Usability of Semantic Web for Enhancing Digital Living Experience

[1]Nirmalya Roy, [2]Kevin Brooks and [1]Christine Julien

[1]*The Center for Excellence in Distributed Global Environments, The University of Texas at Austin*

[2]*Experience Designing & Prototyping Lab, HCI, Motorola Labs, Lowell, Massachusetts*

TR-UTEDGE-2010-012

Usability of Semantic Web for Enhancing Digital Living Experience

[1]Nirmalya Roy, [2]Kevin Brooks and [1]Christine Julien
[1]The Center For Excellence in Distributed Global Environments, The University of Texas at Austin
Email: {nirmalya.roy, c.julien}@mail.utexas.edu
[2]Experience Designing & Prototyping Lab, HCI, Motorola Labs, Lowell, Massachusetts, USA-01851
Email: kevin.brooks@motorola.com

***Abstract*—The number of different types of devices on the home network is expanding rapidly. While this explosion of innovation provides compelling new devices to consumers, there are challenges ensuring compatibility among these devices and providing a comprehensive user interface that supports consumers managing their digital content across several devices. The Digital Living Network Alliance (DLNA) has specified an architecture that enables interoperability between the various devices and allows a user to enjoy the desired content across several devices. To complement DLNA, we investigate ontological representation from semantic web technology to model the interaction between multiple home devices and to provide an enriched and more comprehensive metadata based multimedia search. This removes the user burden of searching each device individually. Development of a prototype incorporating these ideas has begun, using a well known semantic web toolkit Jena.**

***Keywords*-Semantic web, DLNA, multiple device interaction design, ontology, RDF, OWL, user interface visualization**

I. Introduction

The home network is rapidly increasing in complexity: every year brings an expanding array of connected consumer electronics devices, intended for deployment on the home network [6]. While this rapid churn delivers innovative devices to the consumer, it also brings a number of problems, particularly: (i) how to achieve a consistent, powerful and engaging user experience among the expanding array of connected or related devices, and (ii) how to achieve a unified point of control for this expanding array of devices. Towards this end, the Digital Living Network Alliance (DLNA) [1] is defining a framework that provides basic network, service and digital media interoperability for forthcoming CE's applications. The home, however, is a deeply heterogeneous environment, combining devices from multiple vendors, from different hardware generations, added to the home over time. In such a setting, problems arise of how to achieve compatibility while allowing evolution, and how to achieve a consistent and unified user experience. To alleviate this problem we have taken the initiative to model the interaction between multiple home devices using Semantic Web [7] and web ontology languages (OWL/RDF).

The rest of the document is organized as follows: Section 2 gives a brief overview of multiple device task representation using semantic web and the web ontology language OWL. In Section 3 we discuss the rationale behind our Semantic Web approach to build a new architecture to enhance digital living experience and its use case scenario. Section 4 describes our preliminary work on prototyping. A dynamic user interface is outlined in Section 5. Section 6 summarizes and concludes the paper.

This work was done while the author N. Roy was an intern in Experience Designing & Prototyping group at Motorola Labs.

II. Multiple Device Task and its Representation

In our initial consideration we include the following devices Cable set-top boxes (STB), IP STB, Mobile Phone, PDA, Laptop, Desktop PC, Stereo Receiver, TV Monitor, Camera, Refrigerator, Washer/dryer, Door bell, Air conditioner, Lighting, Door lock and Car. With each we identified their different tasks as shown in Fig. 1 and modeled their interaction using ontological technique from semantic web. We use Resource Description Framework (RDF), a low level building block from semantic web to represent simple concept taxonomies and ontologies for associated data. The RDF metadata model is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, called triples. The subject denotes the resource, and the predicate denotes properties or aspects of the resource and expresses a relationship between the subject and the object. For example, here we have considered a mobile phone and represented its different tasks in terms of RDF as shown in Table I.

