <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
3   <rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xml:base="http://example.org/schemas/world">
4   <rdf:Description rdf:ID="Animal">
5       <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
6   </rdf:Description>
7   <rdf:Description rdf:ID="Mamal">
8       <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
9       <rdfs:subClassOf rdf:resource="#Animal"/>
10  </rdf:Description>
11  <rdf:Description rdf:ID="Horse">
12      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
13      <rdfs:subClassOf rdf:resource="#Mamal"/>
14  </rdf:Description></rdf:RDF>
```

Figure 3.29 Demonstration of RDFS

RDF schema facilities are provided via another RDF vocabulary whose name space is “http://www.w3.org/2000/01/rdf-schema#” and whose prefix is usually “rdfs”. The RDF schema can be interpreted by RDF applications becuase it does not have any difference from normal RDF documents. However without built-in understanding of RDFS it is not possible for an application to understand what it means.

Classes are described using the RDF Schema resources “rdfs:Class” and “rdfs:Resource”, and the properties “rdf:type” and “rdfs:subClassOf”. Figure 3.29 demonstrates how RDF Schema deals with classes. At line 4 it starts to describe “http://example.org/schemas/world#Animal” resource, which is written in abbreviated form via using “xml:base”. Then at line 5 it says this resource is a type of, an instance of, “http://www.w3.org/2000/01/rdf-schema#Class”. At line 7, it starts to describe another resource “http://example.org/schemas/world#Mamal”, and then it says that it is a RDFS class and also a sub class of “Animal” class.

Below graph belongs to Figure 3.29, one important thing is that rdf classes are transitive, in below example “Horse” class is sub class of “Mamal” and “Mamal” class is sub class of Animal, this enables any application capable of understanding

RDF Schema to interfere that any instance of “Horse” class is also sub class of Animal.

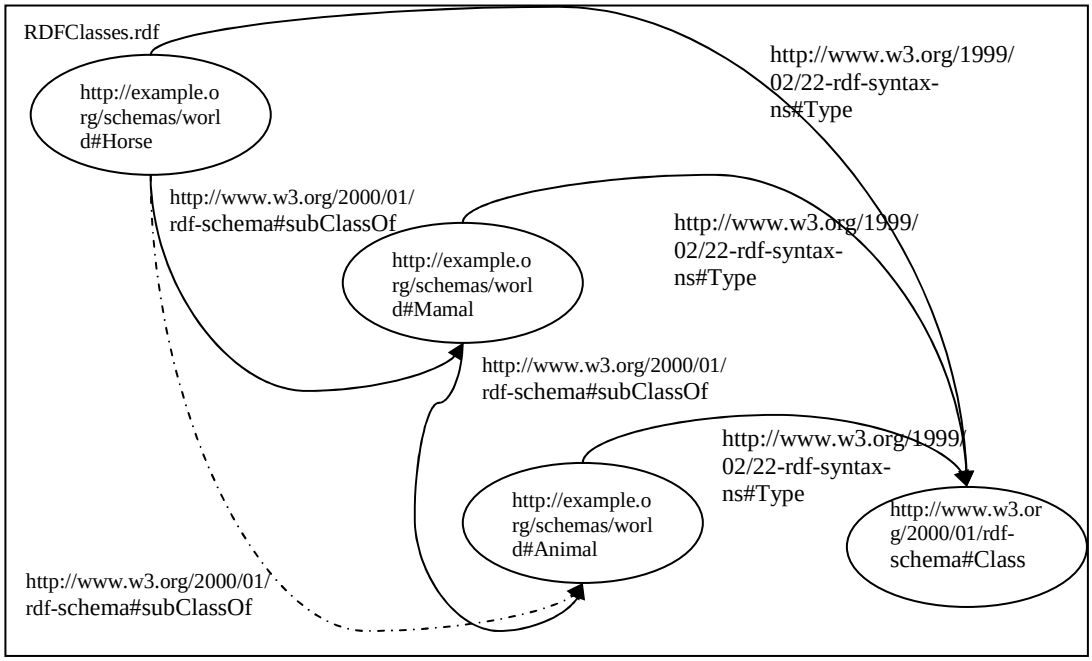


Figure 3.30 Graph Demonstration of Collections

The below triple representation demonstrates RDF classes, it is clearly seen that every RDFS class is also an instance of “rdfs:Class”.

1	PREFIX	rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#	
2	PREFIX	rdfs:http://www.w3.org/2000/01/rdf-schema#	
3	PREFIX	world: http://example.org/schemas/world#Horse	
4	world:Animal	rdf:type	rdfs:Class
5	world:Mamal	rdf:type	rdfs:Class
6	world:Horse	rdf:type	rdfs:Class
7	world:Mamal	rdfs:subClassOf	world:Animal
8	world:Horse	rdfs:subClassOf	world:Mamal

Figure 3.31 Triple Demonstration of RDFS Classes

The RDFS document can be abbreviated like in Figure 3.32, “rdf:Description” is replaced with the type name which is “rdfs:class”, note that “rdfs:Class” is also a type (class). However this abbreviation can be done for one class, if an instance belongs to more than one classes, other classes must be declared via “rdfs:type”, only one class declaration can be abbreviated.

```
1   <?xml version="1.0"?>
2   <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
3   <rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xml:base="http://example.org/schemas/world">
4   <rdfs:Class rdf:ID="Animal" />
5   <rdfs:Class rdf:ID="Mamal">
6         <rdfs:subClassOf rdf:resource="#Animal"/>
7   </rdfs:Class>
8   <rdfs:Class rdf:ID="Horse">
9          <rdfs:subClassOf rdf:resource="#Mamal"/>
10  </rdfs:Class>
11  </rdf:RDF>
```

Figure 3.32 Abbreviated Demonstration of RDFS

RDF Schema also provides a way to define properties which describe classes. Properties are described using the RDF class “rdf:Property”, and by the RDF Schema properties “rdfs:domain”, “rdfs:range”, and “rdfs:subPropertyOf”. Every property is an instance of “rdf:Property” class. “rdfs:Range” is used to denote the classes whose instances can be among the values of a property, and a property might have 0 or more “rdfs:range” properties. If a property has more than one range property then each value of this property has to be instances of each class defined by range properties. “rdfs:range” can also be used to say that the value of a property is a typed literal like “xsd:integer”, it is good practice to also define that a URIref is actually a data type via using “rdfs:Datatype”. “rdfs:Domain” property is used to denote that the particular property applies to specified classes. A property might have zero or more “rdfs:domain” property. If it is the case that a property has more than one “rdfs:domain” property, then any resource having this particular property has to be instance of all the classes defined by “rdfs:domain” properties. RDFS also enables specialization of properties as it does for classes, for this purpose predefined “rdfs:subPropertyOf” property is used.

```
1   <rdf:property rdf:ID="weight”>
2        <rdfs:domain  rdf:resource="#Animal"/>
3         <rdfs:range  rdf:resource="&xsd;decimal"/>
4   <rdf:property>
5
6   <rdfs:Datatype  rdf:about= "&xsd;decimal" />
```

Figure 3.33 Property Declarations

Figure 3.33 demonstrates usage of “rdfs:domain”, “rdfs:range” properties and “rdf:property” class and below there is triple representation of the example. It is assumed that the code piece belongs to code in Figure 3.33.

1	world:weight	rdf:type	rdf:property
2	world:weight	rdfs:range	xsd:integer
3	world:weight	rdfs:domain	world:Animal
4	xsd:decimal	rdf:type	rdfs:Datatype

Figure 3.34 Triple Demonstration of Property Declarations

It is said that RDFS type system is similar to the other object oriented programs’ type systems. However this match is not exact and involves many core differences. In an object oriented language a class is collection of specific properties, however in RDFS a properties applies specific classes (via “rdfs:range” and “rdfs:domain”). It means properties are global in RDFS where properties in an object oriented language are specific to classes. Besides in object oriented languages an class can not exists without a its particular properties and these constraint is applied by language. However in RDFS one application can simply see these declarations as just simple declarations and does not give any problem in a case of inconsistency where another application program behaves in opposite way.

3.2 SPARQL

SPARQL is a query language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL is "data-oriented" in that it only queries the information held in the models; there is no inference in the query language itself [25] [26].

There are four examples of simple SPARQL query below. The numbers at the beginning of each query does not belong to query, it is just for identification purposes. SPARQL syntax shows similarities with SQL. The both examples below equals to “SELECT * FROM <Table_Name>” or “SELECT att-1, att-2, … att-n FROM <Table_Name>” in SQL. However as the structure and semantic of the source that SPARQL queries differs from the source that SQL queries, this similarity

will be mostly in the sense of syntax. First two queries return same results over a particular RDF document. “SELECT” clause identifies the variables to be displayed in the result set. “WHERE” clause identifies the pattern to be matched with triples. The keywords that start with “?” or “$” are variables, and these signs can be used instead of each other. In the rest of this thesis, “?” notation will be used. In the “WHERE” clauses of the queries the variables associated with the subject, predicate and object. The first two queries does not impose any constraint on any elements of triple therefore whole triple set will be in the result, however in the third query the object is associated with “Ahmet”, therefore only the subject and predicate pairs who has “Ahmet” as object will be in the result set.

1	SELECT ?x ?y ?z WHERE { ?x ?y ?z }
2	SELECT * WHERE { $subject $predicate $object }
3	SELECT * WHERE { $subject $predicate “Ahmet” }
4	SELECT * WHERE {$subject <http://www.sales.com/person#name> “Ahmet” }

Figure 3.35 Sample SPARQL Queries

The result set below appears when query three is executed over the RDF document in Figure 3.8.

Table 3.1 Result of Third Query on Figure 3.8

subject	**predicate**
http://www.sales.com/asoylu	http://www.sales.com/person#name

The fourth query will only return subject “http://www.sales.com/asoylu” as result. The important thing to note there is URIs must be enclosed between “<” and “>” characters. The fourth query can also be written like in below query via using PREFIX in order to avoid writing long URI names every time.

5	PREFIX person:<http://www.sales.com/person#> SELECT * WHERE { ?subject person:name "Ahmet" }

Figure 3.36 Sample SPARQL Query with Prefix

The query in Figure 3.37, query six, fetches the surname property value of the resource whose name property value is “Ahmet”. The first triple pattern matches all

triple sets whose predicate is “person:name” and object value is “Ahmet”, and the second triple pattern matches all the triples whose predicate is “person:surname”. The important thing is how first pattern and second pattern is combined, this happens like a “join” operation in SQL which is applied to the first result set and the second result set based on “subject” variable. Therefore the same variable name “subject” is used in both patterns.

```
6   PREFIX  person:<http://www.sales.com/person#>
    SELECT  ?surname  WHERE { ?subject  person:name  "Ahmet".
                                        ?subject  person:surname ?surname }
```

Figure 3.37 Sample SPARQL Query Two Patterns

Query results can also contain blank nodes the second query applied over the RDF document in Figure 3.13 where there is a blank node and result set displayed on Table 3.2. It is important to note that blank node identifiers are assigned randomly; therefore it is not safe to base a query on blank node identifier.

Table 3.2 Result of Second Query on Figure 3.13

subject	**predicate**	**object**
http://www.sales.com/asoylu	http://www.sales.com/person#name	Ahmet
http://www.sales.com/asoylu	http://www.sales.com/person#surname	Soylu
http://www.sales.com/asoylu	http://www.sales.com/person#age	_:b
_:b	http://www.sales.com/person#byear	1984
_:b	http://www.sales.com/person#bmonth	11
_:b	http://www.sales.com/person#bday	26

“FILTER” can be used to restrict RDF literals; one common example is “REGEX” which allows “SQL like” style tests. The below example query seven involves “REGEX” function demonstration for “FILTER”. It returns any triple that has object value which starts with “Ahmet” word.

```
7   SELECT * WHERE {?subject ?predicate ?object
                            FILTER  regex(?object,"^Ahmet")
    }
```

Figure 3.38 Sample SPARQL Query Using Filter

The example query below, query eight, demonstrates another “FILTER”, it returns only the day of birth triples whose value is bigger than 20 or smaller than 18.

8	. PREFIX person: <http://www.sales.com/person#>
	SELECT * WHERE {
	?subject person:bday ?object
	FILTER regex(?object < 18 || ?object > 20)
	}

Figure 3.39 Sample SPARQL Query Using Regex

There are many other functions that can be used with filter like, isIRI, isBlank, isLiteral, sameTerms etc., whose names are already self-descriptive, and also allows some operands to be applied over variables. Those can be checked via SPARQL syntax.

RDF is a semi-structured data and it is not forced to use all elements, therefore in order to prevent pattern matching failures because of missing properties several ways exists. For instance imagine and RDF document in Table 3.3 where a new person added however without surname property, but when a query is applied which wants to fetch every name-surname pair even no surname value is defined. SPARQL allows this to be done via “optional” facility. The below query, query nine, is demonstration of “optionals” in SPARQL.

9	PREFIX person: <http://www.sales.com/person#> SELECT ?name ?surname WHERE {?subject person:name ?name. OPTIONAL {?subject person:surname ?surname}}

Figure 3.40 Sample SPARQL Query Using Optional

The query in Figure 3.40 fetches all name-surname pairs, and defines “surname” as optional, therefore for resources that surname is not defined a blank value will be returned for surname.

SPARQL also allows usage of “UNION” operation which enables several alternative triple patterns to match and take place in the result set. Consider the following example, query 10;

10	PREFIX person: <http://www.sales.com/person#> PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#> SELECT ?name WHERE { {?subject person:name ?object } UNION {?subject vCard:name ?surname }}

Figure 3.41 Sample SPARQL Query Using Union

The above query depends on a case where name of a person can be presented via vCard vocabulary or person vocabulary. In order not to miss any result item, a “UNION” operation, which combines result sets of both notations, is used.

Until now the assumption was that the graph, the RDF document, to be queried is loaded via SPARQL protocol by application program or API. However it is also possible to define this default graph via SPARQL. Moreover data about resources might reside in different graphs therefore different graphs might need to be queried within the same query. All these needs are solvable via “FROM”, “FROM NAMED” and “GRAPH” clauses. In the following query, query eleven, default graph to be queried is given in the query itself;

11	SELECT ?z FROM <c:/ex.rdf> WHERE { ?x ?y ?z }

Figure 3.42 Sample SPARQL Query Using From

.

“FROM” clause is used in a way to similar to the SQL, the graph is assumed to reside in computers physical storage, however it might also reside in the internet therefore a URI could also be given. It is necessary to note that multiple FROM clauses can be used, than combination of graphs creates the default graph.

“FROM NAMED” clause also defines the graphs to be queried, however it does not involve this graphs as a part of default graph, for instance if the eleventh query is written via “FROM NAMED”, the result set would be empty because no default

graph exists. When “FROM NAMED” clause is used “GRAPH” clause is used to identify graph. Consider the following query 12, it defines two named graphs and then fetches object values of each triple from this graphs, the variable “g” after graph keyword identifies the resource graph name which can be “ex1.rdf” or “ex2.rdf” for any result.

```
12  SELECT   ?g ?z
             FROM NAMED    <c:/ex1.rdf>
             FROM NAMED    <c:/ex2.rdf>
             WHERE  { GRAPH ?g { ?x ?y ?z } }
```

Figure 3.43 Sample SPARQL Query Using From Named

There are also several solution sequences and modifiers available in SPARQL like in SQL. The following query demonstrates several of them;

```
13  SELECT  DISTINCT ?z
              WHERE { ?x ?y ?z }
               ORDER BY ASC(?x)
               LIMIT 10
               OFFSET 5
```

Figure 3.44 Sample SPARQL Query Using Modifiers and Solution Sequences

The “DISTINCT” clause in query 13, simply eliminates duplicate results, and “ORDER BY ASC(?z)” clause orders result in ascending way with respect to subject of triple, “LIMIT 10” clause causes only top ten results to be fetched and “OFFSET 5” causes result set to start after 5th element in the result set.

“SELECT” clause is used during in all examples; however there are other clauses that can be used in order to get different result sets. Those are “CONSTRUCT”, “ASK” and “DESCRIBE”. “CONSTRUCT” clause returns an RDF graph constructed by substituting variables in a set of triple templates. “ASK” clause returns a boolean indicating whether a query pattern matches or not. Finally “DESCRIBE” clause returns an RDF graph that describes the resources found.

3.3 GRDDL

GRDDL introduces mark-up based on existing standards for declaring that an XML document includes data compatible with the Resource Description Framework (RDF) and for linking to algorithms (typically represented in XSLT), for extracting this data from the document [27].

GRDDL gleaning approach is based on either directly associating individual documents with their transformations or associating this transformation with “profiles” and name space documents. It is already noted this transformation usually represented in XSLT which is already introduced in this chapter, the status “name space” concept is also same. The “profiles” concept also will be introduced in coming pages. XSLT is already purposed to transform XML document in to form of other documents, here GRDDL defines a specification so that every XML or XHTML document can be associated with a transformation and any GRDDL aware agent can understand this transformation association and able to apply this transformation.

