Designing and Implementing an Active Personal Health Record System

Juha Puustjärvi
University of Helsinki
Helsinki, Finland
juha.puustjarvi@cs.helsinki.fi

Leena Puustjärvi
The Pharmacy of Kaivopuisto
Helsinki, Finland
leena.puustjarvi@kolumbus.fi

Abstract— **Current personal health record (PHR) systems are passive in the sense that they just provide a means for storing individuals PHR but they neither analyze nor provide potential improvements to individual's healthcare. This is regrettable, as active PHR systems could provide a wide variety of opportunities for improving the quality of healthcare in a cost effective way. In this paper, we describe our work on designing and implementing an active PHR system. In order to achieve semantic interoperability between the system and its information sources, we have developed a specific ontology for the active PHR system. The ontology based (RDF coded) PHRs are physically stored in a relational database as the active features of the PHR system can be easily implemented by the triggers supported by most relational database systems. We also give rules and illustrative examples how ontology based RDF data can be transformed into relational model, and how database triggers can be specified on that data.**

Keywords - personal health record; ontologies; OWL; RDF; active databases; relational databases; triggers

I. INTRODUCTION

Currently there is no generally accepted definition of personal health record (PHR). In general, the term PHR is variably used to describe either a data file or a software application using either personal computer or Internet technologies [1, 2, 3].

In this paper, by the term PHR we refer to a collection of information about individual's health and health care that is gathered from different sources such as from health care providers, pharmacies, insurers, the consumer, and third parties such as gyms. A PHR typically includes information about medications, allergies, vaccinations, illnesses, laboratory and other test results, and surgeries and other procedures [4]. By the term PHR system we refer to a software application that manages PHRs.

Through the introduction of sophisticated PHR systems we can contribute to preventive medical care and achieve better health and well-being while reducing healthcare cost. However, achieving this goal presupposes that many changes on current PHRs and PHR systems are met.

A key requirement of PHRs is that it should be based on controlled vocabulary [5]. By controlled vocabulary we refer to an organized collection of words and phrases that some group has placed [6]. Further, controlled vocabulary should be shared by PHRs and the sources from which data is gathered into PHRs; otherwise the gathered data have to be transformed into format that is compliant with the vocabulary.

XML based CCD- [8] and CCR-standards [9] have commonly used as controlled vocabularies within PHRs. Also, in order to increase the expression power of vocabularies, ontology-based vocabularies for PHRs have been developed.

The controlled vocabulary that we use is ontology based. The reason for not using CCR – or CCD –standard is that these standards are based on XML-schemas. The problem with XML-schemas is that they only specify the structure of the documents (i.e., nesting of tags); and due to the lack of semantics we cannot represent them in a format that is compliant with the database schema, which specifies the semantics of the database.

Also a problem with XML schema based PHRs is that they organize data in document-centric way, i.e., they are collections of documents such as documents including lab tests, prescribed medications and illnesses. By contrast, PHR's effective usage and analysis is data centric, meaning that data should be extracted from various documents and then integrated according to specific criteria. Unfortunately the computation required by such queries is not provided by the query languages that are designed to address XML documents, e.g., XPath [10] and XQuery [11].

Apart from the problems of PHRs representation formats is the problem of current PHR systems' passivity, i.e., they just provide a means for managing individuals PHR but they neither analyze nor provide potential improvements to individual's healthcare. This is regrettable, as active PHR systems could provide a wide variety of cost effective opportunities for improving the quality of healthcare.

In this paper, we report our work on developing and implementing an active PHR system. The key idea is to implement an active PHR system by exploiting the functionalities provided by active relational database systems. The key point of active database systems is their ability to support *alerts* in the form of SQL triggers [9].

An SQL trigger is procedural code that is automatically executed in response to certain events on a particular relation (table) in a database [9]. For example, when inserting a new test result occurs then an analysis on the new values are executed, and based on the analysis a possible action is taken (e.g., generating an email to patient's physician or updating another relation).