TABLE I
Representation of different task of a mobile phone using OWL/RDF

Subject (Resource)	Predicate (Property)	Object (Property Value)
Device:mobile phone	hasMake	PhoneCall
Device:mobile phone	hasReceive	PhoneCall
Device:mobile phone	hasMessaging	SMS/MMS
Device:mobile phone	hasTake	Pictures/Video
Device:mobile phone	hasStore	Pictures/Video
Device:mobile phone	hasDisplay	Pictures/Video
Device:mobile phone	hasEdit	Pictures
Device:mobile phone	hasStore	Audio/Video
Device:mobile phone	hasTransfer	Audio/Video
Device:mobile phone	hasRecord	Audio/Video
Device:mobile phone	hasPlay	Games/Music
Device:mobile phone	hasDisplay	Time
Device:mobile phone	hasSet	Alarm

Cable STB	IP STB	Mobile Phone	PDA	Laptop	Desktop PC	Stereo Receiver	TV Monit
tune to a channel	tune to channel	make a call	create and store data files	create and store data files	create and store data files	play audio files	display vid signal
volume control	volume control	receive a call	run programs	run programs	run programs	volume control	volume control
record (DVR)	record (DVR)	messaging (SMS/MMS)	download programs	download programs	download programs	retrieve/display CD info	change channel / input
playback	playback	take pictures/video	play video/audio	play video/audio	play video/audio	amplify other audio sources	
transfer/store pictures	transfer/store pictures	store/display pictures/video	record video/audio	record video/audio	record video/audio		
transfer/store audio/video	transfer/store audio/video	store/transfer audio/video files	handwriting recognition	handwriting recognition	handwriting recognition		
display EPG	display EPG	play games	monitor gps location	qwerty KB	qwerty KB		
VoD	VoD	play music	bluetooth	bluetooth	bluetooth		
purchase media	purchase media	record audio	wifi	wifi	wifi		
access multiple audio channels	video telephony	edit pictures	office apps	office apps	office apps		
access closed captioning	online shopping	(pedometer)	display time	display time	display time		

Fig. 1. Multiple device and Task

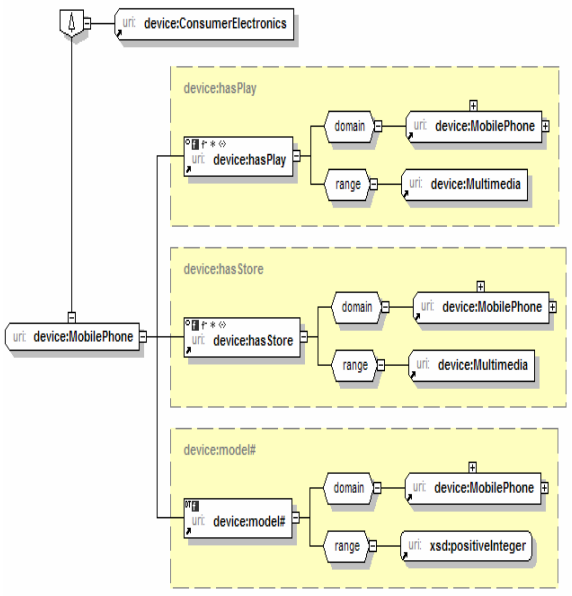

Fig. 2. A snapshot of RDF Schema Diagram of MobilePhone

A. *Ontological Representation of Multiple Device Interaction*

We create a new ontology for multiple device interaction using an ontology compliance level (OWL Lite) and generate the model and RDF/XML schema in the Semantic Works interface from Altova Inc [3]. We have defined three classes 1) ConsumerElectronics 2) MobilePhone 3) Multimedia.

We define MobilePhone as a subclass of ConsumerElectronics, which essentially states that any instance of the MobilePhone class must also be an instance of the ConsumerElectronics class.

We use the Multimedia class to (i) define it as the range of a property called device:hasPlay or device:hasStore and (ii) create instances of Multimedia.