```
        <html xmlns=”www.w3.org/1999/xhtml”
                xmlns:grddl=”www.w3.org/2003/g/data-view#”
                grddl:transformation=”gleantitle.xls
                                        http:\\www.w3.org/2001/sw/grddl-wg/td/getauthor.xls”
1       >
2       <head>
3                                                       ***
4       <head>
5                                                       ***
6       </html>
```

Figure 3.45 Sample GRDDL Use [27]

This XHTML document in Table 3.45 associated with two GRDDL transformations, one is “http:\\www.w3.org/2001/sw/grddl-wg/td/getauthor.xls“ and other one is “gleantitle.xls”, however the location of the second transformation is not given therefore it is found based on the base URI. These two transformations will produce two RDF documents, and then these RDF documents will be combined into one singe RDF document.

```
1   <?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:product="product" xmlns:person="person"
    attributeFormDefault="qualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:data-view=”www.w3.org/2003/g/data-view#”
2       Data-viewl:transformation=”gleantitle.xls
                        http:\\www.w3.org/2001/sw/grddl-wg/td/getauthor.xls”

    >
3                                   ***
```

Figure 3.46 GRDDL Transformation via Schema

The example in Figure 3.46 was for the case where a transformation associated with an individual document. However it is also possible to associate a GRDDL transformation with the documents who share the same name space. The usage is as in Figure 3.46.

Another way of association is via “profiles”, a profile way of example is given in Figure 3.47 below;

```
1   <html xmlns="http://www.w3.org/1999/xhtml">
2   <head profile=”http://www.w3.org/2003/g/data-view”>
3   <link rel="transformation" href="http://www.test.com/t1.xsl" />
4   <link rel="transformation" href="http://www.test.com/t2.xsl " />
5                                   ***
```

Figure 3.47 GRDDL Transformation via Profiles

Chapter 4
Semantic Web and Microformats

4.1 Semantic Web

Currently information on the web pages are based on natural human language and aimed for human, therefore human-being is able to process information on the web, this means being able to deduce facts, to draw reasons, to associate and link different kind of information. It doesn't matter whether all the information presented in a single page or presented in several web pages, human is able to interpret, and locate and link necessary information for a particular task or activity, but what about computers?

Computers are not able to apply same actions (derive, associate, link etc.) automatically as humans do with the information on the web, they only deal with presentation and some other routine processing because web does not consist of automatically process able information rather it is pure source of information which is collection of documents linked to each other without any implied reason why. Semantic Web will offer variety of opportunities for web, a popular example of Tim Berners-Lee, who is a founder of HTML, HTTP, and Semantic Web Vision, is Pete and Lucy's hospital appointment example which is carried out by their agents via processing hospitals web page and other related pages about distance, appointment times, preferences and doctors etc. The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users [28]. Semantic web is not something totally new; it is extension of the current World Wide Web. The first basement of semantic web challenge is based on knowledge representation (KR), which has its roots before World Wide Web. KR's main

objective is to represent knowledge in a way that computers can understand and draw reasons from it. Semantic Web employs KR in a decentralized manner as web does. XML and RDF technologies already constitute a basis for knowledge representation for semantic web. In *Chapter 3*, XML, RDF, and RDFS introduced in details with other related technologies, an important point is that XML just gives structure to the data and does not say anything about meaning where RDF expresses the meaning; however RDF is not enough to describe complex relationships between objects such as cardinality constraints, unions, disjoint classes etc. RDFS gives a limited capability to define only simple relationships. At this point ontologies come up as a solution, they are used to define complex relationships and set of interference rules between data objects by providing more vocabulary. The most typical kind of ontology for the Web has a taxonomy (which defines classes of objects and relations among them), and a set of inference rules [28]. There are also many efforts for ontologies for semantic web, important ones are DAML and OWL ontology languages and both of these languages are based on XML, RDF and RDFS. XML, RDF and OWL can be seen as the layers of a three where each layer requires different skills and targets different needs [29]. For instance XML constitutes the first layer of this structure which requires fewer skills and offers solutions to the less complex problems then RDF or OWL offers. In *Chapter 3*, SPARQL and GRDDL is introduced SPARQL allows querying data structured with RDF and GRDDL allows us to harvest these data on web pages, this is one step more which allows us to use semantically structured data.

All these efforts move us to a point where information on the web can be moved into a form where it is machine understandable and semantically and ontologically enhanced. However XML, RDF, OWL etc. all these technologies aims machine understandability not for human, here it seems like we are at the beginning again. It would be a big burden to duplicate efforts, meaning creating a piece of data both in the form of such as RDF, for machines, and in the form of simple XHTML, for human. This would also bring other problems like when a change is needed; user need to both update metadata and presentation which would raise data consistency and syncronization problems. There are also efforts to prevent this double work burden. eRDF, RDFa and Microformats are leading technologies in this area, they allow embedding semantic data into web pages that can be both machine and human

understandable. RDFa is a W3C specification based on expressing structures via attributes of languages of XHTML and HTML [30]. eRDF, Embedded RDF, allows some very important parts of the RDF model to be embedded but does not attempt to extend this to the full RDF model [31]. eRDF is inspired by some basic principles of Microformats, those principles will be introduced later in this chapter. We already defined Microformats in *Chapter 1* as, XHTML tags that are used to embed information into web pages which are understandable both by machines and humans while considering the human as first priority.

All these technologies, RDFa, eRDF and Microformats, have their advantages and disadvantages coming from their nature. RDFa and eRDF is based on RDF framework and they just provide syntax in order to express RDF via XHTML attributes, attributes in an XHTML page are not viewable by users, so RDFa and eRDF uses these attributes to embed RDF vocabulary into XHTML page. The main difference between RDFa and eRDF is that, RDFa reflects full capability of RDF via completing missing abilities of XHTML with new attributes, however eRDF does not introduce any new attributes, and it prefers not to express RDF models which are not expressible by the capabilities of XHMTL such as blank nodes, containers, typed literals. This can be denoted as every eRDF structure is expressible via RDF but not every RDF structure is expressible via eRDF. RDF introduces new attributes therefore it is not compatible with XHTML 1.0 and breaks the syntax of XHTML, therefore any XHTML cleaning tool like “Tidy” [32] would break up the RDFa structure if it is applied, however RDFa is expected to be supported by later versions of XHTML.

RDFa and eRDF is based on the framework provided by RDF where Microformats offers both syntax and a set of fixed vocabulary as it does not rely on RDF or any other framework. Therefore Microformats are domain specific; you can not express everything with Microformat unless its syntax and vocabulary is defined by its community. This also implies that the extracting procedure is same for every eRDF and RDFa involving XHTML pages where it is different for every Microformat involving document. However as Microformat does not have any relation with RDF or such framework, it can be directly embedded into XHTML page without any need of both XHTML and RDF mark-up writing. eRDF and RDFa’s being based on RDF

enables users to mix and use different name spaces, however Microformat uses a flat name space, it is already predefined and you can not extend or mix it, besides you can not add a new metadata element. It is obvious that RDFa and eRDF provides much more flexibility then Microformats besides there strong ontology languages built on them such as OWL and DAML however yet there is no real life application of neither RDFa nor eRDF where there are many real life examples of Microformats both used by hobbyists or some enterprises like “Yahoo Locals”. However it is important to note that Microformats does not neither aim to be a panacea for all taxonomies, ontologies or and other such abstractions not aims to infinitely extensible or open ended [33].

While Microformats can encode explicit information to aid machine readability, Microformats do not address implicit knowledge representation, ontological analysis, or logical interference [34]. However simplicity of Microformats moves it to the practical real life; that is why people call Microformats as “lower case” semantics. Besides Microformats does not aim to reflect everything, it is developed by community so anyone can not simply offer a new Microformat and every domain is not suitable to be expressed with Microformats. They usually base their implementations to express well-established and accepted metadata-standards such as Dublin Core, iCal, vCard, Foaf, Atom etc. For instance using the hAtom Microformat, a website becomes the feed by intelligently marking up items. There is no longer a need to publish the same content in other formats like RSS since applications can extract raw Atom data from the (X)HTML of the website, again without converting to and from RDF [35]. Many people think that Microformats has its place now, and it allows us to reach of limited benefits of semantic web, and one day people needs will reach at the borders of these limitations and then RDFa and eRDF will have their place and they are expected to clear their weaknesses till that time.

Microformats will be under consideration of this thesis as it has many real life applications and might be a good instrument for e-Learning metadata standards and might offer variety of opportunities for e-Learning domain. Rest of this chapter will include details on Microformats.

4.2 A deeper look at Microformats

Semantic web could evolve from a collection of loosely linked pages to an enormous database that can be searched, filtered, and re-assembled in new ways. [36] Microformats offer a simple way in order to do so, which does not require any extra work, anyone with basic HTML knowledge can be able to employ Microformats. Instead of throwing away what works today, Microformats intend to solve simpler problems first by adapting to current behaviors and usage patterns (e.g. XHTML, blogging). [37]

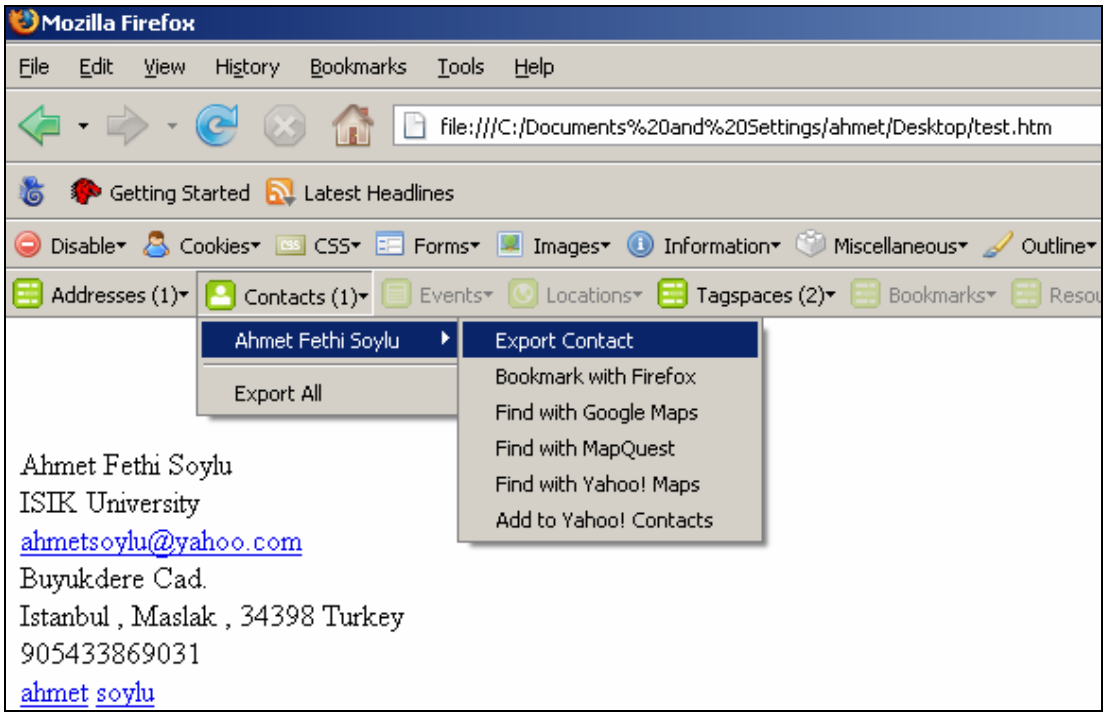


Figure 4.1 Example Use of Microformats

Figure 4.1 demonstrates use of Microformats, the simple HTML page shown includes an hCard Microformat. hCard Microformat is based on vCard which is a standard defining the format of an electronic business card, it offers variety of elements to be used such as name, organization, address etc. and how they should be used. As it is said Microformats are based on well established standards, vCard is the base of hCard. The example is demonstrated in a Firefox browser which has a Microformat plugin, “Operator” [38], installed on it. This plugin is capable of detecting several kind of Microformats, as it is shown in the given snap-shot, the contact information is detected, it can be exported to vCard format and can be used

in vCard supported applications, or several other operations are possible depending on the extractor application.

4.2.1 POSH

The first step before going further into Microformats world in the sense of either creating a new Microformat, or understanding how Microfrmats work, or employing Microformats in own web pages, is POSH. It means “Plain Old Semantic HTML”, and it aims separating presentational and semantic behavior of HTML 4.01 or XHTML 1.0 by using semantic elements and attributes separate from presentational elements and attributes. Furthermore while leaving semantic elements and attributes on XHTML page; it moves all presentational behavior over CSS, Cascading Style Sheet, files.

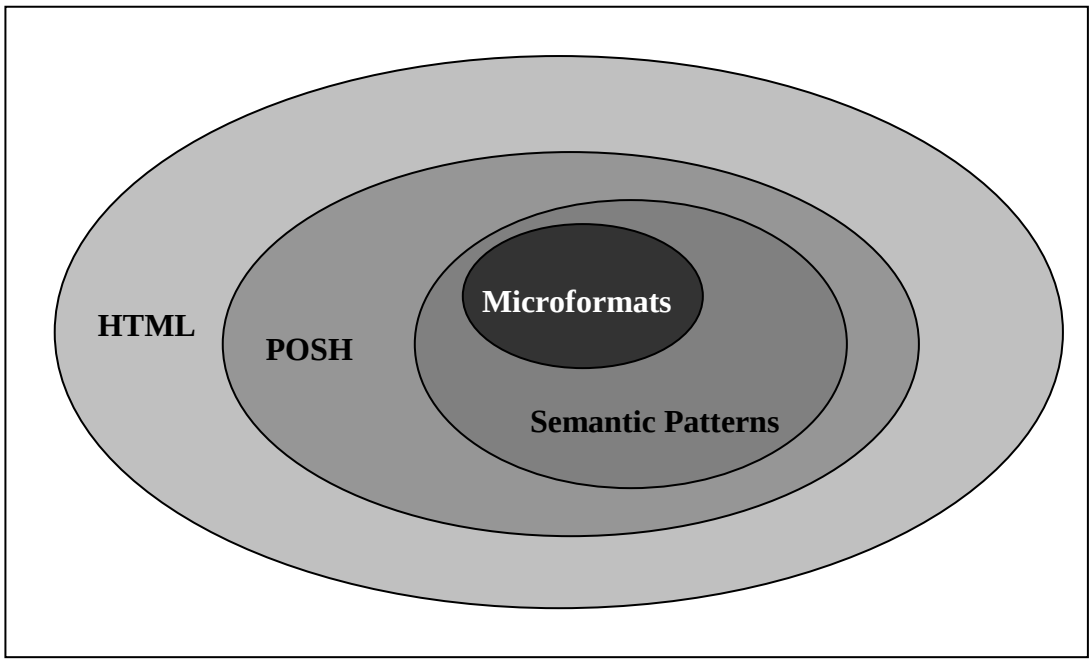


Figure 4.2 HTML, POSH and Microformats [39]

Some important principles for a valid POSH, can be summarized and sampled as follows. First of all the XHTML pages must be validated, that means these pages must follow basic syntax and rules of XHTML like correct nesting, and should use Strict HTML4.0/XHTML1.0. Secondly, semantic markup and presentational markup should not be mixed into each other, instead presentational issues should be handled

via CSS, for instance to create paragraph headings and sub-headings, “<h1>Heading 1<h1>” format should be used which is a semantic markup element for headings (h1, h2… etc.). Its presentational behaviors might be presented via CSS as follows; “h1 {font: bold xx-large; }” which would present heading in bold with a large font size. A wrong use to represent same heading with mixing presentational and semantic elements would be as follows; “<font size=’5’> This is a heading <font>”. Fewer HTML elements should be used, many presentational elements and attributes can easily represented with CSS such as “color” attribute or “ ” attribute. Another example can be use of “
” element, whose affect can be easily accomplished via use of semantic “ ” element. Such as instead using “

This is a paragraph.

This is another paragraph

”, “ This is a paragraph This is another paragraph ” can be used. All XHTML elements should be used appropriately and should be used for their intended use, for instance “<h1>” should not be used for getting large text, or “” or “” element should not be used for getting bold text, or tables should be used for tabular data not for layout etc.

4.2.2 Basic Principles

Now some important basic principles of Microformats, which also inspired other works such as eRDF, will take place;

- **Don’t Repeat Yourself (DRY):** This principle is against the cases where meta-data and presentation is separated from each other and when update is needed both need to be updated. Data should reside only in a single place, and when metadata is updated, it has to reflect all layers.
- **Visible Metadata**: There have been several attempts to associate metadata with HTML pages like “meta” element which is invisible to the user but can be detected by computers. However “meta” element has been an instrument for abusing search bots such as Google or other web search engines, besides as it is not visible many have been forgotten and stayed out date. Visible metadata principle is a lesson learnt from these previous attempts and making metadata visible prevents these problems.

- **Re-use:** Microformats are built on widely adopted standards such as XHTML and it uses well established schemas and standards such as RSS, vCard etc. Besides enables Microformats to be compound meaning a Microformat can be used inside another Microformat. For instance a Microformat which requires expressing people should use hCard Microformat which is used to express people.
- A specific problem should be solved and solution should be as simple as possible.
- And the most famous one, “human first machine second”.

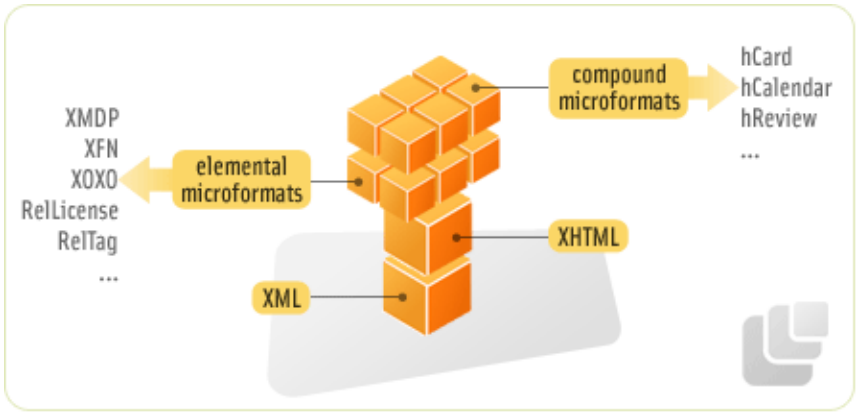


Figure 4.3 Foundation of Microformats [39]

4.2.3 Design Patterns

Design patterns give Microformat authors a vocabulary for expressing their ideas consistently with what has already been done [40].

Abbr Design Pattern: Allows encoding a text in a more formal way or machine readable way around presented human readable data for machines by using “<abbr>” element. In below in the first example, data value presented more formally for machines and in the second example name and surname of a person presented in more formal way where a nick is provided for presentation

1	<abbr class=”dtstart” title=”20080404T11:10:00-0200”>at 11:10 am of 4th April</abbr>
2	<abbr class=”author” title=”Ahmet Soylu”>Ahmet<abbr>

Figure 4.4 Abbr Design Pattern Sample

Date-time Design Pattern: This design pattern is used to make date time values both human readable and formally machine readable. The following case demonstrates usage;

3	<abbr class=”semantic_name” title=”YYYY-MM-DDTHH:MM:SS+ZZ:ZZ”>Date Time </abbr>

Figure 4.5 Date Time Design Pattern Sample

Class Design Pattern: This pattern is mostly used one among Microformat design patterns. It encodes semantic meaning via using “class” attribute, it is important to note that if there is a semantic proper XHTML element, such as “<h1>” for title, available it must be used, if it is not available “ ” or “” element should be used, any presentational element should be avoided. Below example demonstrates how an author name is encoded via “class” attribute.

4	Ahmet Soylu

Figure 4.6 Class Design Pattern Sample

Rel Design Pattern: This design pattern is used to express the meaning of a link by using “rel” attribute. Here below an example is given;

5	<a href=”icamp.htm” rel=”tag”>iCamp</tag>

Figure 4.7 Rel Design Pattern Sample

Elemental Microformats: Aimed to solve minimal problems and used to be part of larger compound Microformats. “rel-license”, “rel-nofollow”, ”rel-tag” etc. are some examples.

Compound Microformats: A compound Microformat consists of elemental Microformats and HTML element where each one can be expressed by “rel” or “class” attributes. “hReview”, “hCalandar” , “hCard” etc. are some examples.

Include Pattern: This pattern allows data re-usability within the same source document. A Microformat structure that takes place in one portion of page can be reused included, in another Microformat structure. This pattern has some problems those are still waiting for solution that comes up differently in different browsers. This pattern is realized by either links or objects.

4.2.4 An example Microformat

The below example, Table 4.1, includes a hCard Microformat embedded XHTML code whose output is shown in Figure 4.1 above.This example is created with the help of hCard creator service, hCard Creator [41], of Microformats community. This creator and creators for other Microformats can be found in Microformats community page. The specification for hCard can also be found in their community page in details.

Properties of hCard are [42];

- Fn
- n(family name, given-name, additional name, honorific-prefix, honorofix-suffix)
- nickname, sort-string
- url, email(type, value), tel(type, value)
- adr(post-office-box, extended-address, street-address, locality, region, postal-code, country-name, type, value), label
- geo(latitude, longitude), tz
- photo, logo, sound, bday,
- title, role, org, (organization-name, organization-unit)
- category, note
- class, key, mailer, uid, rev

Basically “fn”, full name, and “n”, name, properties are mandatory rest of the properties are optional such as “photo”, “tags”, “url” etc., “n” property is a complex property which consist of “family-name”, “given-name”, “additional-name”, “honorific-prefix” and “honorific-suffix” in given order. Each property embedded as class names and their values are usually values of XHTML elements. However in some cases such as in a case of using “abbreviation” pattern the value is the value of “title” attribute of “<abbr>”, or when “<a>” element is used the for the “photo” property the value of “href” attribute makes the property value. Another point about specification, in Figure 4.8 the vCard elements “NAME”, “VERSION” and “SOURCE” is not gleaned from embedded Microformat and it is not part of the specification to embed this information, rather this information is extracted from the source of page such as title of page becomes “NAME” and URL of the page becomes “SOURCE” etc.

The Figure 4.8 gives the code which belogs to demonstration in Figure 4.1.