In this paper, we restrict on the issues that relate to the designing and implementing an active PHR system for open environment. By open environment we refer to an infrastructure were systems are open in the sense they can interoperate, and so also the organizations are able to work together.

 ISBN: 978-1-61208-119-9

First, in Section 2, we represent our developed controlled vocabulary, called *extended PHR-ontology*. It does not only include concepts that relate to individual's health but also information that is required for conceptualizing information therapy as well as the active elements of the ontology; and hence we use the adverb *extended*. In particular, we represent a subset of the extended PHR ontology in a graphical form and in OWL. We also illustrate how its instances are presented in RDF as the PHR system receives all imported data in RDF-format. This format is compliant with the extended PHR ontology.

In Section 3, we first give the rules that are used in transforming the extended PHR-ontology into relational model. Then we give rules and examples how RDF coded instances of the OWL ontology are transformed into tuples and stored in relations. We also illustrate the role of the notion of *views* in storing PHRs in relational databases. In addition, we give an example of defining an SQL trigger for a PHR. Finally, Section 5 concludes the paper by discussing the advantages and limitations of our developed solutions as well as our future research.

II. EXTENDED PHR ONTOLOGY

A. Graphical Representation of Extended PHR Ontology

In the context of computer science, an ontology is a general vocabulary of a certain domain, and it can be defined as "an explicit specification of a conceptualization" [13]. It tries to characterize that meaning in terms of concepts and their relationships [14]. It is typically represented as classes, properties, attributes and values. As an example consider a subset of the *extended PHR ontology* presented in Fig. 1.

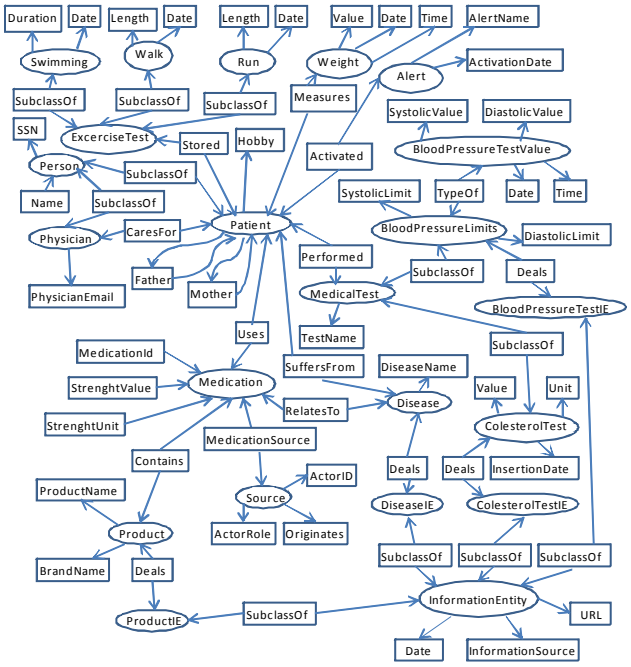

Figure 1. A subset of the extended PHR ontology.

In this graphical representation ellipses represent classes and subclasses while rectangles represent data type and object properties. Classes, subclasses, data properties and object properties are modeling primitives in OWL (Web Ontology Language) [15]. Object properties (e.g., Uses) relate objects to other objects while data type properties (e.g., MedicationId) relate objects to datatype values. In Fig. 1 we have presented only a few of objects' datatype properties.

B. Developing Extended PHR Ontology

Ontology development process is comprised of many stages. First, the scope of the ontology should be specified. The purpose of the PHR-ontology is to describe the concepts of the domain in which PHRs take place. Hence, a PHR-ontology describes the concepts (as well as their relationships) such as demographics, immunizations, allergies, diagnoses, procedures and medication.

In developing a PHR-ontology we do not have to start from scratch as we can find the most relevant concepts from the XML-Schema of the CCR-file [9]. In transforming the XML Schema to OWL-ontology we have used on the whole the following rules.

1. The complex elements of the XML-Schema are transformed into OWL classes.
2. The simple elements of the XML-Schema are transformed into OWL data properties such that the complex element is the domain of the data properties.
3. The attribute of the XML-Schema are transformed into OWL data properties.
4. The relationships between complex elements must be named and transformed to OWL object properties.