We define the class MobilePhone to be the domain of the properties hasStore/hasPlay, and the class Multimedia to be the range of the properties hasStore/hasPlay. This would mean that the properties hasStore/hasPlay applies to the class MobilePhone and takes values that are instances of the class Multimedia.

Properties are created at a global level and then related to different classes. In our ontology, we deal with two properties: a) hasStore, to carry information about the type of the multimedia files. The multimedia can be Audio, Video or Picture. We will create this property as an object property. Doing this enables us to relate one resource to another. In this case we wish to relate instances of the MobilePhone class to instances of the Multimedia class via the hasStore property. The class (or classes) that the property applies to is called the property's domain, while the set of values the property can take is called the property's range.

b) model#, is a literal value indicating the model number of the MobilePhone. We will create this property as a datatype property. It relates instances of the MobilePhone class to a positive integer (which is the name or number of the model).

In Fig. 2 we see that the class MobilePhone is a subclass of the class ConsumerElectronics, and has two properties: the object property hasStore and the datatype property model#. So far we have created three classes, ConsumerElectronics, MobilePhone, Multimedia, and two properties, the object property hasStore and the datatype property model#. We have defined both properties to apply to the MobilePhone class (by making this class the domain of the properties). Further, we have defined (i) the range of the hasStore property (that is the values this property can take) to be instances of the Multimedia class, and (ii) the range of the model# property to be a literal value of the XML Schema datatype positiveInteger.

Now we will first create three instances of the Multimedia class, which will be simple instances like video, picture and audio as shown in Fig. 3.

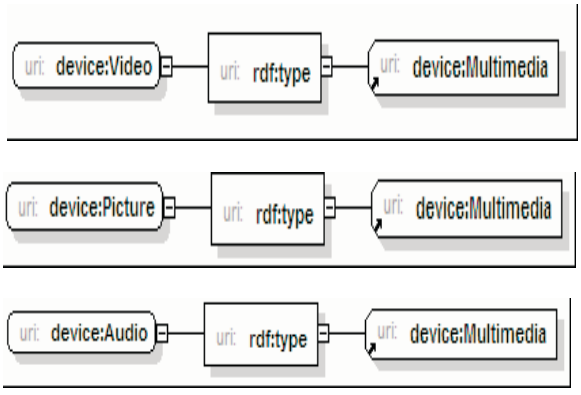

Fig. 3. Multimedia Instances

Next we define three more instances of the MobilePhone

class as shown in Fig. 4 and add a predicate. MobilePhoneAudio instance with its predicate is shown in Fig. 5.

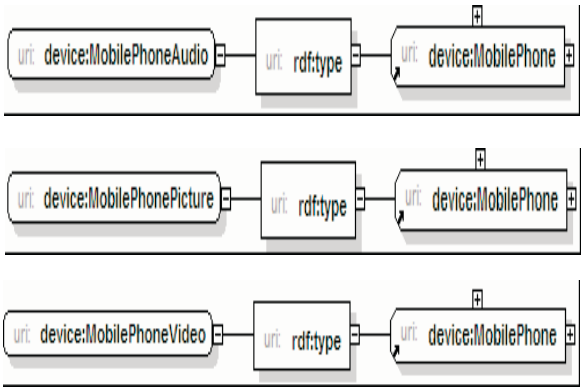

Fig. 4. MobilePhone Instances

The instance MobilePhoneAudio, MobilePhonePicture, MobilePhoneVideo has therefore been defined to:

- Be an instance of the class MobilePhone,
- Have an object property hasStore that takes the instance Multimedia as its object and have an object property hasPlay that takes the instance Multimedia as its object,
- Have a datatype property model# that takes the positiveInteger value SGH609 as its literal value.

In the next section we discuss besides the interaction of multiple devices, how the underlying metadata access can be enhanced by making use of semantic web technology.