```
1                                                    * * *
2       <div id="hcard-Ahmet-Fethi-Soylu" class="vcard">
3       <span class="fn n">
4       <span class="given-name">Ahmet</span>
5         <span class="additional-name">Fethi</span>
6       <span class="family-name">Soylu</span>
7       </span>
8       <div class="org">ISIK University</div>
9       <a class="email" href="mailto:ahmetsoylu@yahoo.com">ahmetsoylu@yahoo.com</a>
10      <div class="adr">
11        <div class="street-address">Buyukdere Cad.</div>
12      <span class="locality">Istanbul</span>,
13        <span class="region">Maslak</span>,
14        <span class="postal-code">34398</span>,
15        <span class="country-name">Turkey</span>
16      </div>
17      <div class="tel">905433869031</div>
18      <div class="tags">
19      <a href="www.mysite.com/tag/ahmet" rel=”tag”>ahmet</a>
20      <a href="www.mysite.com/tag/ahmet" rel=”tag”>soylu</a>
21      <div>
22      </div>
23                                                   * * *
```

Figure 4.8 hCard Example

It is important to note again it does not matter which XHTML elements used, only the semantic class names assigned allows detection of Microformat properties. Using

presentational XHTML elements must be avoided, however Microformat should be easily presentational format able via CSS or even without CSS it should be presented in a human readable way. For instance in Figure 4.8 neither presentational XHTML elements are used nor CSS is used, however it is still viewable in a human readable way as it is shown in Figure 4.8.

Note that “rel-tag” elementary Microformat is also used in this example in lines 19 and 20, and these tags are also detected in Figure 4.1.

The code piece in Figure 4.9 belongs to the vCard representation of above embedded hCard Microformat after a conversion done with an extractor application such as a GRDDL agent. This is totally a valid vCard file which can be directly used with any supporting application such as Outlook. It is also important to note that vCard is a wildly accepted and used standard.