In order to illustrate these rules consider the following simplified example of a CCR-file [9] in Fig. 2.

```
<ContinityOfCareRecord>
  <Patient> <ActorID>Person.12345></ActorID></Patient>
  <Medications>
    <Medication>
      <Source>
        <ActorID>Pharmacy of Kaivopuisto</ActorID>
        <ActorRole>Pharmacy</ActorRole>
      </Source>
      <Description>
        <Text>One tablet ones a day</Text>
      </Description>
      <Product>
        <ProductName>Voltaren</ProductName>
        <BrandName>Diclofenac</BrandName>
      </Product>
      <Strenght>
        <Value>50</Value><Unit>milligram</Unit>
      </Strenght>
      <Quantity>
        <Value>30</Value><Unit>Tabs</Unit>
      </Quantity>
    </Medication>
  </Medications>
</ContinityOfCareRecord>
```

Figure 2. A simplified example of a CCR file.

This figure represents a CCR file that has a medication list (element Medications), which is comprised of one medication (element Medication) that is source stamped by the Pharmacy of Kaivopuisto.

Actor is a complex element as it includes simple elements *ActorId* and *ActorRole*. Hence, according to rule 1, complex element *Source* is transformed into OWL class *Source*. According to rule 2, simple elements *ActorId* and *ActorRole* are transformed to OWL data properties such that the domain of these elements is the OWL class *Source*. According to rule 4, the relationship between the complex elements *Medication* and *Source* should be named and transformed to OWL object property. As presented in Figure 1, we have named this object property '*MedicationSource*'.

Note that in the extended PHR ontology, besides the health related concepts, we have presented information that is used by the alerts. For example, the class *Alert* is comprised of the alerts that a patient can activate for his or her PHR. Further, the data properties *SystolicLimit* and *DiastolicLimit* (a patient's blood pressure is usually expressed in terms of the systolic pressure and diastolic pressure (mmHg), e.g., 130/80) are patient specific limits for blood pressure in the sense that if patient's blood pressure is below or over these values then an alert can take an appropriate action, e.g., generate an email to patient's physician. An SQL specification of such an alert will be given in the next Section.

C. Capturing Information Entities into PHR Ontology

A useful feature of the extended PHR ontology is that it also supports information therapy. The goal behind information therapy is to prescribe specific evidence based medical information to specific patients at just the right time to help them make specific health decisions or behavior changes [16, 17]. Through the extended PHR ontology we can easily contribute to information therapy as it models information entities (class *InformationEntity* in the graphical ontology in Fig. 1) as well as their relationships to relevant terms (e.g., *Disease* and *BloodPressureTest*) in the ontology. As a result, we can specify an alert, which is activated when a disease is discovered from a patient, and then the information entities or their links (if any), which deals the disease, are automatically delivered to the patient.

Fundamentally extended PHR ontology comprises the vocabulary that the patient can use in describing his or her personal health information. Hence we do not assume that a patient uses all the terms of the vocabulary (ontology). For example datatype properties *Father* and *Mother* are included in the vocabulary, but the patient does not have to give values for these properties.

Note also that the datatype property *Activated* connects classes *Patient* and *Alert*. By giving the values for the datatype properties of the class *Alert* the patient indicates, which alerts he or she has activated. That is, activating an alert requires only an update of patient's ontology. Based on patient's ontology the PHR system then generates an appropriate SQL trigger [12] in the database.

D. Representing the extended PHR Ontology in OWL

A subset of the graphical ontology of Fig. 1 is presented in OWL in Fig. 3.