III. Application of Semantic Web in DLNA to Enhance the User Experience

With the multitude of home electronics devices available, it is becoming difficult for consumers to manage their digital content. The *Digital Living Network Alliance* (DLNA) has specified an architecture that enables interoperability between the various devices and allows a user to enjoy the desired content. However, in a typical home, users now or in the future might store thousands of multimedia items across several devices, making it is tedious to search every single device individually. Additionally, in the case of a metadata based content search, not all metadata or the semantics associated with it might be available, resulting in incomplete search results. In this paper, we extend the DLNA architecture to address these problems. We use a specialized device as a service enabling platform and to hide the complexity of distributed content storage. Furthermore, we use the Semantic Web to provide an enriched and more comprehensive metadata-based multimedia search experience to enhance the user experience.

A. Architecture

The standard DLNA architecture as shown in Fig. 6 consists of several Digital Media Servers (DMSs) and a control point that communicates with these DMSs to search for multimedia content. Typically, a user uses a control point to invoke the Content Directory Service (CDS) actions to search for the

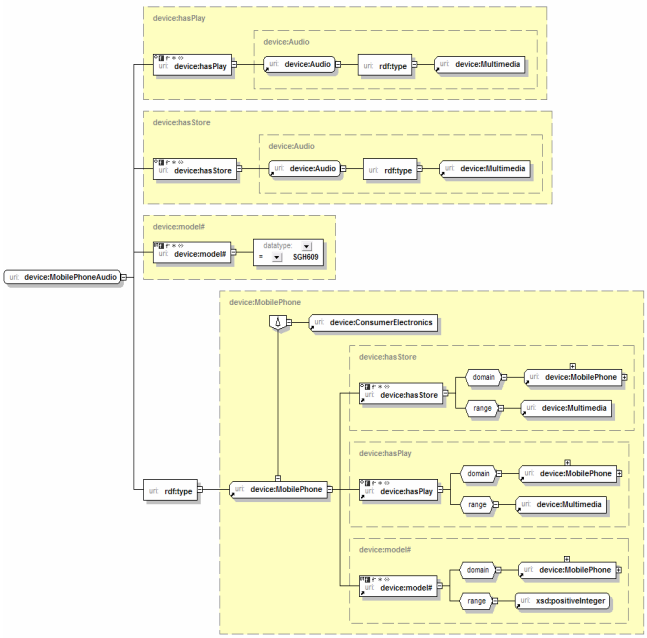

Fig. 5. A snapshot of RDF schema diagram of MobilePhoneAudio

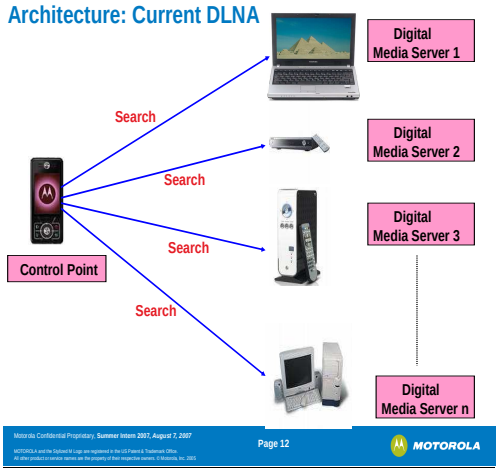

Fig. 6. An existing DLAN architecture

desired content hosted on that particular DMS, and he is required to repeat this for all available DMSs until he finds what he wants. Next we use a Virtual Media Server (VMS), which acts as a single point of contact for the user, as shown in Fig. 7, to relieve the user from the burden of contacting multiple DMSs separately. We also introduce another new component into the architecture: the Semantic Web Engine (SWE) as shown in Fig. 8. The SWE uses knowledge imported from external ontologies and data sources to extrapolate the