```
1   BEGIN:VCARD
2   PRODID:-//kaply.com//Operator 0.8//EN
3   SOURCE:file:///C:/Documents%20and%20Settings/ahmet/Desktop/test.htm
4   NAME:
5   VERSION:3.0
6   N;CHARSET=UTF-8:Soylu;Ahmet;Fethi;;
7   ORG;CHARSET=UTF-8:ISIK University
8   FN;CHARSET=UTF-8:Ahmet Fethi Soylu
9   UID:
10  EMAIL:ahmetsoylu@yahoo.com
11  ADR;CHARSET=UTF-8:;;Buyukdere Cad.;Istanbul;Maslak;34398;Turkey
12  TEL;TYPE=VOICE:905433869031
13  END:VCARD
```

Figure 4.9 vCard Format after Exporting

Chapter 5
Learning Object Metadata Standards

5.1 Learning Standards

Needs of e-Learning has evolved a lot since the expectations moved from generic features and functionalities (such as quizzes, calendars etc.) of tools and systems to pedagogic and discipline based needs and features. These increasing needs have populated e-Learning world with many tools and nearly every e-Learning related party tried to employ or employed their own tools and systems. These movements enforced significant existence of standards in e-Learning world because otherwise it is impossible to ensure the return of this huge amount of investment.

Standards, brings important benefits together if they really manage to have wild acceptance and practice in their targeted domains. These benefits referred in *Chapter 1* shortly and they can be summarized as follows;

- **Re-usability:** Learning content can be easily split up, re-used and formed again in order to create different contexts
- **Interoperability:** Multiple Learning Systems or tools can communicate each other, besides they can share and mix their learning resources.
- **Manageability:** Tracing of learning content and users by learning tools or systems becomes easy.
- **Accessibility:** Learning content can be accessed by several learning tools and devices, besides appropriateness of content can be ensured.
- **Durability:** Users are not bound to any specific system, tool or device, interoperability and reusability is ensured.

- **Scalability:** Functionalities of systems and tools and investment over these items can be increased depending on the needs and requirements.
- **Affordability:** It can be ensured that Learning investments are appropriate and risk free.

There are several works done on learning standards area conducted by different bodies such as IMS, IEEE, ADL etc. depending on the discipline, geography and market. These bodies usually conduct research on Learning Standards in general, and they usually offer bundle of standards together. It is not uncommon for these standard bodies to adapt one or several learning standards of other bodies as a part of their learning standards bundle. However it is obvious that each application might have its own requirements and it is also obvious that one standard can not cover needs of all different applications. Therefore applications have to choose from alternatives of these learning standards or have to adopt these standards according to their needs without breaking their compliance to the standards.

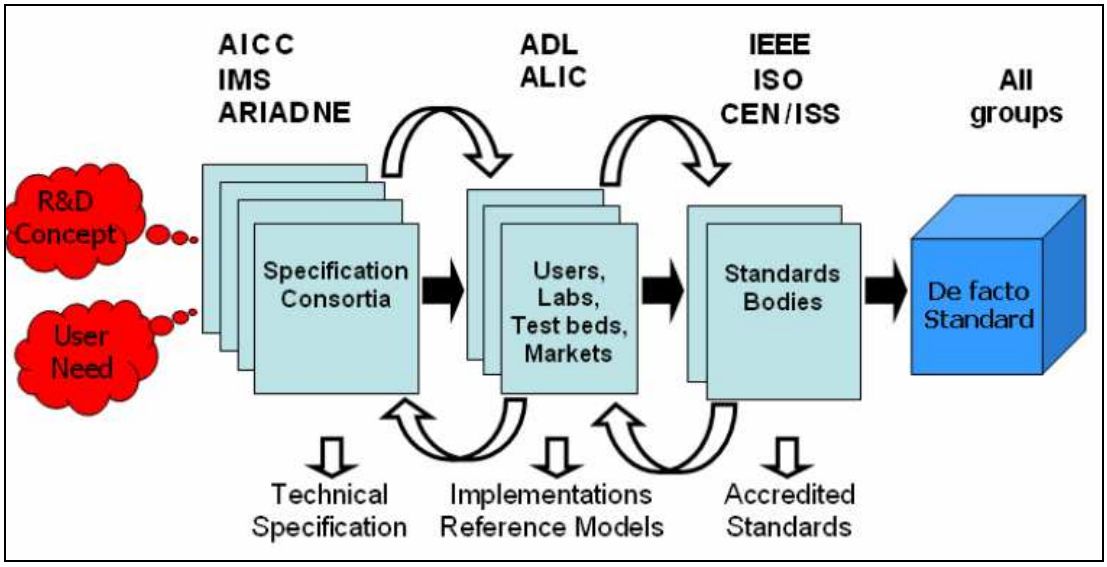


Figure 5.1 How Standards are Formed [7]

Figure 5.1 demonstrates how e-Learning standards are formed; actually same principles apply for the different domains. However before going in deep of this process, two important concepts need to be given. These are “de facto” and “de jure” standards, “de jure” standards are specifications which are accredited by an accredited body such as IEEE, ISO etc, and “de jure” standards are specifications

which have wild acceptance, adoption and use without any accreditation. Ideally a “de facto” standard is also a “de juro” standard. Now back to Figure 5.1, first some research bodies identifies possible solutions and prepares specifications depending on the requirements, then this specification is tested to check whether it works or not, then if it is thought that the specification is complete and works appropriately by the accredited body, it becomes an accredited standard and it is made available by accredited body. The Figure 5.1 gives some examples of bodies taking part in each stage.

Some example standards in E-Learning world are learning resource metadata, content packaging, question and test interoperability, content sequencing, tool interoperability, learner profile etc. Bodies those offer e-Learning standards, specifications or best practices summarized briefly in Table 5.1. Relatively important ones are IMS, ADL-Scorm and IEEE. IMS Global Learning Consortium is a global, non-profit, member association that provides means of shaping and growing the learning and educational technology industries through collaborative support of standards, innovation, best practice and recognition of superior learning impact [43]. IMS is like an umbrella group which involves several work groups (more than 30), besides it is in cooperation with other organizations such as ADL, IEEE, AICC, Dublin Core etc. ADL is a US Department of Defence initiative, ADL refers to Advanced Distributed Learning. ADL employs a structured, adaptive, collaborative effort between the public and private sectors to develop the standards, tools and learning content for the learning environment of the future [44]. Scorm is wildly known and adopted initiative of ADL which refers to Sharable Object Content Reference Model. Scorm provides a reference model and framework, this means it is not a standard it self but collection of individual standards and specifications (adopted from other bodies such as IMS, IEEE etc.). This reference model ensures interoperability, re-usability and manageability of learning content. IEEE employs IEEE Learning Technologies Standards Committee (IEEE LTSC). Its aim is to develop internationally accredited technical standards, best practices, and guides for learning technology [45]. It also collaborates with different bodies that create e-Learning standards and specifications. This collaboration occurs in a formal or informal manner. Accredited standards of IEEE have been wildly adopted by different bodies, and user parties.

Table 5.1 Snapshot of Standard Organizations [46]

Organisation	Geography	Primary Market	Function
IMS	Worldwide	K-12, HE, Corp, Gov	The largest and most comprehensive standards group, the IMS is composed of over 30 working groups addressing interoperable specifications.
ADL-SCORM	Worldwide	K-12, HE, Corp, Gov	White House/USDOD project. Sharable Content Object Reference Model (SCORM) is Advanced Distributed Learning's (ADL's) most widely known initiative. Current releases include a model for packaging learning content, a run-time API communications model, and, in SCORM 1.3, content sequencing
AICC	Worldwide	Corp	Aviation Industry Computer-based-training Committee (AICC) defines airline standards for computer managed instruction (CMI) that have been adopted by the e-Learning industry and is the basis for the SCORM run-time communication model.
OKI and Sakai	Worldwide	HE	Open Knowledge Initiative (OKI) defines service-based component architecture and a set of service interface definitions and APIs to support interoperability. Sakai is a new project aimed at refacturing the OKI SIDs for a product.
SIF	US and UK	K-12 and Schools	Schools Interoperability Framework (SIF) is an open specification for interoperability among K-12 instructional and administrative software applications.
Shibboleth/Internet 2	Worldwide	HE	Shibboleth is an open, standards based authentication integration solution allowing users to access controlled information securely without additional passwords.
IEEE	Worldwide	All	Institute of Electrical and Electronics Engineers (IEEE) working groups generally take IMS and other e-Learning specifications
JISC	UK	HE and FE	The Joint Information Systems Committee (JISC) supports further and higher education by providing strategic guidance, advice and opportunities to use Information and Communications Technology (ICT) to support teaching, learning, research and administration. JISC is funded by all the UK post-16 and higher education funding councils.
CETIS	UK	HE and FE	Centre for Educational Technology Interoperability Standards (CETIS) represents UK higher-education and further-education institutions on international learning technology standards initiatives.
BECTA	UK	All	British Educational Communications and Technology Agency (Becta) is the Government's lead agency for ICT in education. Working to support the development of ICT in education throughout the UK, Becta's unique contribution is to combine knowledge of the needs of education with an understanding of the power of technology.
JA-SIG uPortal	US and UK	HE	uPortal is a free, sharable portal under development by institutions of higher-education.
e-Portfolio Consortium	Worldwide	HE	The Electronic Portfolio Consortium, or ePortConsortium, is the collaboration of higher education and IT institutions working to define, design, and develop electronic portfolio software environment and management systems.

5.2 Learning Object Metadata Standards

In *Chapter 1*, a learning object is referred as “a digital part of a course ranging in size and complexity from a single graphic to entire course”. However it is important to note that there is no common definition of a learning object as the definition might change depending on the requirements of applications.

Whatever definition it has the important gain that comes together with “learning objects” is their small granularity. A learning object thought to be as smaller as it can be, so it can be re-used or re-assembled in different contexts. However it is important to note that these small learning objects should be meaningful. An example can be thought as follows, imagine a video file, an image file, or a text file as learning objects, they can together form a lesson and the lessons can together form a course. Providing that learning objects are independent from each other and smaller as much as they can be, their mobility, and re-usability is ensured. In this way investments can be saved because there will be no burden of creating same learning objects again and again for different contexts. According to Higgs *et al* (2003), essential characteristics of a learning object can be listed as follows;

- Independent
- Sharable / Reusable
- Interoperable
- Instructional value
- Discoverable
- Minimal context

However independence and small granularity will not provide a learning resource to be re-used completely. It must also ensure that these learning objects are interoperable. Learning Object Metadata can efficiently facilitate description, discovery and retrieval of learning content [47]. Yet it also enables interoperability of learning object by allowing different applications to share their learning objects. Metadata has been also introduced in *Chapter 1*; it is simply data about data. It is like catalogue carts in a library where each book or source tracked with information

charts. The information tracked might be the title, subject, place etc. of the resource. Metadata can be tagged directly over information or can be kept separately from the source. Some web technologies like RDF, XML has already been introduced and these technologies are being used to carry metadata information in general by using the vocabulary set, like vCard, developed and proposed by some bodies.

There are several bodies which do research and produce specifications or standards or best practices for Learning Object Metadata; most of these organizations have already been introduced in previous pages. The relatively important works are IEEE LOM (IEEE Learning Object Metadata), ADL-Scorm which have been known wildly and adopted by different bodies, organizations and other related parties. Dublin Core is also an important metadata standard even thought it is not specifically for Learning Objects but for general, it has an importance for Learning Object Metadata works. The coming pages will involve the works done in the area of Learning Object Metadata, however before proceeding further, an important concept need to be clarified briefly. This is “application profile” which has been denoted in *Chapter 1* shortly.

5.2.1 Application Profile

It is already denoted that it is not possible for a single learning object metadata standard to cover all requirements of different applications, in the mean time a learning object metadata set might also give much more than an application require. Here application profiles plays a key role, they aim to facilitate the application-oriented implementation of educational metadata specifications by allowing mixing and matching metadata elements in order to meet specific requirements for a particular context [48, 49]. Simply saying, an application profile might be a subset of a standard or can be mixture of elements from different standards.

Designing an application profile requires a systematic approach which should be carefully followed. A guideline has been published for application profile development in 2006 CEN/ISSS WS/LT workshop [50]. It is important to highlight the followings;

- One or more application profile can be selected as base standards,
- A target system which is compliant with the base standards must be also compliant with the application profile, or a target system compliant with the application profile should be compliant with the base standard.
- The application profile can not be less restrictive than the base standards which means application profile can make restrictions of base standards harder but can not relax them.

The basic guidelines can be summarized as follows in given order;

- **Requirements:** The first task must be identifying requirements in order to do so scope and purpose of the application profile must be defined, then according to scope and purpose use cases should be documented to identify requirements appropriately.
- **Selection of Data Elements:** Base standards need to be selected; those standards should be selected from the ones which share the same scope and purpose with proposed application profile.
- **Multiplicity Requirements:** The concept “Smallest Permitted Maximum” (SPM) is important here. It means at least how many times and element should appear and target system should process an application profile can reduce this number or can leave it equal, however can not increase this number.
- **Data Elements from Multiple Namespaces:** Proposed application profile might depend on multiple metadata schemas.
- **Local Data Elements:** An application profile may involve local data elements.
- **Obligation of Data Elements:** Typical values include, “Mandatory”, “Recommended”, “Conditional”, “Optional”. However application profile can not soften the obligation for instance can not reduce obligation from “Mandatory” to “Optional“, however can harden the obligation.
- **Value Space:** Value space defines the domain or set of values which the data element can have values from. This can be two fold, a vocabulary can be given or a reference to another standard can be given. As usual an application

profile can not be less restrictive. In vocabulary case application profile might choose a subset of specified vocabulary and in a case to referencing another standard application profile might choose an application profile of the referenced standard.

- **Relationship and Dependencies:** Complex relationships among data elements can be defined by not violating the rule which does not allow application profile to be less restrictive than base standards.
- **Data Type Profiling:** Example data types might be “LangString”, “DateTime”, “Duration” etc. which are some of the data types those belongs to LOM. Data types themselves are also a metadata schema therefore all the rules for application profile also apply data types.

5.2.2 IEEE LOM

This standard (IEEE LOM 1484.12.1-2002) is a multi-part standard that specifies Learning Object Metadata and for this standard, a learning object is defined as any entity, digital or non-digital, that may be used for learning, education or training [51] [10]. A vocabulary, extensions and description of semantics are involved in this standard. LOM has been accepted, adopted and used by many bodies and organizations such as ADL, IMS etc.

LOM vocabulary grouped into nine categories which are globally accepted. Those categories are summarized below briefly;

- **General:** In this category, general elements those describe the Learning Object with its general properties are grouped.
- **Lifecycle:** In this category, elements those are related with the history and current state of the learning object are grouped during its life cycle.
- **Metadata:** In this category, elements those describe metadata instance itself are grouped.
- **Technical:** In this category, technical properties and requirement of Learning Object are grouped.

- **Educational:** In this category, elements those describe educational and pedagogic properties of Learning Objects are grouped.
- **Rights:** In this category, elements those describe intellectual property rights and conditions of Learning Objects are grouped.
- **Relation:** In this category elements those describe relation of Learning Object with other Learning Objects are grouped.
- **Annotation:** In this category, elements those give comment on educational use of Learning Object grouped. Besides elements those give information about the comment provider are also grouped in under this category.
- **Classification:** In this category elements those describe classification of Learning Object with respect to a particular classification system are grouped.

In Table 5.2 LOM elements are listed with respect to their category in given order.

Table 5.2 LOM v1.0 Elements

#	Category	Elements
1	General	Identifier (Catalog, Entry), Title, Language, Description, Keyword, Coverage, Structure, Aggregation Level
2	Lifecycle	Version, Status, Contribute (Role, Entity, Date)
3	Metadata	Identifier (Catalog, Entry), Contribute (Role, Entity, Date), Metadata Schema, Language
4	Technical	Format, Size, Location, Requirement (orComposite (Type, Name, Minimum Version, Maximum Version)), Installation Remarks, Other Platform Requirements, Duration
5	Education	Interactivity Type, Learning Resource Type, Interactivity Level, Semantic Density, Intended End User Role, Context, Typical Age Range, Difficulty, Typical Learning Time, Description, Language,
6	Rights	Cost, Copyright and Other Restrictions, Description
7	Relation	Kind, Resource (Identifier (Catalog, Entry), Description)
8	Annotation	Entity, Type, Description
9	Classification	Purpose, Taxon Path (Source, Taxon (Id, Entry)), Description, Keyword

It is important to note that LOM elements have a hierarchical structure, and an element might be “simple” or “aggregate” element. Simple elements are leaves and they have value spaces and data types, however aggregate elements do have neither value spaces nor data types. In Table 5.2 “General.Identifier” is an aggregate element which contains some sub elements, however “General.Identifier.Catalog” is a simple element.

Table 5.3 A Sample from LOM v1.0 [10]

#	Name	Explanation	Size	Order	Val. Space	Data Type	Example
1.1	General	This category groups the general information that describes the learning object as a whole.	1	unspecified	-	-	-
1.1.1	Identifier	A globally unique label that identifies this learning object.	spm :10	unspecified	-	-	-
1.1.2	Catalog	The name or designator of the identification or cataloging scheme for this entry. A name space scheme.	1	unspecified	Repertoire of ISO/IEC 10646-1:2000	CharacterString, SPM:1000	“ISBN”, “ARIADNE”, “URI”
* * *							
1.7	Structure	Underlying organizational structure of this object.	1	unspecified	atomic: An object that is invisible. (in this context) collection: A set of objects with no relationships among them. networked: A set of objects with relationships that are unspecified. hierarchical: A set of objects whose relationships can be represented by a tree structure. linear: A set of objects that are fully ordered. (E.g: A set of objects that are connected with “previous” and “next” relationship.)	Vocabulary (State)	NOTE—A learning object with Structure=“atomic” will typically have 1.8:General.Aggregation-Level=1. A learning object with Structure=”collection”, “linear”, “hierarchical” or “networked” will typically have 1.8:General.Aggregation Level= 2, 3, or 4.
9	Classific.	This category describes where this learning object falls within a particular classification system.	spm :40	unordered	-	-	-
* * *							

Table 5.2 does not include whole details of the elements, however more details can be found in the standard document (LOM 1484.12.1, 2002). In this standard document every element is described by a “name”, “explanation”, “size”, “order” and “example”. In Table 5.3 a demonstration is given.

“Name” and “explanation” are already self descriptive. “Size” shows how many times this LOM element must appear. An exact number gives the number of exact occurrences, however if a Smallest Permitted Maximum, SPM, is defined this gives the minimum number of occurrences that an application should process. In example “1.1.1 Identifier”’ has SPM of 10 while “9 Classification” has SPM of 40. The rest of the elements given in example have an exact value of 1. “Order” is used to indicate whether the orders of values of an element are significant or not, this is used for the elements where a vocabulary is defined for value space. A famous example for this case can be the authors of a paper, the order of the authors are important because of they are usually listed according to the level of their contribution, or at least the first author is the main author of the paper. “Value Space” is used for simple elements, and it gives a set of values, a vocabulary, which is applicable for this element. A reference to another standard can be given for a value space, for instance vCard can be used as a value space for an element which refers to a person or organization. An example from the Table 5.3, “1.1.2 Catalog” value space refers to a standard where “1.7 Structure” value space refers to a set of values. “Datatype” refers to type of the value; this is also valid for simple elements. Finally “Example” gives a demonstration of the element use.

Data types for LOM v1.0 are also worth to mention here, the valid data types are; “LangString”, “DateTime”, “Duration”, “Vocabulary”, “CharacterString” or “Undefined”. Data type also might include a SPM value like “Size”, for instance a SPM indicates how much characters should be processed by application for “CharacterString” data type having values. LOM elements represent a hierarchy, and the numbering represents this hierarchy like “1.1.1” or “1.1.2”. Data types in LOM are also represented by a schema in standard document in the same format and descriptive elements in Table 5.3. In Table 5.4, this schema demonstrated basically. Whole definitions and descriptions can be found in standard document.

Table 5.4 LOM v1.0 Data Types

#	Datatype	Elements
1	LangString	LangString(Language, String)
2	DateTime	DateTime, Description
3	Duration	Duration, Description
4	Vocabulary	Source, Value
5	CharacterString	-

All elements of the LOM are currently optional; this means no of them is forced to be used. If the application profile rules are recalled, it was saying no obligation can be relaxed, as all elements are already optional there is no need to consider this rule. However when producing an application profile for LOM, if a set of new local elements to be added, if there is already a match between LOM elements and the new set, LOM elements should stay but the matching element in the new set should be ignored. Furthermore if an extension is about to be defined for a data element, no value space or data type should be defined for aggregate elements.

Another document, "IEEE LOM v1.0 Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata" (IEEE1484.12.3.2005), accompany standard document. This standard defines World Wide Web Consortium (W3C) Extensible Markup Language (XML) structure and constraints on the contents of XML 1.1 documents that can be used to represent learning object metadata (LOM) instances as defined in IEEE Std 1484.12.1-2002.1 and this standard defines the structure and constraints of the XML 1.1 documents in W3C XML Schema definition language [53].

5.2.3 ADL-Scorm – Scorm Metadata

SCORM is a collection of standards and specifications adapted from multiple sources to provide a comprehensive suite of e-learning capabilities that enable interoperability, accessibility and reusability of Web-based learning content [54].

Scorm Metadata is based on IEEE 1484.12.1-2002 LOM and IEEE1484.12.3.2005 Standard for Extensible Markup Language (XML) Binding for Learning Object Metadata. Approximately 64 LOM metadata elements are defined by Scorm. Scorm CAM (Content Aggregation Model) document provides definitions of all these

elements with their example XML bindings. While Scorm provides, best practices and requirements for efficient use of LOM, it also encourages other parties to decide on their own element sets and requirements.