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-nsl#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#">

        <owl:Ontology rdf:about=""Extended PHROntology/>
         <owl:Class rdf:ID="Person/">
         <owl:Class rdf:ID="Alert">
         <owl:Class rdf:ID="Medication/">
        <owl:Class rdf:ID="Disease/">
        <owl:Class rdf:ID="Patient/">
        <rdfs:subClassOf rdf:resource="#Person"/>
        </owl:Class>

        <owl:DatatypeProperty rdf:ID="ActivationDate">
                    <rdfs:domain rdf:resource="#Alert"/>
                    <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>

        <owl:DatatypeProperty rdf:ID="ISSN">
                    <rdfs:domain rdf:resource="#Person"/>
                    <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>

         <owl:ObjectProperty rdf:ID="Activated">
                    <rdfs:domain rdf:resource="#Patient/">
                    <rdfs:range rdf:resource="#Alert/>
        </owl:ObjectProperty>

        <owl:ObjectProperty rdf:ID="Uses">
                    <rdfs:domain rdf:resource="#Patient/">
                    <rdfs:range rdf:resource="#Medication/>
        </owl:ObjectProperty>

        <owl:ObjectProperty rdf:ID="Suffers">
                    <rdfs:domain rdf:resource="#Patient/>
                    <rdfs:range rdf:resource="#Disease"/>
        </owl:ObjectProperty>
                    .
                    .
                    .
</rdf:RDF>
```

Figure 3. A subset of the extended PHR ontology in OWL.

In importing data into active PHR the data items are presented in RDF, which is a standard model for data interchange on the Web. It has come to be used as a general method for conceptual description of information that is implemented in web resources, using a variety of syntax formats. RDF data is often stored in relational database.

RDF itself is a data model. Its modeling primitive is an object-attribute-value triple, which is called a statement [18]. In order that RDF data can be represented and transmitted it needs a concrete syntax, which is given in XML, i.e., RDF statements are usually coded in XML. Hence, RDF inherits the benefits associated with XML. However, other syntactic representations are also possible, meaning that XML-based syntax is not a necessary component of the RDF model.

One *RDF description* may contain one or more *RDF statements* about an object. For example, in Fig. 4, the description concerning "Voltaren" contains two RDF statements: the first states that its type is *ProductName* in the extended PHR ontology, and the second states that its *BrandName* in the extended PHR ontology is Diclofenac.

 ISBN: 978-1-61208-119-9

```
<rdf:RDF
   xmlns : rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
   xmlns : xsd=”http://www.w3.org/2001/XMLSchema#”
   xmlns : po=http://www.lut.fi/Extended_PHR_Ontology#>

      <rdf:Description rdf:about=”120962-K3”>
         <rdf:type rdf:resource=“&po;Patient”/>
          <po : PatientName>Lisa Smith</po : PatientName>
                <po : Uses>MO-5481</po:Uses>
          <po : Performed>H-257L</po : Performed>
      </rdf : Description>

      <rdf:Description rdf:about=” MO-5481”>
          <rdf:type rdf:resource=“&po;Medication”/>
          <po : Contains>Voltaren</po : Contains>
          <po : StrenghtValue rdf:datatype=
                ”&xsd;integer”>30</po : StrenghtValue>
          <po : StrenghtUnit>Tabs</po : StrenghtUnit>
      </rdf : Description>

      <rdf:Description rdf:about=” 211708-8”>
          <rdf:type rdf:resource=“&po;Source/>
          <po : ActorRole>Pharmacy</po : ActorRole>
      </rdf : Description>