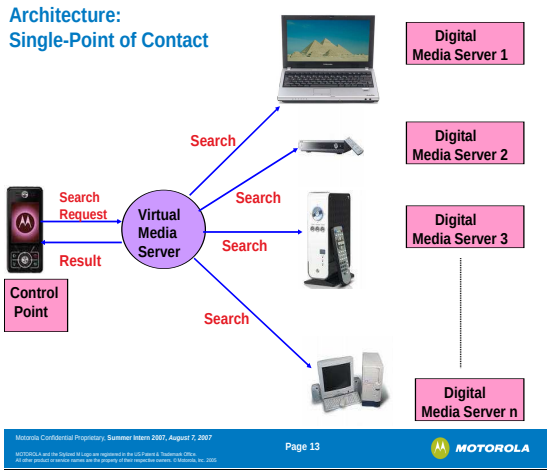

Fig. 7. Enhancement of DLNA architecture

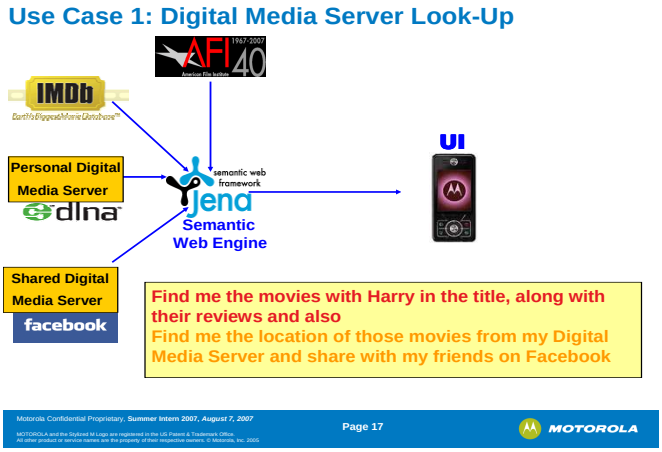

Fig. 9. Application of Semantic Web Engine in case of DLNA

semantics of the available metadata, thus enabling a smarter content search mechanism.

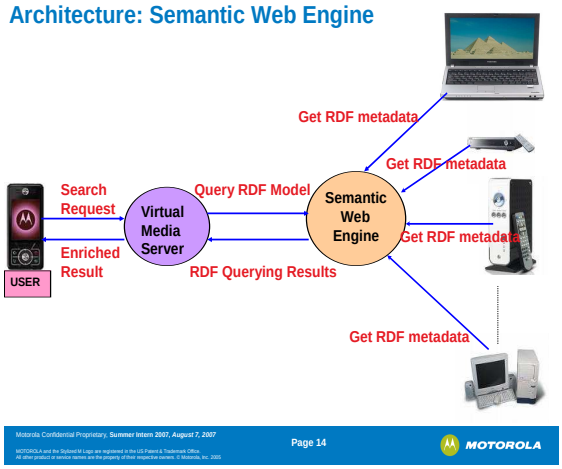

Fig. 8. Role of Semantic Web Engine with DLNA

B. Use Cases

We could think of the following scenarios for getting the benefit of semantic web engine and DLNA. In an ideal case, we could have searched the IMDb (Internet Movie Database) and AFI (American Film Institute) as an external ontological data sources to get movie information (such as movie reviews, ratings, background production information etc.) and then search the user's personal digital media server to see if that particular film is available in his personal collections. Also to extend this, the SWE can get the contact list (including shared DMS information) of the user's friends from an external ontological data sources (such as Facebook) and in a similar way each one of the user's friends could share the movie information they have.