```
1  <lom>
2      <general>
3          <identifier>
4              <catalog>URI</catalog>
5              <entry>http://www.example.com/object1</entry>
6          </identifier>
7          <title>
8              <string language="en">An example title</string>
9          </title>
10         <language>en</language>
11         <description>
12             <string language="en">Description</string>
13         </description>
14         <keyword>
15             <string language="en">Test, Object</string>
16         </keyword>
17         <coverage>
18             <string language="en">Animals</string>
19         </coverage>
20         <structure>
21             <source>LOMv1.0</source>
22             <value>atomic</value>
23         </structure>
24         <aggregationLevel>
25             <source>LOMv1.0</source>
26             <value>2</value>
27         </aggregationLevel>
28     </general>
29 </lom>
```

Figure 5.2 Scorm Metadata – General Category XML Binding Example

Above code piece is a demonstration of “General” category elements by Scorm. More details can be found in Scorm CAM (Content Aggregation Model) document [55]. The document provides examples, and practices and gives description, name space, multiplicity requirements and data type information for each element.

5.2.4 Other LO Metadata Standards and Application Profiles

There are several other initiatives which provide metadata standards, application profiles, best practices or reference models for learning object metadata. Some of them are listed and explained below briefly;

- **ARIADNE Metadata:** Current version of this specification is an application profile of LOM whose scope is Higher Education and Professional Training. Provides 43 elements, and 23 of these elements directly maps to the IEEE LOM elements. Early version of this specification was the basis of IEEE LOM.
- **UK Learning Object Metadata Core:** This is also an application profile of IEEE LOM which is specifically aimed for UK education. The main purpose is to provide best practices, guidelines to metadata implementers, users and creators. The current specification is not accompanied with bindings.
- **IMS Global Learning Consortium:** IMS and ARIADNE submitted a joint proposal to IEEE which resulted with the draft proposal of IEEE LOM and creation of IEEE LTSC working group [56]. IMS uses IEEE LOM as its basis now, and publishes best practice specifications. IMS/GLC and IEEE still have close cooperation for learning object metadata initiative.
- **CanCore Metadata:** Originally called Canadian Core, and it is totally based on IEEE LOM Standard and IMS specification. CanCore selects sub elements of LOM (currently 46 active elements) according to their minimalist approach and gives detailed information, practices and recommendations about these elements. More information about specification can be found in specification document [57].

5.2.5 Dublin Core

Dublin Core (DC) Metadata is not specifically produced for Learning Object Metadata but rather for general use. That is why it is considered in above Learning Object Metadata sets; however DC has a significant importance for Learning Object Metadata. It mainly aims facilitating discovery and retrieval of learning objects. Many standard and specification bodies adopted or used DC, one example can be CanCore.

DC is actually can be seen as lowest common denominator. DC provides 15 elements which can be considered pretty small with respect to IEEE LOM, this mean high

manageability for DC. Actually all elements of the DC can be mapped into IEEE LOM, Table 5.5 represents DC elements and their mapped IEEE LOM element.

Table 5.5 DC Elements Mapping over IEEE LOM Elements [52]

Dublin Core Data Element	IEEE LOM Data Element
DC.Identifier	1.1.2:General.Identifier.Entry
DC.Title	1.2:General.Title
DC.Language	1.3:General.Language
DC.Description	1.4:General.Description
DC.Subject	1.5:General.Keyword or 9:Classification with 9.1:Classification.Purpose equals “Discipline” or “Idea.”
DC.Coverage	1.6:General.Coverage
DC.Type	5.2:Educational.LearningResourceType
DC.Date	2.3.3:LifeCycle.Contribute.Date when 2.3.1:LifeCycle.Contribute.Role has a value of “Publisher.”
DC.Creator	2.3.2:LifeCycle.Contribute.Entity when 2.3.1:LifeCycle.Contribute.Role has a value of “Author.”
DC.OtherContributor	2.3.2:LifeCycle.Contribute.Entity with the type of contribution specified in 2.3.1:LifeCycle. Contribute.Role.
DC.Publisher	2.3.2:LifeCycle.Contribute.Entity when 2.3.1:LifeCycle.Contribute.Role has a value of “Publisher.”
DC.Format	4.1:Technical.Format
DC.Rights	6.3:Rights.Description
DC.Relation	7.2.2:Relation.Resource.Description
DC.Source	7.2:Relation.Resource when the value of 7.1:Relation.Kind is “IsBasedOn.”

DC Metadata initiative also involved in development of educational elements for Dublin Core to describe educational materials which is called DC-Ed. DC-Ed is extension of DC, it has 5 elements. DC-Ed is aims describing Learning Objects in a more general way rather than describing them extensively as IEEE LOM does. IEEE LTSC LOM Working Group and Dublin Core Metadata initiative has accepted a memorandum to cooperate for developing metadata. Thus DCMI-EMS (Dublin Core Metadata Initiative – Educational Metadata Set) is proposed which consist of 15 DC elements, 5 DC-Ed elements and 3 IEEE LOM elements. The DCMI-EMS metadata set is given in Table 5.6.

Table 5.6 DCMI-EMS Metadata Schema [58, 59]

Element Name	**Description**
Title (DC)	A name given to the resource
Contributor (DC)	An entity responsible for making contributions to the content of the resource
Creator (DC)	An entity primarily responsible for making the content of the resource. Datatype
Publisher (DC)	An entity responsible for making the resource available
Subject (DC)	The topic of the content of the resource.
Description (DC)	The topic of the content of the resource.
Date (DC)	A date associated with an event in the life cycle of the resource.
Type (DC)	The nature or genre of the content of the resource
Format (DC)	The physical or digital manifestation of the resource.
Identifier (DC)	An unambiguous reference to the resource within a given context
Language (DC)	A language of the intellectual content of the resource.
Source (DC)	A Reference to a resource from which the present resource is derived
Coverage (DC)	The extent or scope of the content of the resource.
Rights (DC)	Information about rights held in and over the resource.
Audience (DC-Ed)	A category of user for whom the resource is intended.
Audience.Mediator (DC-Ed)	An entity that mediates access to the resource.
Standards (DC-Ed)	A reference to an established education or training standard to which the resource is associated
Standards.Identifier (DC-Ed)	Where available, an identifier that serves to uniquely identify the standard being associated
Standards.Version (DC-Ed)	Information identifying the version of the standard being referenced (e.g., a year of publication, a version number, etc.)
InteractivityType (LOM)	The flow of interaction between this resource and the intended user
InteractivityLevel (LOM)	The degree of interactivity between the end user and this resource.
TypicalLearningTime (LOM)	Approximate or typical time it takes to work with this resource.

Chapter 6
Implementation

The work done here has several phases, the first phase is proposing an application profile, second phase is based on defining the XML, RDF bindings, XSLT files for transformations and proposing Microformat for this application profile, third phase consists of setting up an SQI service which is able to extract RDF data from different pages via GRDDL and makes these RDF files to be queried via SPARQL. An example semantic search engine which uses this SQI service will be created; this will be the final phase of the implementation.

6.1 Application Profile Proposal

In order to propose an application profile the guideline given in *Chapter 5* will be used roughly. Scope and purpose, use cases, defining data types, selection and definition of metadata elements, and finally providing XML and RDF bindings constitutes the mile stones of proposing the application profile for this thesis.

6.1.1 Scope and Purpose

The first step for creating an application profile is deciding on the purpose and scope of the application profile as it is indicated in *Chapter 5*. Purpose and scope of the application profile which are derived from the context of this thesis are as follows;

Purpose: The primary purpose of the proposed application profile is to facilitate search and harvesting of Learning Objects both by individuals and automated software applications such as agents. The minimalist approach defines the borders of this application profile; it is major to cover the common needs of different interested parties instead of providing a comprehensive profile because of the nature of the

domain that this application profile will be used for. This application profile will be used to develop a Learning Object Microformat and it is important to remind that schemas of Microformats are defined once and they are not subject to change often.

Scope: This application profile defines minimal set of metadata elements and their structure for Learning Objects. This application profile accepts any digital or non-digital entity, which is used as a part of digital learning activity, as Learning Object. Elements of this application profile are derived from IEEE LOM 1482-12-1 2002, and Dublin Core. This application profile's reference model and XML bindings are based on SCORM 2004 3rd Edition.

6.1.2 Use Cases

Three important use cases are provided below which demonstrates how the end user would make use of the proposed application profile.

First use case is given in Figure 6.1 below;

Use Case Name	Creating Learning Object Metadata Instances
Purpose	A related party wants to create descriptions for learning resources by only defining generic elements and major commonly used learning related elements such as subject, competence etc. and they want to ensure searching and harvesting of these learning objects via these descriptions.
Primary Actors	Cataloguer (anyone who is responsible for providing description for learning resources)
Preconditions	• Existence of related application profile, • Existence of learning object description editor.
Flow of Events	• Cataloguer selects the resource to be described, • System retrieves and provides the metadata fields to be filled according to application profile, • Cataloguer provides necessary information by filling metadata fields, • Cataloguer saves the description, • System binds the description into XML or RDF.

Figure 6.1 Use Case I

Second use case is given in Figure 6.2 below;

Use Case Name	Searching Learning Objects
Purpose	Purpose: An individual user or an automated software application wants to apply search operations over a repository that provides learning resources which are described by learning object metadata.
Primary Actors	• Student, • Software Agent (as a Search Medium)
Preconditions	• A learning resource repository, • Described Learning resources in repository.
Flow of Events	• Student starts the agent which can query repository (system), • Student provides search criteria into the agent, • Agent connects repository and submits queries according to given criteria, • Repository scans the learning resource descriptions and returns the matched learning object descriptions, • Agent provides the appropriate results to the student.

Figure 6.2 Use Case II

Third use case is given in Figure 6.3 below;

Use Case Name	Harvesting Learning Objects
Purpose	A teacher uses an automated software application and a learning object description detector tool to harvest and process one or more results of learning object search.
Primary Actors	• Teacher, • Software Agent (e.g. Reasoning Algorithm Applier)
Preconditions	• Teacher already submitted a query, • Search results are already returned, • Teacher has a tool capable of detecting learning resources, • Teacher has an agent which applies reasoning algorithms over learning resource descriptions.
Flow of Events	• Teacher displays the search results, • Teacher selects different learning objects, • Detector software detects the Learning Objects, • Teacher shares these learning object descriptions with her agent via detectors software, • Agent applies several reasoning algorithms to derive some reasons from input such as competence learning object relationship.

Figure 6.3 Use Case III

6.1.3 Metadata Elements

First data types those will be used in the application profile must be defined, before selection and definition of metadata elements. For the sake of simplicity and minimalist approach, only CharacterString and Vocabulary data types going to be used. However as noted before Vocabulary element for Vocabulary data type is an aggregate element and it consists of “source” and “value” elements. In this case Vocabulary is used as simple element; actually it is interpreted as CharacterString

element. Here only difference of Vocabulary type from CharacterString type is its having a defined set of vocabulary as value domain.

Selection of metadata elements follows the same approach that is followed when selecting data types to be used; simplicity and minimalist approach. This comes from the scope and purpose that proposed application profile going to serve for. Only thing that is expected from this application profile is enabling search and harvest of learning resources in the frame of Micrformats. Therefore a learning object do not need to be represented with all its details by this application profile, rather it will facilitate discovery and limited reflection of these objects into web with limited set of elements. Complete IEEE LOM Metadata set could be mapped into Microformats, however this would break the Microformats principles and would cause manageability problems. Therefore complex features can still be accessible by means of using complete IEEE LOM set via RDF or XML, while ensuring simple and important features with a limited set of elements via Microformats. The proposed set is completely derivable from any IEEE LOM instance, therefore it does not arise compatibility or manageability problems.

IEEE LOM and Dublin Core including DC-Ed and DCMI-EMS are used to select Metadata elements. IEEE LOM is a wildly accepted specification by for learning community and Dublin Core is wildly accepted for defining general web resources. The problem that IEEE LOM arises is manageability because of the number of elements (more than 71) it provides. The conventional wisdom is that learning object should be accompanied by metadata whose minimal form would contain the information typically found in the description of a book, journal etc. [60]. There are already some analysis work done for selection of data elements which are wildly used by other application profiles, the important ones are Friesen and Campbell spreadsheets [61, 62] and later work of Carol Jean Godby [60] which is based on Friesen and Campbell spreadsheets. Total of 35 application profile has been investigated during these works and the results showed a close match between Dublin Core elements and mostly used IEEE LOM elements. Therefore the idea that is going to be followed here is mapping Dublin Core (15 elements) elements into IEEE LOM elements and including a few important learning specific elements for application profile need to be proposed. Total of 18 elements has been proposed

below, 7 of them adopted from Dublin Core name space, 9 of them adopted from IEEE LOM name space and 2 of these elements are derived from IEEE LOM over a custom name space.

Identifier

Description: This element provides a way to assign a globally unique label to the learning object.

XML Name Space: http://course.isikun.edu.tr/custom/

XML Binding Representation: <identifierURI>, <IdentifierURN>, <IdentifierDOI>, <IdentifierISBN>, or <IdentifierISSN>

Multiplicity Requirements: This element must appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters. This element is derived from IEEE LOM, however it is serialized for the sake of simplicity, <identifierURI> is mandatory, and others are optional. The serialization done over the following vocabulary derived for <catalog> element of IEEE LOM.

- URI: Universal Resource Identifier,
- URN: Universal Resource Name,
- DOI: Digital Object Identifier,
- ISBN: International Standard Book Numbers,
- ISSN: International Standard Serial Numbers.

Title

Description: This element provides the name of the learning object.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <title>

Multiplicity Requirements: This element must appear 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters.

Language

Description: This element provides the language of the learning object. Language code must be provided according to the ISO 639:1988. Country sub code can be also presented (Langcode - Subcode) according to the ISO 3166-1997. “none” can be provided as a value of language element if the content is not lingual.(e.g.: en-GB)

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <language>

Multiplicity Requirements: This element must appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters.

Description

Description: This element provides a textual description for the learning object in general.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <description>

Multiplicity Requirements: This element must appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters.

Keyword

Description: This element provides keywords describing the learning object.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <keyword>

Multiplicity Requirements: This element shall appear 0 or more time and has SPM of 10.

Data type: This element is CharacterString type and it has SPM of 40 characters.

Classification

Description: This element provides a means of classification by providing a subject a phrase, or a classification code.

XML Name Space: http://course.isikun.edu.tr/custom/

XML Binding Representation: <classification>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters. This element is derived from IEEE LOM; it matches with the <description> sub element of IEEE LOM <classification> element.

Coverage

Description: This element provides the description for time, geography, or region for which learning object belongs to.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <coverage>

Multiplicity Requirements: This element shall appear 0 or more times and has SPM of 2.

Data type: This element is CharacterString type and it has SPM of 1000 characters.

Type

Description: This element provides the kind of the learning object.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <type>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is Vocabulary type and proposed vocabulary is as follows;

- exercise, simulation questionnaire, diagram, figure, graph, index, slide, table, narrative text, exam, experiment, problem statement, self assessment, lecture.

End User

Description: This element annotates the intended end user for which the learning resource purposed.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <endUser>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is Vocabulary type and proposed vocabulary is as follows;

- all, teacher, author, learner, manager.

Context

Description: This element annotates the intended environment for which the learning resource purposed.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <context>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is Vocabulary type and proposed vocabulary is as follows;

- school, higher education, training, other.

Date

Description: This element annotates the learning object of date of any event associated with learning object.

XML Name Space: http://purl.org/dc/elements/1.1/

XML Binding Representation: <dateTime>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is DateTime type.

Creator

Description: Provides the primarily responsible entity for making the resource.

XML Name Space: http://purl.org/dc/elements/1.1/

XML Binding Representation: <creator>

Multiplicity Requirements: This element shall appear 0 or more times and has SPM of 2.

Data type: This element is CharacterString type and it has SPM of 1000 characters. All values shall be presented in vCard format.

Contributor

Description: Provides the entities who contribute to the learning object.

XML Name Space: http://purl.org/dc/elements/1.1/

XML Binding Representation: <Contributor>

Multiplicity Requirements: This element shall appear 0 or more times and has SPM of 2.

Data type: This element is CharacterString type and it has SPM of 1000 characters. All values shall be presented in vCard format.

Publisher

Description: Provides the entities that are responsible of making resource available.

XML Name Space: http://purl.org/dc/elements/1.1/

XML Binding Representation: <publisher>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters. All values shall be presented in vCard format.

Format

Description: Provides the information about the format of the learning object.

XML Name Space: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <format>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 500 characters. All values shall be presented in vCard format. The CharacterString shall be a MIME type (IANA registration or string: “non-digital”).

Rights

Description: Provides the information about rights held in and over the learning object.

XML Name Space: http://purl.org/dc/elements/1.1/

XML Binding Representation: <rights>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters.

Relation

Description: Provides the information about a related resource.

XML Name Space: http://purl.org/dc/elements/1.1/

XML Binding Representation: <relation>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters.

Source

Description: Provides the information about a resource which learning object is derived from.

XML Name Space: http://purl.org/dc/elements/1.1/

XML Binding Representation: <source>

Multiplicity Requirements: This element shall appear 0 or 1 time.

Data type: This element is CharacterString type and it has SPM of 1000 characters

```
1   <?xml version="1.0" encoding="utf-8"?>
    <mLom xmlns:dc="http://purl.org/dc/elements/1.1/"
2   xmlns:lom="http://ltsc.ieee.org/xsd/LOM/"
    xmlns:my=”http://course.isikun.edu.tr/custom/”>
3       <my:identifierURI> http//course.isikun.edu.tr/LO1</my:identifierURI>
4       <lom:title>Learning Object</lom:title>
5       <lom:language>en-GB</lom:language>
6       <lom:description>Textual description of Learning Object</lom:description>
7       <lom:keyword>eLearning</lom:keyword>
8       <lom:keyword>Learning Object</lom:keyword>
9       <my:classification>ADL Scorm Cocepts<my:classification>
10      <lom:coverage>eLearning</lom:coverage>
11      <lom:type>narrative text</lom:type>
12      <lom:endUser>Learner</lom:endUser>
13      <lom:context>School</lom:context>
14      <dc:date>2001-07-30</dc:date>
15      <dc:creator> entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Ahmet
    Author&#13;&#10;END:VCARD</dc:creator>
16      <dc:contributor> entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:
    John Author&#13;&#10;END:VCARD</dc:contributor>
17      <dc:publisher> entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;
    ORG:ISIK Author&#13;&#10;END:VCARD </dc:publisher>
18      <lom:format>text/html</lom:format>
19      <dc:rights>http://www.isikun.edu.tr/termsandconditions.htm</dc:rights>
20      <dc:relation>Part of Scorm 3rd Editon, Content Aggregation Model</dc:relation>
21      <dc:source>Scorm 3rd Edition</dc:source>
22  </mLom>
```

Figure 6.4 Sample XML Binding of Application Profile

Sample XML binding of the proposed application profile is given in Figure 6.4. All elements directly mapped to their associated XML binding representation. Three name spaces have been used. “lom” represents IEEE LOM name space, “dc” represents Dublin Core name space and “my” represents the custom name space for the elements which are adopted from IEEE LOM by changing their original form.

6.2 Microformat Proposal

Proposal of Microformat is quite straight forward in this case, because the base applications profile itself is already simple and flat because there is no aggregate element used in application profile at all.

In Figure 6.5, an example Microformat demonstration is given, for this example “dl” (definition list) HTML element has been used to embed Microformat.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4  <head> <link rel="stylesheet" type="text/css" href="mLOM.css"  />
5  <title>LOM Microformat</title>
6  </head>
7  <body>
8          <dl class="mLom">
9                  <dt>Title</dt>
                     <dd class="title">Sharable Content Object Reference Model - Sequencing
10 and Navigation</dd>
11                 <dt>Language</dt>
12                 <dd><abbr class="language" title="en-GB">British English</abbr></dd>
13                 <dt>Description</dt>
                     <dd class="description">The SCORM SN book describes how SCORM
14 conformant     content may be sequenced to the learner through a set of learner or system-
   initiated navigation events.
                   </dd>
15                 <dt>Creators</dt>
16                 <dd class="creator vcard"><span class="fn">Daren</span></dd>
17                 <dt>Contributors</dt>
18                 <dd class="contributor vcard"><span class="fn">Alice</span></dd>
19                 <dt>Publisher</dt>
20                 <dd class="publisher vcard"><span class="org">Advanced Distributed
   Learning  (ADL)</span></dd>
21                 <dt>Identifier</dt>
22                 <dd class="identifierURI">http://course.isikun.edu.tr/LO2</dd>
23                 <dt>Classification: </dt>
24                 <dd class="classification"> ADL Scorm Cocepts </dd>
25                 <dt>End User: </dt>
26                 <dd class="endUser"> Learner </dd>
27                 <dt>Classification: </dt>
28                 <dd class="context"> School </dd>
29                 <dt>Coverage</dt>
30                 <dd class="coverage">eLearning</dd>
31                 <dt>Keywords</dt>
32                 <dd class="keyword">
33                         <a href="www.example.com/eLearning"
   rel="tag">eLearning</a>,
34                         <a href="www.example.com/LO" rel="tag">Learning Object</a>
35                 </dd>
36                 <dt>Type</dt>
37                 <dd class="type">narrative text</dd>
38                 <dt>Date</dt>
39                 <dd><abbr class="date" title="2001-07-30">30 July 2007</abbr></dd>
40                 <dt>Format</dt>
41                 <dd class="format">txt/html</dd>
42                 <dt>Rights</dt>
43                 <dd><a href='http://www.isikun.edu.tr/termsandconditions.htm'
   class="rights">http://www.isikun.edu.tr/termsandconditions.htm</a></dd>
44                 <dt>Relation</dt>
45                 <dd class="relation">Part of Scorm 3rd Editon, Content Aggregation
   Model</dd>
46                 <dt>Source</dt>
47                 <dd class="source">Scorm 3rd Edition</dd>
48         </dl><body></html>
```

Figure 6.5 Sample Microformat of Proposed Application Profile (LO2)

However note that any other HTML element could be used, proper assignment of class attribute is the only important thing. Therefore any XHTML element can be used as far as appropriate class names are assigned, however it is important to remind that these elements must be semantic elements instead of presentational elements. In this proposal again only title and identifier element is mandatory and the rest of the elements are not forces, shortly the Microformat proposal follows the application profile constraints.

Most of the elements has been directly associated with a “class” attribute. The “creator”, “contributor” and “publisher” elements used together with “vcard” identifier. This is because the domains associated with these elements are based on “vcard” as previously noted in related element definitions. Class design pattern is used for most of the elements, and abbr-design pattern has been used for “date” and “language” elements. Only one elemental Microformat has been used, and this is rel-tag elemental Microformat which is used for “keyword” element.

In Figure 6.6 the view of example code in Figure 6.5 has been demonstrated.