      <rdf:Description rdf:about=” Voltaren”>
          <rdf:type rdf:resource=“&po;ProductName”/>
          <po : BrandName>Diclofenac</po : Contains>
      </rdf : Description>
 </rdf:RDF>
```

Figure 4. An instance of active PHR in RDF.

III. STORING ACTIVE PERSONAL HEALTH RECORDS IN A RELATIONAL DATABASE

A. *Transforming OWL Ontologies into Relation Schemas*

In a relational database, all data is stored and accessed via relations [19]. A relation is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object.

For example in Fig. 5, the first tuple in the first relation indicates that the *SSN* of Lisa Smith is 1112444-A2 and her SystolicLimit is 157 and DiastolicLimit is 71. The second relation indicates three test values of Lisa Smith’s blood pressure tests.

BloodPressureLimits (SSN Name SystolicLimit DiastolicLimit)

111244-A2	Lisa Smith	157	71
121248-B9	John Kent	171	77
120351-A2	Jack Cruz	144	61
120941-C5	Bob Jones	164	68

BloodPressureTestValue (SSN Date SystolicValue DiastolicValue)

111244-A2	06042011	142	75
111244-A2	07042011	155	79
111244-A2	09042011	160	64

Figure 5. Relations BloodPressureLimits and BloodPressureTestValue.

We have used the following rules in transforming the extended PHR ontology into relational schema:

1. The name of the OWL class is the name of the relation.
2. Each property of the OWL class is an attribute of the relation.
3. The key of the relation is comprised of the identification of the OWL class and of the identification of those OWL classes that are in a multivalued relationship to the OWL class.

B. *Transforming RDF descriptions into Tuples*

We can easily transform the information presented in RDF into tuples as each tuple presents information of an object, and so a tuple corresponds a RDF-description (a set of RDF statements about an object). For example, the first tuple of the relation BloodPressureLimits is derived by transforming the following RDF-description:

```
<rdf:RDF
      xmlns : rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
      xmlns : xsd=”http://www.w3.org/2001/XMLSchema#”
      xmlns : po=http://www.lut.fi/ontologies/p-ontology#>

   <rdf:Description rdf:about=”111244-A2”>
       <rdf:type rdf:resource=“&po;Patient”/>
       <po : Name>Lisa Smith</po : Name>
      <po : SystolicLimit> 157</po : SystolicLimit>
      <po : DiastolicLimit> 71</po : DiastolicLimit>
    </rdf : Description>

</rdf:RDF>
```

C. *Specifying Views on Perwsonal Health Records*

Unlike ordinary relations in a relational database, a *view* does not form part of the physical schema: it is a dynamic, virtual relation computed from data in the database [19]. Technically, a view consists of a stored query accessible as a virtual relation composed of the result set of a query.

Views have an important role in implementing the active PHR system by relational database systems: as PHRs are only accessible by the patient and those that are authorized by the patient, we have to control the access of PHRs. This control can be easily carried out by views. For example, the first tuple of the relation BloodPressureLimits in Fig. 5 should be only accesses by Lisa Smith. To enforce this we can specify a view ‘SmithBloodPressureLimits’ as follows in SQL:

```
Create View SmithBloodPressureLimits AS
      SELECT *
      FROM BloodPressureLimits
      WHERE Name = ‘Lisa Smith’;
```

Now, the virtual relation behind the view ‘SmithBloodPressureLimits’ corresponds to the relation presented in Fig. 6.

 ISBN: 978-1-61208-119-9

After the view is specified, we can then easily set restrictions on its use, e.g., that Lisa Smith has rights to read and update it while her physician Ian Taylor has only rights to read it.

SmithBloodPressureLimits (SSN Name SystolicLimit DiastolicLimit)

111244-A2	Lisa Smith	157	71

Figure 6. The virtual relation of the view SmithBloodPressureLimits.

D. *Specifying Triggers on PHRs*

A database *trigger* [20] is procedural code that is automatically executed in response to certain events on a particular relation or view in a database. Triggers are commonly used to enforce and execute business rules, e.g., notify a manager every time an employee's bank account number changes.

Technically we use triggers in a similar way as in business applications. The domain is only different: we restrict on PHR related data and on physicians and patients rather than on employees and managers.

Triggers, also called *event-condition-action rules* or *ECA rules*, contain the following three parts:

- *Event*: A change to the database that activates the trigger.
- *Condition*: A query or test that is run when the trigger is activated.
- *Action*: A procedure that is executed, when the trigger is activated and its condition is true.

The SQL trigger statement gives the user a number of different options in the event, condition and action parts. As a result, there is a wide variety of useful triggers that can be specified on PHRs. For example, the following trigger sends an email to Lisa Smiths physician, if her systolic value exceeds her systolic value limit (157mmHg).