Another interesting example could involve the user searching for songs sung by a specific singer. In addition to the simple CDS:Search() results matching the artist's name, the VMS could provide additional results with the help of the SWE. Let's say that the SWE can access an internet RDF audiography database. It can look up this information to find the bands that the given artist has been part of and then list music available in the digital home by those bands as additional search results. The SWE derives a URI representing

TABLE II
SEARCH RESULTS FOR SONGS

	Songs	Sung by
Results after simple Search through VMS	Dance Tonight	Paul McCartney
	Gratitude	Paul McCartney
Additional Results with Semantic Web Engine	Live and Let Die	Wings
	Band on The Run	Wings
	Hey Jude	Beatles
	Let It Be	Beatles

an RDF resource from the search keyword by prepending it with the base URI of the ontology data being used. In our example search for songs sung by Paul McCartney, the SWE uses the URI "http://www.somewhere.edu/audiographyband#PaulMcCartney" as the starting node for the RDF query, assuming the base URI "http://www.somewhere.edu/audiographyband#". It then uses the audiography band data imported from external data sources along with the content metadata requested from each of the DMSs to perform the RDF query and extract additional matches for the user's search request. The songs matching the search query are listed in Table II. We can observe that in addition to the songs marked by Paul McCartney, other songs from other bands "Beatles", "Wings" where Paul McCartney was in, have also been included in the results. Thus the search results were enriched and enhanced with the use of external ontological data.

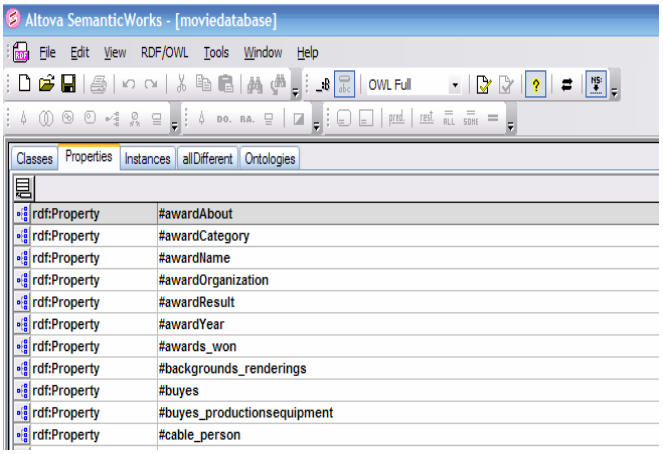

Fig. 10. A snapshot of the RDF vocabulary for Movie and Music

```
package jena.examples.rdf ;
import com.hp.hpl.jena.rdf.model.*;
/**Vocabulary definitions from moviedatabase.rdf @author
Auto-generated by schemagen on 10 Jul 2007*/
public class Moviedatabase {
/** <p>The RDF model that holds the vocabulary terms</p> */
private static Model m_model ModelFactory.createDefaultModel();
/** <p>The namespace of the vocabulary as a string</p> */
public static final String NS "http://movie";
/** <p>The namespace of the vocabulary as a string</p>*/
public static String getURI() return NS;
/** <p>The namespace of the vocabulary as a resource</p> */
public static final Resource NAMESPACE
m_model.createResource(NS);
public static final Property awards_won
m_model.createProperty( "http://movie/#awards_won");
public static final Property provides_audio
m_model.createProperty( "http://movie/#provides_audio");
public static final Property AlternativeTitle
m_model.createProperty( "http://movie/#AlternativeTitle");
}
```