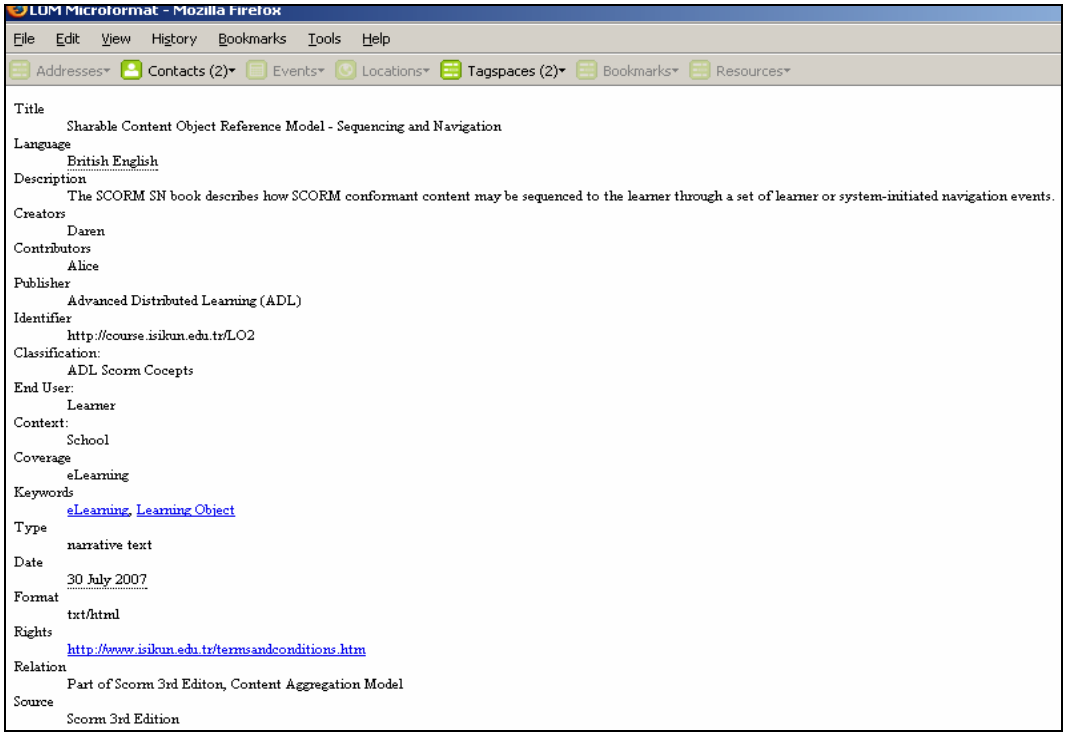


Figure 6.6 Demonstration of Proposed Microformat

It is plain, free of any presentational code, and understandable. The elemental Microformat rel-tag and the information closured by “vcard” are already detected by the Firefox Operator plug-in as.

The code piece in Figure 6.7 is an example CSS file produced for the Microformat example in Figure 6.5. Main HTML elements are directly assigned with a presentational CSS like “dt” and “dl”, for the class names “.class_name” format has been used, so for each application profile element any individual presentational features can be assigned like “.title” in given example CSS code in Figure 6.7.

```
1       dt{font-weight:bold; }
2       dl{
3               padding:.5em; background:#ccc; border:1px solid black;
4               margin-right:2em;
5             width:700px; }
6       .title{ font-style:italic; }
```

Figure 6.7 Sample CSS File

In Figure 6.8 the view of example code in Figure 6.5 has been demonstrated after applying the CSS code given in Figure 6.7.

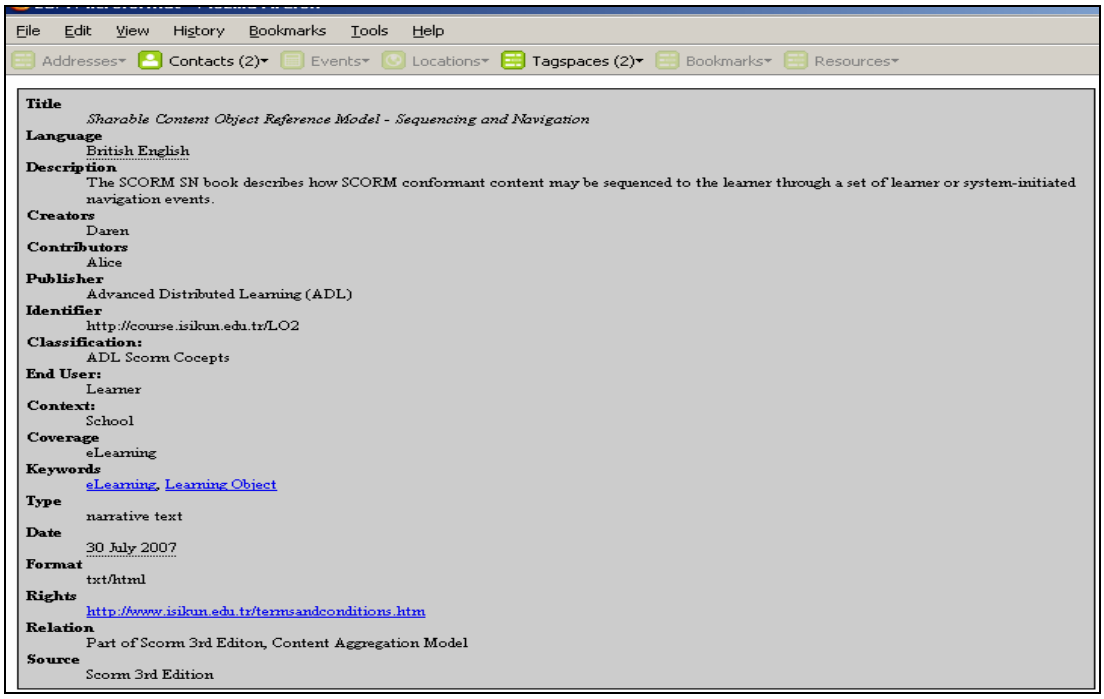


Figure 6.8 Demonstration of Proposed Microformat, CSS Applied

Any shape and look and feel can be created over proposed Microformat by just manipulating CSS code without involving any presentational features inside the HTML of Microformat.

6.3 Case Example: Sematic Search Engine for LOs

Implementation work for the case example can be summarized as follows;

- Preparation of the example Microformat embedded web pages,
- Defining the structure of RDF document,
- Preparation of XSLT transformation file for XHTML (Microformat involving) to RDF conversion,
- Creation of a SQI service,
- Integration of GRDDL API into SQI service,
- Integration of SPARQL API into SQI service,
- Preparation of XSLT transformation file for XML to XHTML(Microformat involving) transformation,
- Setting up Search Engine Client,
- Setting-up Search Engine End-user interface

Three example Web pages has been prepared which involves Microformat embedded information for some example Learning Objects. One of the examples has already been introduced in Table 6.2 which resides in the CD as “mLOm2.htm” under “randPage2” folder which includes Learning Object Two (LO2). Other example files are “mLom1.htm” and “mLom3.htm” which include Learning Object One and Learning Object Three in given order.

The sample RDF document for LO2 in Figure 6.9 provides the basic structure of the RDF form of the application profile.

```
1   <?xml version="1.0" encoding="UTF-8" ?>
    <rdf:RDF
      xml:base="http://localhost/thesisApp/randPage2/mLom2.htm#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2     xmlns:lom="http://ltsc.ieee.org/xsd/LOM/"
      xmlns:htm="http://www.w3.org/1999/xhtml"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:my="http://course.isikun.edu.tr/custom/">
3   <rdf:Description rdf:about="http://localhost/thesisApp/randPage2/mLom2.htm">
4       <lom:title>Sharable Content Object Reference Model - Sequencing and
    Navigation</lom:title>
5       <lom:language>British English</lom:language>
        <lom:description><![CDATA[The SCORM SN book describes how SCORM conformant
6   content may be sequenced to the learner through a set of learner or system-initiated navigation
    events.]]></lom:description>
7       <dc:creator>Daren</dc:creator>
8       <dc:contributor>Alice</dc:contributor>
9       <dc:publisher>Advanced Distributed Learning (ADL)</dc:publisher>
10      <my:identifierURI>http://course.isikun.edu.tr/LO2</my:identifierURI>
11      <my:classification>ADL Scorm Cocepts</my:classification>
12      <lom:endUser>Learner</lom:endUser>
13      <lom:context>School</lom:context>
14      <lom:coverage>eLearning</lom:coverage>
15      <lom:keyword>eLearning</lom:keyword>
16      <lom:keyword>Learning Object</lom:keyword>
17      <lom:type>narrative text</lom:type>
18      <dc:date>2001-07-30</dc:date>
19      <lom:format>txt/html</lom:format>
20      <dc:rights>http://www.isikun.edu.tr/termsandconditions.htm</dc:rights>
21      <dc:relation>Part of Scorm 3rd Editon, Content Aggregation Model</dc:relation>
22      <dc:source>Scorm 3rd Edition</dc:source>
23  </rdf:Description>
24  </rdf:RDF>
```

Figure 6.9 Sample RDF Document of LO2

The Figure 6.10 provides the general graph representation of the proposed application profile. However it is important to note that for elements like “creator”, “contributor” etc. the predicates will also be resource. However for simplicity all are threaded as simple elements here.

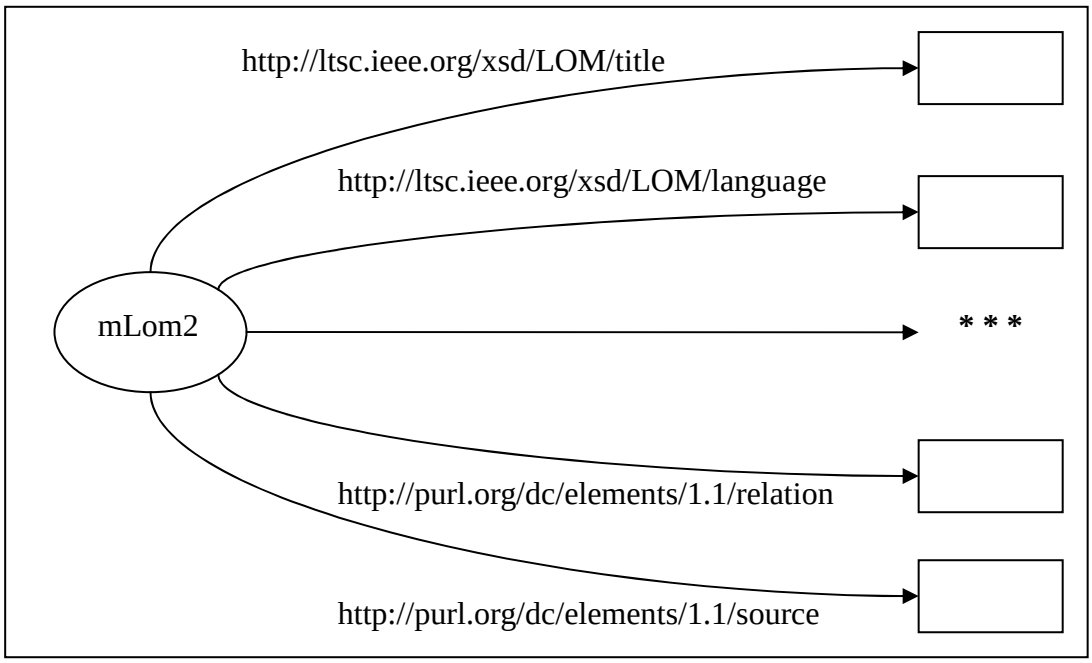


Figure 6.10 Graph Representation of LO2

The code view in Figure 6.11 gives the general structure of XSLT file used for XHTML to RDF conversion. Only first half of the XSLT file has been shown, however rest of the file is repeating sequence of the code at lines 11-13 for each element. Main template locates the “mLom” class name having XHTML elements via “//*[@class='mLom']"” pattern, each match represents a learning object. Then for each application profile element applies the similar matching pattern to retrieve the content.

```
1   <?xml version="1.0" encoding="UTF-8" ?>
    <xsl:stylesheet version="1.0"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
2     xmlns:htm="http://www.w3.org/1999/xhtml"
      xmlns:lom="http://ltsc.ieee.org/xsd/LOM/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:my="http://course.isikun.edu.tr/custom/" >
3   <xsl:template match="/">
4   <xsl:for-each select="//*[@class='mLom']">
5           <rdf:Description>
6                   <xsl:attribute name='about'></xsl:attribute>
7                   <xsl:apply-templates/>
8           </rdf:Description>
9   </xsl:for-each></xsl:template>
10    <xsl:template match="htm:*[@class='identifierURI']">
11      <my:identifierURI><xsl:value-of select="."/></my:identifierURI>
12    </xsl:template>* * *
```

Figure 6.11 View from XSLT File for Microformat to RDF Conversion

The code view in Figure 6.11 gives the general structure of XSLT file used for XHTML to RDF conversion. Only first half of the XSLT file has been shown, however rest of the file is repeating sequence of the code at lines 11-13 for each element.

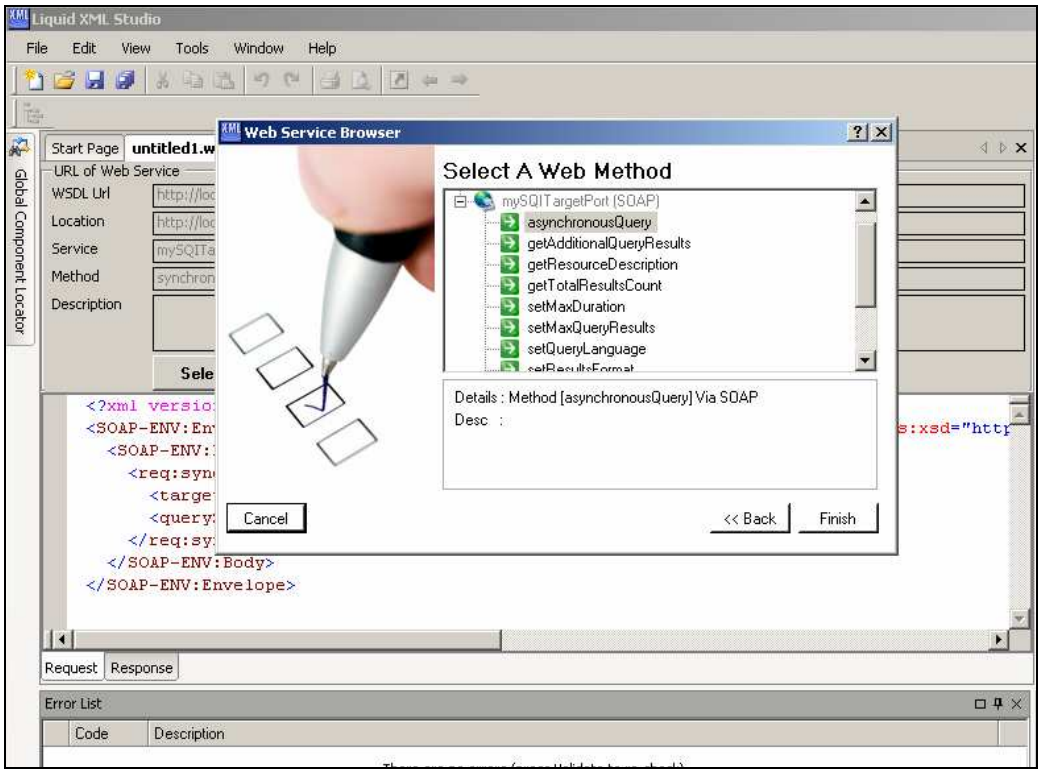
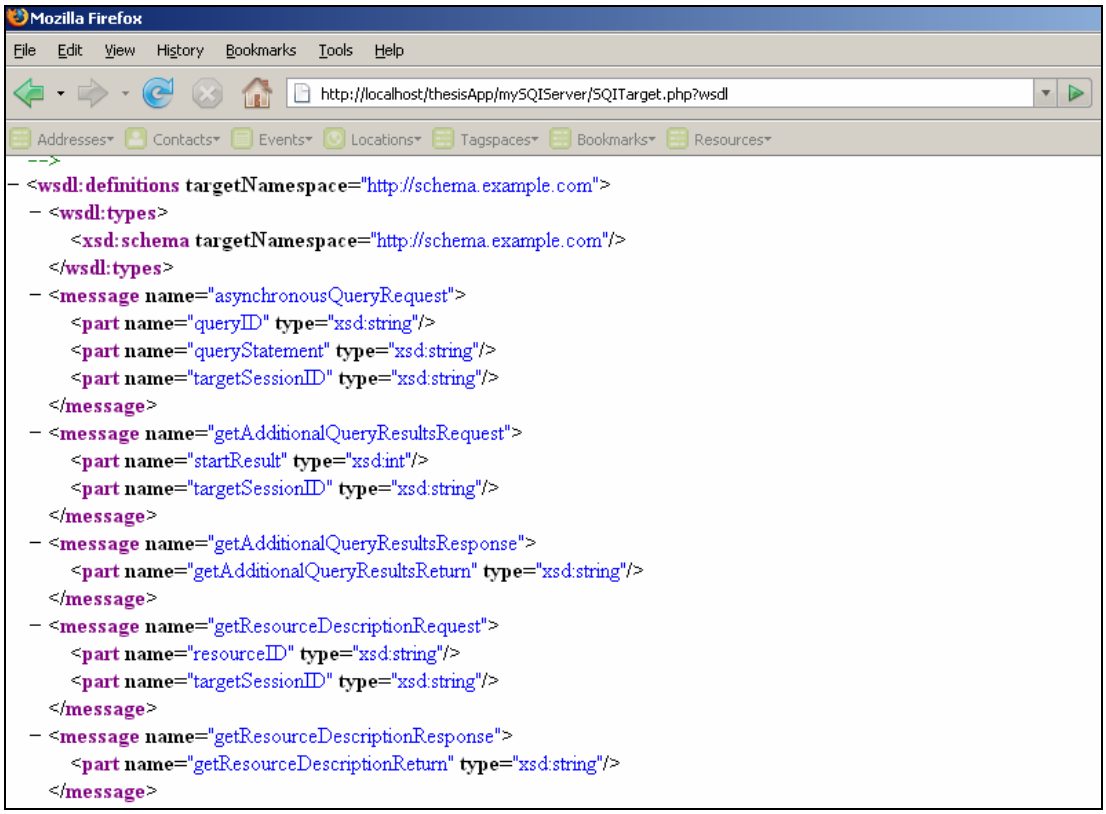


Figure 6.12 Web Service Detected by Liquid Studio

The above phases enabled the basics of proposed application. After these steps a SQI service has been created via PHP according to the specification given for SQI in *Chapter 2*. However this implementation covers the limited features of SQI, for instance session management is not handled at all. However the features provided are enough for the proof of concept. In Figure 6.12, the SQI web service has been detected by the Liquid XML studio application, and list of functions provided by this web service are listed. This application also enables testing of these functions by generating SOAP messages and then forwarding these messages to web service. Response given by the web service is displayed by the Liquid XML studio which makes debugging easier. When the end-point of the SQI web service is called via browser the WSDL document describing the Web service is forwarded to the browser, this has been demonstrated in Figure 6.13.




```
- <wsdl:definitions targetNamespace="http://schema.example.com">
  - <wsdl:types>
      <xsd:schema targetNamespace="http://schema.example.com"/>
    </wsdl:types>
  - <message name="asynchronousQueryRequest">
      <part name="queryID" type="xsd:string"/>
      <part name="queryStatement" type="xsd:string"/>
      <part name="targetSessionID" type="xsd:string"/>
    </message>
  - <message name="getAdditionalQueryResultsRequest">
      <part name="startResult" type="xsd:int"/>
      <part name="targetSessionID" type="xsd:string"/>
    </message>
  - <message name="getAdditionalQueryResultsResponse">
      <part name="getAdditionalQueryResultsReturn" type="xsd:string"/>
    </message>
  - <message name="getResourceDescriptionRequest">
      <part name="resourceID" type="xsd:string"/>
      <part name="targetSessionID" type="xsd:string"/>
    </message>
  - <message name="getResourceDescriptionResponse">
      <part name="getResourceDescriptionReturn" type="xsd:string"/>
    </message>
```

Figure 6.13 Web Service Called via Browser

After setting up the SQI service, it has been enhanced with the GRDDL feature, for this purpose RAP Rdf API [63] for PHP (an open source semantic toolkit) has been integrated with the SQI service. This API provides easy to use functions for GRDDL transformation. Each time a particular function in SQI target is called, SQI target first executes GRDDL procedures of RAP over defined XHTML pages and RDF transformation occurs. Extracted RDF data of web pages are stored in the SQI target physical space as RDF files. Each source references the XSLT transformation page provided in Figure 6.11, RAP API detects these references and applies the referenced transformations over these pages.

The existence of RDF files means that SQI target has sources that it can run queries over, via SPARQL; therefore SQI target has been also enhanced with SPARQL by using RAP Rdf API again. The SQI target modified so it takes the query as a parameter of the related function instead of taking query parameters and substituting these values inside a predefined query. The query results are returned as an associated array by the API therefore before forwarding the results back to the client,

results are transformed into their respective XML binding in the format provided in Figure 6.4.

After having an operational SQI target which successfully executes forwarded queries, another XSLT file has been created which processes XML response into Microformat embedded XHTML again. This transformation file is used by client and creates the complete result structure that will be displayed to the end-user. Client procedure is pretty simple, it only receives the query parameters from end-user interface, and after that prepares necessary query and forwards this query to SQI target. When SQI target sends back the related result, client applies necessary transformation via XSLT document and forwards the processes result for direct display of end-user interface.

In Figure 6.14 result of search for keyword “elearning” has been displayed, the end-user interface has been created by using mainly AJAX, JavaScript and CSS.

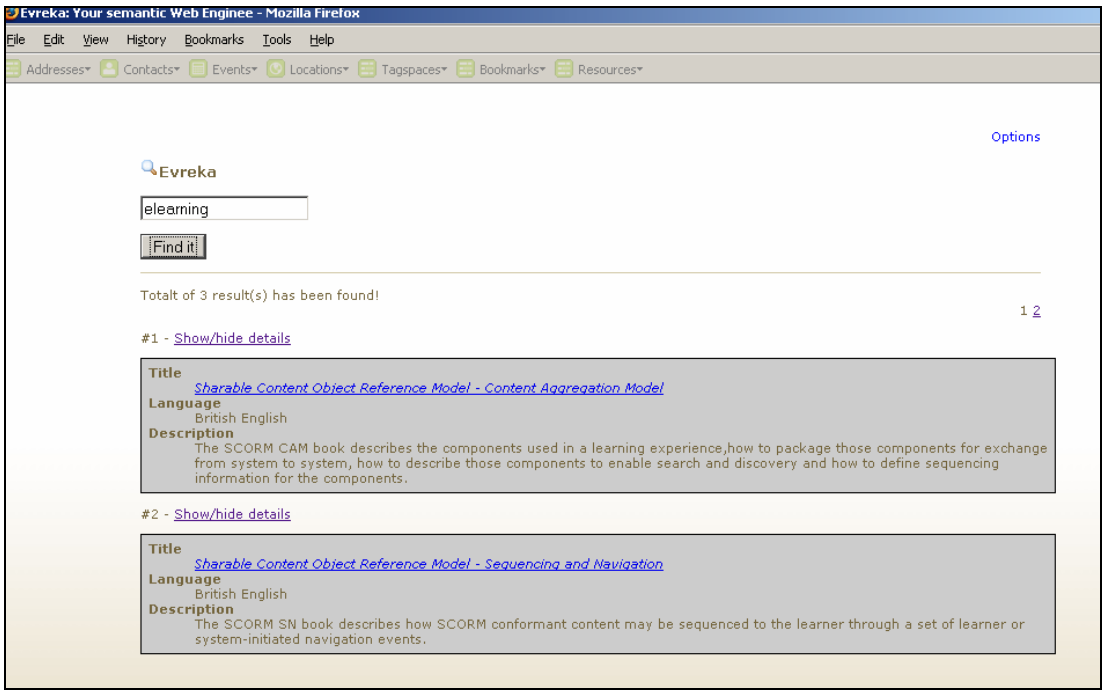


Figure 6.14 End-user Interface Displaying Results of a Query

For your information; JavaScript is an interpreted language for use of HTML and totally runs on client’s browser and enables dynamic behaviour of the page. AJAX (Asynchronous JavaScript and XML) is a technique for communication of the client

(browser) and server behind the scene silently, this enables web applications to behave like desktop applications.

Prototype API [64] has been used for providing easy-to-use AJAX functionalities; however it is important to note that an API is not necessary for using AJAX it is already supported by most of the browsers; however each browser supports different structure and procedures even though they resemble each other a lot. Handling all these differences by yourself is big burden, therefore this APIs rescues you from this burden and enables your code to work in any browser while providing more usable and easy-to-use functions (via DOM manipulation features). The query request, transitions between pages, hiding and showing of results or transition between menus are all handled via AJAX and JavaScript. Therefore most of the work has been done in the client side by neither overloading the client application server nor SQI application server. This is already the idea behind AJAX; only loading necessary information and placing or replacing it on the browser via DOM manipulation without reloading the whole page again and again.

In Figure 6.15, the options panel is displayed.

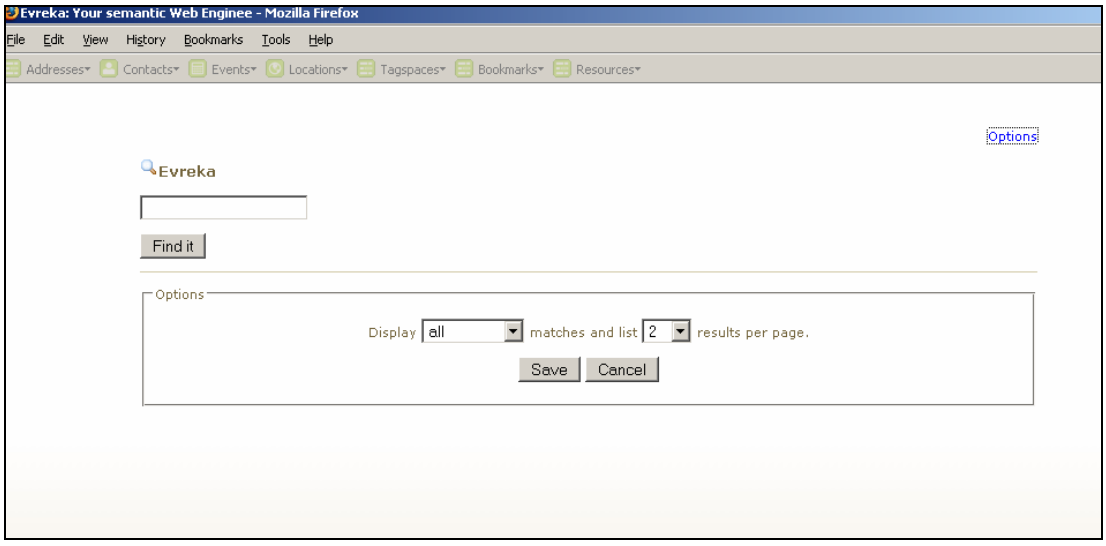


Figure 6.15 End-user Interface Displaying Options Panel

This panel is pretty simple and allows user to choose whether she wants her keywords to have an exact match or like match, besides enables user to choose the number of results that she want to be displayed in each page.

This basic SQI Service and simple Search Engine provides a detailed proof of the concept.

Chapter 7
Evaluation of Model and Application

So far a learning object harvesting model has been proposed and a sample application has been developed based on this model. The model is based on a learning object metadata application profile proposal derived from IEEE LOM, Dublin Core and Scorm, and a Microformat proposal employing this application profile. Several semantic web technologies have been used to realize this harvesting model, these technologies enabled learning object metadata to be embedded into XHTML pages, and to be harvested from XHTML pages. For the proof of concept, a search client and server employing the harvesting model and SQI target have been set up over these technologies and the model. Proposed model and the application can be evaluated form different perspectives.

It is already denoted in definition of e-Learning that it consists of actions delivering, manipulating, and managing etc. e-Learning resources, outcomes, and activities. Learning resources provides the basic ground of the e-Learning therefore success of e-Learning environments, systems etc. strictly based on how they deal with learning objects. Therefore various e-Learning object metadata standards and application profiles have been investigated; the obvious problem was the manageability of these standards and profiles because of the number of elements provided. Although most of these application profiles aimed to provide the most common set of elements apart from best practices on use of these elements, a panacea for every related party did not arrive. The application profile proposed by this thesis employs the set of elements from IEEE LOM which are mapped from Dublin Core generally, and this application profile has a quite general nature apart from a few elements directly related with learning aspects. This is mainly based on the fact that most of the LOM

elements are not used by most of the application profiles, in order to come up with a common solution a common set tried to be identified.

Table 7.1 LOM Element Use [60]

The most commonly recommended LOM elements			*The least commonly recommended LOM elements*		
LOM Element	Count	Dublin Core equivalent	LOM Element	Count	Dublin Core equivalent
General	28		**General**		
Title	33	Title	Structure	5	
Description	34	Description	Coverage	6	Coverage
Identifier	31	Identifier			
Language	25	Language	**Technical**		
Keyword	26	Subject	OrComposite	2	
			Name	6	
Lifecycle	29		MaximumVersion	5	
Contribute	29	Contributor	MinimumVersion	4	
Entity	27	or			
Role	23	Publisher	OtherPlatformRequirements	8	
Date	24		InstallationRemarks	7	
		Date	Duration	6	
Meta-metadata					
Medatascheme	22		**Educational**		
			SemanticDensity	4	
			Difficulty	8	
			Language	6	
Technical					
Format	25		**Relation**		Relation
Location	22	Format	Kind	8	
		Identifier	Description	4	
			Catalog	4	
			Entry	4	
Rights	24	Rights	**Annotation**	8	
Cost	24				
Copyright AndOtherRestrictions	25				
Classification	22	Subject			
Purpose	25				
	Total: 35			Total: 35	
	Average: 26.3			Average: 5.6	

The Table 7.2 demonstrates use of the LOM elements among 35 different Application profiles, selection of elements for proposed application profile considers this usage statistics. This set of commonly used elements is enhanced with some learning related elements and it can be evaluated against following criteria. The main disadvantage is lack of ontological expression. IEEE LOM classification element actually provides capability to express any ontological relation; however it increases the complexity of the application profile, therefore proposal of such element requires an in-depth analysis of learning object examples. Ontological expression power will increase accessibility and interpretation of learning objects while increasing utilization.

Table 7.2 Evaluation of Learning Object Metadata Proposal

Re-usability	Proposed application profile is lowest common denominator for most of the application profiles, high re-usability.
Interoperability	Proposed application profile has been derived from IEEE LOM, Dublin Core and SCORM which are the most commonly adopted standards, this gives high interoperability.
Manageability	The size of proposed set of elements is quite light 18, this provides high manageability.
Accessibility	Higher size for elements in the application profile might be considered as a plus for accessibility, however as most of these elements are out of use in many application profiles it can be claimed that the trade of is not that much, besides the minimal size ensures the appropriateness of the results. A disadvantage comes from the lack of representing complex (ontological) relations between learning objects.
Durability	Interoperability and usability has already been ensured therefore, durability is considered as high.
Scalability	This application profile is thought to provide minimal set of elements for e-Learning domain; however by nature application profile itself provides scalability.
Affordability	This application profile can be considered as the core of all other profiles therefore affordability is ensured because it is easy to switch between alternatives.

The main aim of thesis is to enable harvesting of learning objects embedded in XHTML pages, therefore Microformats proposal is the second main stone of this thesis. The Microformats approach has been already compared with its alternatives (eRDF, RDFa) previous chapters. However apart from their advantages and disadvantages with respect to each other, the main idea of embedding semantic data into XHTML pages greatly contributes into the e-Learning, it enables the biggest source of information, WWW, to be harvested by every single learner. Microformats can be thought as the second layer of this model therefore it has been evaluated according to the same criteria below;

Table 7.3 Evaluation of Microformats Proposal

Re-usability	Enables re-usability of learning content on World Wide Web.
Interoperability	Proposed Microformat employs an application profile derived from wildly accepted standards, therefore enables interoperability.
Manageability	Managing information in XHTML pages is quite straight forward, enabling technologies do exists, and besides the size of application profile that Microformat proposal is based on is quite minimal.
Accessibility	Learning Objects are accessible form any type of device and system as they are embedded into presentation free, POSH pages.
Durability	Interoperability and usability has already been ensured therefore, durability is considered as high.
Scalability	Although application profile itself provides scalability, the nature of Microformats is a barrier. They do not allow name spaces plus they are defined once and not subject to change often.
Affordability	The base application profile can be considered as the core of all other profiles therefore affordability is ensured, because the information presented by Microformat does not go beyond the standards.

Main problem of the Microformats proposal does not come from the proposal but rather from comes from the Microformat idea itself. During this thesis RDF has been used, however Microformats lacks cooperating the features provided by the RDF, those are;

- Open-ended design,
- Ontological expression power,
- Extendibility,
- No predefined format.

eRDF and RDFa can be considered as alternatives to the Microformat approach, however they do not have real life examples and acceptance yet. As already noted previously eRDF is not capable of expressing every RDF structure, however RDFa has this power but it uses some XHTML elements which are not in use yet. A switch from Microformats to RDFa is possible when it is mature enough, however because of the acceptance and ease-of-use of Microformats, it will keep leading till RDFa arrives.

The technological infrastructure used in sample application is subject to change depending on the case; this already the one of the ideas behind, the proposed model provides high level of independency. Therefore it is not bound to SQI or SOAP, alternatives can be used. However evaluation of sample application can be done as follows by indicating the plusses and minuses below;

Plusses;

- Application demonstrates an important view where the proposed model might be applied primarily
- Application stands as the basic proof of concept of the proposed model,
- Application stands as a compact demonstration of tool and data interoperability,
- Application provides a demonstration of the importance of standard compliance,

Minuses;

- SQI infrastructure is not totally implemented,
- Some constrains of the model omitted for the sake of simple design,
- Search client does not employ any ranking system or ontological inference.

Chapter 8
Conclusion and Recommendations for Future Work

8.1 Overview

The purpose of this thesis was to come up with a web based interoperable Learning Object harvesting model and its application in order to exploit use of huge amount of information contained by World Wide Web. Two main challenges have been identified which are interoperability and semantics. Various related technologies and standards have been investigated for this purpose.

Interoperability has been considered in two folds during this thesis, first one is tool interoperability and the second one is data interoperability, actually in this case rather meta-data interoperability. Tool interoperability is known as; different applications being able to communicate each other. The most known solution for World Wide Web (WWW) is web services. Therefore various XML based technologies and standards have been investigated to come-up with a proper solution for e-Learning domain. SAOP and SQI (which is based on SOAP) have been used as the key elements of tool interoperability. The second part of interoperability is called as data interoperability; it refers to different applications being able to use and share same data. In general this has been considered to be solved via investigating wildly known Learning Object metadata standards and their possible bindings into XML, RDF and especially into XHTML as Microformats. In this sense data interoperability moved into a broader concept which is usually called semantic interoperability; it means applications are not only capable of using same data but also capable of understanding, linking, and interpreting the data

.