Fig. 11. A snapshot of Java class file generated by Jena schemagen

IV. Prototyping

We have developed the Semantic Web Engine using Jena 2.5.3 [2] an open source semantic web toolkit. The demonstration software includes a RDF vocabulary of movie and music which has been generated using SemanticWorks [3] as shown in Fig. 10. This vocabulary has been converted to a java class file using the schemagen utility from Jena API which helps to get access to different properties of RDF graph based model. A sample of this class file is shown in Fig. 11. We assume there are two Digital Media Servers, one for the movie database and another for the music database, both of which host data in RDF metadata format as shown in Fig. 12 and Fig. 13. We use SPARQL [4], a query language which can select RDF triples from a triple set. Applications based on Jena API can search the movie-database and music-database individually by specifying a SPARQL query or can make a composite search on both databases at the same instant as shown in Fig. 14. The output of this query is also shown at the end of the Fig. 14.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:Moviedatabase="http://movie/#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://somewhere/
HarryPorter1">
<Moviedatabase:awardAbout>FlimFestival
</Moviedatabase:awardAbout>
<Moviedatabase:AlternativeTitle>Harry Potter and the
Philosopher's Stone </Moviedatabase:AlternativeTitle>
<Moviedatabase:awardCategory>Adventure
</Moviedatabase:awardCategory>
<Moviedatabase:awardName>Oscar</Moviedatabase:awardName>
<Moviedatabase:awardOrganization>France
</Moviedatabase:awardOrganization>
<Moviedatabase:awardResult>Pending
</Moviedatabase:awardResult>
</rdf:Description>
</rdf:RDF>
```

Fig. 12. Movie database for DMS1

```
?xml version="1.0"?>
<rdf:RDF xmlns:Musicdatabase="http://music/#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://somewhere/
HarryPorter1">
<Musicdatabase:assists_musiceditor>Elvis Presley
</Musicdatabase:assists_musiceditor>
<Musicdatabase:assists_propmaster>Michael Jackson
</Musicdatabase:assists_propmaster>
<Musicdatabase:composes>Alfred Newman
</Musicdatabase:composes>
<Musicdatabase:has_music>Bernard Herrmann
</Musicdatabase:has_music>
<Musicdatabase:musical_production>Hollywood
</Musicdatabase:musical_production>
<Musicdatabase:produces_music>Jimmy Hunter
</Musicdatabase:produces_music>
</rdf:Description>
</rdf:RDF>
```

Fig. 13. Music database for DMS2

V. Design of Dynamic User Interface

Tools for querying data have traditionally been text based, although graphical interfaces have been pursued [5]. An analysis of the text based queries had revealed a construction pattern that could be satisfied with a simple visual interface. The pattern consisted of finding intersections of groups of items, where the items could be popular music and movie information, and where the items of the intersection would be composers, singers and musicians. These groups of items were often constrained by several other parameters. But the user doesn't want to go through all these hassles to perform every request. This approach could not scale to a professional user who has to make hundreds of queries in a database. Therefore, we set out to build a Query UI that did not require the human user to become an expert of the system. The ongoing challenge is to allow a novice user to navigate the metadata and formulate the queries without much technical intervention. RDF vocabulary metadata navigation would be the first challenge for our users. This vocabulary consists of properties about the data. For our prototype application, there are three vocabularies, movie vocabulary, music vocabulary and iTunes vocabulary. Each vocabulary has its own nuances,

```
// create an empty model
Model model1 ModelFactory.createDefaultModel();
Model model2 ModelFactory.createDefaultModel();
// use the class loader to find the input file
InputStream in1 FileManager.get().open(Movie-metadata);
InputStream in2 FileManager.get().open(Music-metadata);
// read the RDF/XML files
model1.read(in1, "" );
model2.read(in2, "" );
// merge the graphs
Model model model1.union(model2);
//Create a new query
String queryString
"PREFIX Moviedatabase: <http://movie/#> " +
"PREFIX Musicdatabase: <http://music/#> " +
"SELECT ?x ?Title " +
"WHERE {" +
" ?x Moviedatabase:awardName "Oscar" +
" ?x Moviedatabase:Title
?Title.FILTER regex(?Title, "harry", "i")"+
" ?x Musicdatabase:composes "Alfred Newman" +
" }";
Output:
x: <http://somewhere/HarryPotter1>
Title: "Harry Potter and the Philosopher's Stone"
```

Fig. 14. An Example of a SPARQL Query and its Output

which must be respected while doing queries, such as movie vocabularies with categories that can be used themselves for querying.

For example, if a user enjoyed and owned music by the violinist Joshua Bell and was curious to find other music by him, today they could type "Joshua Bell" as a query in amazon.com and receive a long list of search results. Buried in the midsts of mostly CD titles is a DVD, The Red Violin, a movie in which Bell did not appear, but performed the haunting violin solos. Perhaps the user would find the buried movie item, perhaps not. The user could also do a search on Joshua Bell in IMDB.com and find The Red Violin among the search result items, but will not find any CD titles by Bell. In neither of these search examples will the user find:

- Music the user already owns by Bell
- Music the user's friends own by Bell
- Joshua Bell music currently on sale by a a local commercial entity

The approach described in this paper, with extensions just slightly beyond the range of our initial work, could provide such user discoveries. From the user's perspective, the search would be conducted on the VMS, which would coordinate the necessary component searches among a number of DMS's. The user would be spared dealing with the complexities of the DMS searches.

Our goals are to make the interface, simple, fun, and informative, but informative only to the level necessary for the user. Simplicity in the interface is important for the reasons previously mentioned in this paper. A simple interface would make it possible for the user to manage a larger and wider range of digital media across devices.

A sense of fun in the interface would encourage inquisitiveness, which in turn would drive down the level of user burden on performing searches. Our theory is that a sense of fun could even, in effect, increase the apparent performance of the system; not based only on the accuracy and usefulness of the search results, but on the number of searches the user comfortably or even enthusiastically performs. The more fun the user has, the more searches they will do, resulting in more desired search results.

Putting aside issues of trust and security which are beyond the scope of this paper, we can consider under what circumstances the user needs to be made aware of the details of the search. Often the underlying details of the system would make no difference to the user. Users simply want their services to work and work well. If one considers the act of making a phone call, the path of that phone call, through which cities or up and down to which satellites, is of no interest to the typical user beyond the quality of service for that call. When entering a web site url in a browser, the number of network hops necessary to reach that server and return the web page contents is of no consequence, beyond the speed and accuracy of the data retrieval. Similarly, when a user performs a search with the system proposed in this paper, they should be able to specify which general sources to search (i.e. which VMS). However, VMS access to specific DMS's may be added or removed without the user's knowledge. It is the job of the VMS to return the best results possible given the resources available at the time. The user should not be burdened with these specifics, unless of course they ask for them or if their quality of service is effected. Unless the search represents a cost of time (for what ever reason the information requested is not immediately available) or a cost of money (information requested is available from a commercial source for a fee), the user need not be burdened.

As the interface design work moves forward, these are the guiding principles we employ to help us envision and implement this work. Given the early stage of the project, specifics on the design will be left for a later publication.

VI. Conclusion

We present an architecture for a digital home, which includes a Virtual Media Server (VMS) and a Semantic Web Engine (SWE). We describe the basic communication between the different components in the described architecture and also present some examples where the presence of the SWE can enrich the quality of metadata-based multimedia search in the digital home.

References

[1] Digital Living Network Alliance; http://www.dlna.org.
[2] Jena Semantic Web Toolkit: http://www.hpl.hp.com/semweb/jena2.htm
[3] SemanticWorks2007:http://www.altova.com/products/semanticworks/ semantic_web_rdf_owl_editor.html
[4] SPARQL tutorial: http://www.w3.org/TR/rdf-sparql-query/
[5] S. N. Murphy, V. Gainer, H. C. Chueh, "A Visual Interface Designed for Novice Users to find Research Patient Cohorts in a Large Biomedical Database", AMIA Annu Symp Proc. 2003; 2003: 489-493.
[6] Sadhna Ahuja, Tao Wu & Ora Lassila, "Using the Semantic Web to Enhance the Digital Living Experience", IEEE CCNC 2006, Las Vegas (NV), January 2006
[7] Duane Degler, Scott Henninger and Lisa Battle, "Semantic Web HCI: Discussing Research Implications", SIGCHI 2007