The results of these investigations resulted in a light weight application profile proposal for Learning Object Metadata (derived from IEEE LOM, Dublin Core and ADL Scorm), creation of related XML, RDF bindings, and Microformat proposal. After all, a case application has been introduced by providing Microformat embedded sample XHTML pages, a SQI web service which employs SPARQL query language for RDF documents extracted from these XHTML pages, a client web application which acts as a search engine by employing the SQI service, besides both applications have been donated with related XSLT files for GRDDL transformations.

8.2 Recommedations for Future Work

Once the application profile defined, the properly formatted metadata might reside or move in World Wide Web in any format such as XML, RDF etc. The example application provided is a good demonstration. In example application data moves from XHTML format to RDF, RDF to XML and XML to XHTML as demonstrated in Figure 8.1. Therefore it will not be wrong to claim that comprise on the metadata standard is much more important than how the metadata is stored physically, because once agreed on a common standard it is not really a big deal to transform data between these physical structures. Of course this is still a burden but not as much as the burden of not having a common metadata standard.

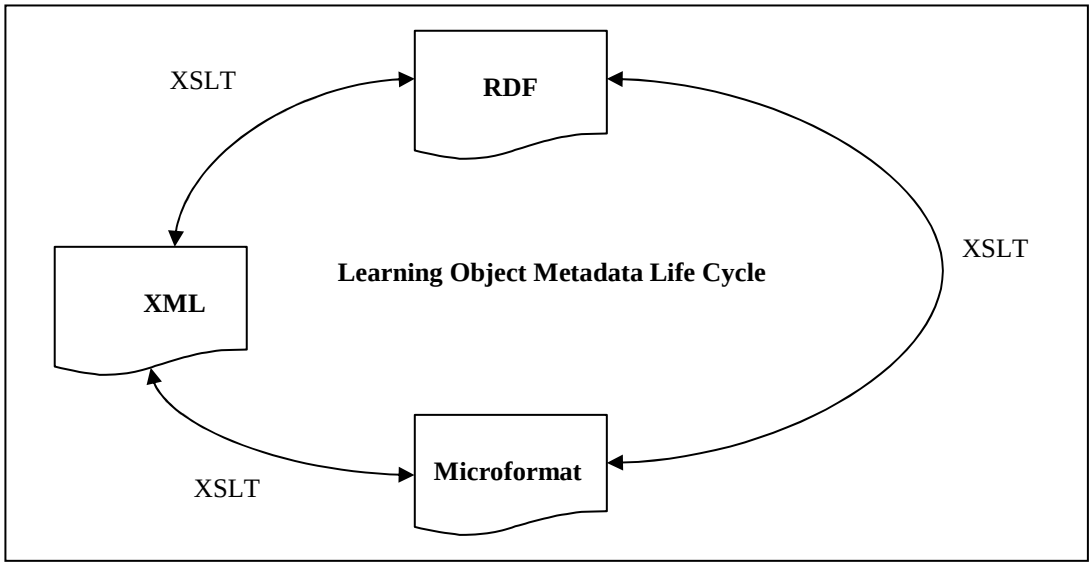


Figure 8.1 Learning Object Metadata Lifecycle

The importance of the XML, RDF, or Microformats is not being ignored by emphasizing the importance of a common metadata standard. A common metadata standard builds a solid ground that these technologies can be exploited by e-Learning domain. RDF provides great utilization of semantic web by itself and ontology languages built upon it such as OWL. Their being able to define resources and complex relationships among them makes the World Wide Web a highly connected network of information from its arbitrarily distributed and disconnected structure. These advancements reach to an upper level of utilization by the existence of Microformats, eRDF, RDFa. Despite their limitations, they save the current nature of World Wide Web from user perspective however provides any information to be understandable by computers without duplicating any information for presentation or storage. XML never loses its importance; it is the mother of all these technologies and still the most efficient way for data transportation and for other uses.

It has been proven that a complete working model can be provided for Learning Object Harvesting, so what can be the next? Actually this thesis just moves us to just to the beginning and the future opportunities are endless, there are a lot to do. Imagine such use of this harvesting model; applying a reasoning algorithm over the data that is queried over RDF documents via SQI target to decide on a competence-activity relation which questions; which activity(s) is used for which competence(s). Such use of the model requires the intervention of ontologies, for this purpose the first step is enhancing our solid base which is metadata application profile proposed. In the context of this thesis it is tried to be kept as simple as possible just for enough to enable search and retrieval of learning objects. However for such ontological needs, each element need to be considered seriously, because they constitute the hot points where ontological relations come up. Classification element is a good example, in original IEEE LOM standard it has much more power of expression from hierarchical position of resources to their competence levels etc. Therefore it is really important from ontological point of view which can not be reflected by just providing it in the form of description. Both for the completeness and the ontological expressiveness of the application profile, many real life learning objects and their associated metadata records need to be analysed, however note that even this task itself is really hard to deal with. Definition of the object changes from application to

application while the metadata elements sets are changing deepening on the location, application etc.

Successfully completing these advancements need to be followed by strong reasoning and data mining algorithms and techniques not just for harvesting but also for analysing possibly huge amount of learning patterns and activities which will be mostly dependent on purpose of the applications. These applications mentioned might be agents harvesting the World Wide Web by roaming from page to page. Those agents are not expected to serve just for traditional PCs but rather for any kind device which has access to internet such as PDAs, cell phones etc. In this way learning will be embedded into real life.

It is already obvious that the improvements mentioned are actually global and not limited with the e-Learning domain. It also should be noted that Word Wide Web is not specifically for any group or level of people, it is open to everybody, and therefore simplicity is the must of this world. Currently anybody using World Wide Web can freely make contribution via Web2.0 tools; therefore active existence of any user without having any complex skills already became indispensable right of everybody. Therefore “Human first, machine second!” principle of Microformats actually should be considered globally for World Wide Web.

Regarding the sample application, obviously many enhancements are possible. Technologies like AJAX provides endless opportunities to increase usability and efficiency of such web applications however apart from these usability dependent enhancements, several important enhancements are possible such as embedding a ranking strategy, and employing ontological inference feature which is dependent on the model proposed indeed. Fully implementation of SQI and providing the missing constraints of the proposed model might be considered the primary future work with respect to sample application.

References

[1] Mayes, T., Freitas, S., "Review of E-Learning Theories, Frameworks and Models", *JISC E-Learning Models Study Report, Joint Information Systems Committee*, 2004

[2] Allen, E. and Seaman, J., "Growing by Degrees: OnLine Education in the United States", *Sloan Consortium*, 2005

[3] Soylu, A., Karahasan, O., Kuru, S., "Çok Uluslu, İşbirlikçi, Sosyal e-Öğrenme: iCamp Örneği", *AB2007 Conference*, Duplupinar University, 31 Jan. – 2 Feb. 2007

[4] Kuru, S., Nawojczyk M., Niglas, K., Butkeviciene E., Soylu A., "Facilitating Cross-border Self-directed Collaborative Learning: The iCamp Case", *EDEN 2007 Annual Conference*, Naples, Italy, 13-16 June 2007

[5] Kieslinger, B., Fiedler, S., Wild, F., Sobernig, S. "iCamp: The Educational Web for Higher Education in an Enlarged Europe", *eChallenges e-2006*, Barcelona, Spain, October 25-27, 2006.

[6] Wade, V., Ashman, H., "Evolving the Infrastructure for Technology-Enhanced Distance Learning", *Internet Computing*, IEEE, Volume 11, Issue 3, P 16-18, 2007

[7] MASIE Centre e-Learning Consortium, "Making Sense of E-learning Standards and Specifications: A decision Makers Guide to their Adoption 2nd Edition", November 2003

[8] Collier, G., Sun Microsystems,"e-Learning Application Infrastructure", *Whitepaper*, 2002

[9] "World Wide Web Consortium", http://www.w3c.com, Retrieved on March 2008

[10] "W3 Schools Online Web Tutorials", http://www.w3schools.com, Retrieved on March 2008

[11] "Extensible Markup Language (XML)", http://www.w3.org/XML/, Retrieved on March 2008

[12] “Introduction to XML”, http://www.w3schools.com/xml/xml_whatis.asp, Retrieved on March 2008

[13] ”Introduction to XML Schema”, http://www.w3schools.com/schema/schema_intro.asp, Retrieved on March 2008

[14] “XHTML 1.0: The Extensible Hypertext Markup Language (Second Edition)”, http://www.w3.org/TR/xhtml1/, Retrieved on March 2008

[15] “XHTML Why?”, http://www.w3schools.com/xhtml/xhtml_why.asp, Retrieved on March 2008

[16] “The Extensible Style Sheet Family (XSL)”, http://www.w3.org/Style/XSL/, Retrieved on March 2008

[17] http://www.w3schools.com/xpath/default.asp, Retrieved on March 2008

[18] “XPath Tutorial”, http://www.w3.org/TR/xslt, Retrieved on March 2008

[19] “Web Services Glossary”, http://www.w3.org/TR/ws-gloss/, Retrieved on March 2008

[20] Simon, B., Massart, D., Assche, V. F., Ternier, S., Duval, E., Branter, S., Olmedilla, D., Miklos, Z., “A Simple Query Interface for Interoperable Learning Resources”, *Proceedings of the 1st Workshop on Interoperability of Web-based Educational Systems*, Chiba, Japan, May, 2005

[21] Simon, B., Massart, D., Assche, V. F., Ternier S., Duval, E., “Simple Query Interface Specification”, 2005

[22] Iannella, R., “An Idiot's Guide to the Resource Description Framework”, The New Review of Information Networking, Vol 4, 1998

[23] “RDF Rules”, http://www.w3schools.com/rdf/rdf_rules.asp, Retrieved on March 2008

[24] “RDF Primer, 2004”, http://www.w3.org/TR/REC-rdf-syntax/, Retrieved on March 2008

[25] “SPARQL Query Language for RDF, 2008”, http://www.w3.org/TR/rdf-sparql-query/, Retrieved on March 2008

[26] “SPARQL Tutorial”, http://jena.sourceforge.net/ARQ/Tutorial/, Retrieved on March 2008

[27] “Gleaning Resource Descriptions from Dialects of Languages (GRDDL)”, http://www.w3.org/2004/01/rdxh/spec, Retrieved on March 2008

[28] Lee, T., Handler, J., Lassila O., “The Semantic Web”, *Scientific American*, May17, 2001

[29] Wilson, M., Matthews, B., “The Semantic Web: Prospects and Challenges”, *Databases and Information Systems*, 2006 7th International Baltic Conference, pages 26-29, July 2006

[30] ”RDFa Primer”, http://www.w3.org/TR/xhtml-rdfa-primer/#id85078, March 2008

[31] “Rdf in HTML:Embedded RDF”, http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml, Retrieved on March 2008

[32] “Clean-up your HTML pages with Tidy”, http://www.w3.org/People/Raggett/tidy/, Retrieved on March 2008

[33] Simpson, J., “Microformats vs. RDF: How Microformats relate to the Semantic Web”, http://www.semanticfocus.com/blog/entry/title/microformats-vs-rdf-how-microformats-relate-to-the-semantic-web/, October, 2007

[34] Khare, R., Çelik, T., “Microformats: A pragmatic path to the Semantic Web”, *15th International World Wide Web Conference*, 2006

[35] Graf, A., “RDFa vs. Microformats”, *DERI Technical Report*, April, 2007

[36] “What is the Next Big thing on the Web? It may be a Small, Simple Thing – Microformats.”,http://knowledge.wharton.upenn.edu/article.cfm?articleid=1247,Knowledge@wharton, 27 July 2005

[37] “About Microformats”, http://microformats.org/about/, Retrieved on March 2008

[38] “Operator:Firefox Add-on”, https://addons.mozilla.org/en-US/firefox/addon/4106, Retrieved on March 2008

[39] “Getting back to POSH”, http://factoryjoe.com/blog/2007/04/21/getting-back-to-posh-plain-ol-semantic-html/, Retrieved on March 2008

[40] “Microformats”, http://microformats.org/wiki/Main_Page, Retrieved on March 2008

[41] “hCard Creator”, http://microformats.org/code/hcard/creator, Retrieved on March 2008

[42] “hCard”, http://microformats.org/wiki/hcard, Retrieved on March 2008

[43] “About IMS Global Learning Consortium”, http://www.imsglobal.org/aboutims.html, Retrieved on April 2008

[44] “About ADL”, http://www.adlnet.gov/about/index.aspx, Retrieved on April 2008

[45] “About IEEE Learning Technologies Standard Committee”, http://ieeeltsc.org/, Retrieved on April 2008

[46] Etesse, C., “Leading the Way on Standards-Based e-Learning”, November 2004

[47] Seth, R., “Learning Object Metadata and its Application”, *DRTC Conference on ICT for Facilitating Digital Learning Environment*, Bangalore, India, 11-13 Jan 2006

[48] Sampson, D., “The evaluation of Educational Metadata: From Standards to Educational Metadata”, *IEEE International Conference on Advanced Learning Technologies ICALT’04*, 2004

[49] Duval, E., Smith, N., Coillie, M.,”Application Profiles for Learning”, *IEEE International Conference on Advanced Learning Technologies ICALT’06*, 2006

[50] Smith, N., Van Coillie, M. and Duval, E. “Guidelines and support for building Application profiles in e-Learning”, *CEN/ISSS WS/LT Learning Technologies Workshop CWA*, 1-26, 2005

[51] “About IEEE Standard for Learning Object Metadata”, http://ltsc.ieee.org/wg12/par1484-12-1.html, Retrieved on April 2008

[52] “IEEE Standard for Learning Object Metadata”, IEEE Computer Society, IEEE Std. 1484.12.1 – 2002, 6th September 2007

[53] “IEEE Standard for Learning Technology—Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata”, IEEE Computer Society, IEEE Std. 1484.12.3 – 2005, 10th November 2005

[54] “About Scorm 2004 3rd Edition”, http://www.adlnet.gov/scorm/index.aspx, Retrieved on April 2008

[55] “Scorm 2004 3rd Edition, Sharable Content Reference Model-Content Aggregation Model”, Advanced Distributed Learning, 2006

[56] “IMS Metadata Best Practice Guide for IEEE 1482-12-1-2002 Standard for Learning Object Metadata”, Version 1.3, Final Specification, IMS GLC, 2006

[57] Friesen, N., Fisher, S., Roberts, A., “CanCore Guidelines for the Implementation of Learning Object Metadata”, Version 2.0, CanCore, 2004

[58] “The DCMI Education Metadata Set”, http://www.schemas-forum.org/registry/schemas/DCMI-Education/index.html, Retrieved on April 2008

[59] “Dublin Core Metadata Element Set, Version 1.1”, http://dublincore.org/documents/dces/, Retrieved on April 2008

[60] Godby, C. J., “What do Application Profiles Reveal about the Learning Object Metadata Standard?”, ARRIADNE issue 41, October 2004

[61] Friesen, N.. “Survey of LOM Implementations”, CanCore, September 2003

[62] Campbell, M. L., “UK LOM Core Update", PowerPoint presentation, CETIS. September 2003

[63] “RAP Rdf API”, Freie Universität Berlin, 2002

[64] ”Prototype JavaScript Framework: Easy Ajax and Dom Manipulation for Dynamic Web Applications”, http://www.prototypejs.org/, Retrived on April 2008

Appendix A: CD Containing Sample Application and Tools

All developed applications and APIs are included in the CD.

Curriculum Vitae

Publications:

[1] Wild, F., Sigurðarson, S., Sobernig, S., Stahl, C., Soylu, A., Rebas, V., Górka D., Danielewska-Tuecka, A., Tapiador, A., **"**An Interoperability Infrastructure for Distributed Feed Networks**",** *Proceedings of the 1st International Workshop on Collaborative Open Environments for Project-Centered Learning*, Crete, Greece, September, 2007

[2] Kuru, S., Nawojczyk, M., Niglas, K., Butkeviciene, E., Soylu, A., "Facilitating Cross-border Self-directed Collaborative Learning: The iCamp Case", EDEN 2007 Annual Conference, Naples, Italy, 13-16 June 2007

[3] Soylu A., Karahasan O., Kuru S., "Multi Cultural, Colloborative, Social E-Learning: iCamp Case**"**, *AB2007 Conference*, Kütahya, Turkey, 31 January – 2 February 2007

[4] Wild, F., Sigurðarson, S., Sobernig, S., Stahl, C., Soylu, A., Rebas, V., Górka, D., Danielewska-Tuecka, A., Tapiador, A., iCamp Deliverable 3.3**,** "An Interoperability Infrastructure for Distributed Feed Networks"**,** July 2007 (iCamp Project funded under FP6)

[5] Väljataga, T., Siegas, M., Soylu, A., Afonin, A., Wild, F., Sobernig, S., Fiedler, S., "iCamp Building Blocks, Version 2", August 2007 (iCamp Project

funded under FP6)

[6] Kolmos, A., Kuru, S., Hansen, H., Eskil, T., Podesta, L., Fink, F., Graaff, E., Wolff, J. U., Soylu, A., “Problem Based Learnig”, TREE – Teaching and Research in Engineering in Europe, August 2007