```
CREATE TRIGGER SmithSystolicAlert
AFTER INSERTION of SystolicValue
      ON BloodPressureTestValue
WHEN SystolicValue > SystolicLimit
EXEC sendmail´physician.ian.taylor@health-house.com,
      'Lisa Smith's systolic limit is exceeded'
```

Assuming that Lisa Smith is authorized his physician Ian Taylor to access her PHR, thus the physician can after receiving the email analyze Lisa Smith's blood pressure values, and then contact Lisa Smith either by email or phone.

E. *Examples of Useful Ttriggers*

Obviously there is a wide variety of triggers that are useful for patients. Here we give examples of some potentially useful triggers:

- When a new medicinal information entity is published (inserted into the extended PHR ontology), it is checked whether it is relevant for a patient, and then delivered to the patient. For example, if a new information entity deals diabetes, then it is delivered for the patients that suffer from diabetes.

- When a new disease is discovered (inserted into patient's PHR), relevant information entities are delivered to the patient.

- When a new drug is prescribed (inserted into patient' PHR), drug-drug interaction analysis with patients previous medication is done, and based on the analysis appropriate actions are taken.

- When a predefined amount of test results are inserted into PHR, then a graphical summary is generated to patient.

- If patient's medication is changed, then the changes in medical test values are delivered to patient's physician.

Which triggers are appropriate for a patient must be determined on case-by-case. By the term *PHR customization* we refer to the process of specifying appropriate alerts (triggers) for a patient.

Note that the data stored in PHR is also dependent on the triggers included in patient's PHR system. For example, if Lisa Smith would not have chosen the *SmithSystolicAlert* then neither her systolic limit value is required to be stored in her PHR.

IV. CONCLUSION

A variety of health care information services, software companies and insurers are beginning to provide PHR tools and services. However, many key challenges to the widespread adoption and exploitation of PHR tools have not been fully addressed. One interesting challenge is the inclusion of active elements in PHR systems.

Our argument is that through active PHR systems we can provide a wide variety of opportunities for improving the quality of healthcare in a cost effective way. It, however, requires that active PHR systems are customized according to individual's needs. That is, in installing a PHR system only those active elements are installed, which are appropriate for the individual.

In this paper, we have restricted on technical aspects of active PHRs. In particular, we have described our work on designing and implementing an active PHR system. Our key idea has been the exploitation of the triggering mechanism supported by most relational database systems. Exploiting relational systems is also a natural choice as they are also suitable for storing RDF coded data.

Using RDF for representing PHRs has also many gains over XML coded PHRs such as CCR or CCD based PHRs. The main defects of such PHRs are due to the fact that XML schemas only specify the structure of the documents while

 ISBN: 978-1-61208-119-9

not expressing any semantics. As a result, semantic interoperability cannot be achieved through such solutions, and so the interoperation must be implemented by hard-coding.

An important requirement in introducing our developed solutions is that the sources of the data that are stored in the personal health ontology transform the delivered data in the format that is consistent with the personal health ontology. This is the extra work required from the health care organizations producing data into the extended PHR ontology.

In our future research we will focus on developing the set of active elements (alerts) that will contribute patient centric healthcare. In particular, we will estimate the gains of the active elements as well as the complexity in introducing and deploying the elements. In addition, an important research problem of active PHRs relates to the risks of using active elements in healthcare. That is, an open research problem is the survey of the healthcare decisions and recommendations, which prescribing can be safety automated, and thus can be carried out by active elements.

REFERENCES

[1] Raisinghani M.S. and Young, E., Personal health records: key adoption issues and implications for management, International Journal of Electronic Healthcare. Vol. 4, No.1 pp.67-77. 2008.

[2] Lewis, D., Eysenbach. G., Kukafka, R., Stavri P.Z., and Jimison, H., Consumer health informatics: informing consumers and improving health care. New York: Springer. 2005.

[3] Tuil, W. S., Hoopen, A. J., Braat, D. D. M., Vries Robbe, P.F., and Kremer J. A. M., Patient-centred care: using online personal medical records in IVF practice, Hum. Reprod., November 1, 21(11). pp. 2955 - 2959. 2006.

[4] Puustjärvi J. and Puustjärvi L., Managing Medicinal Instructions. In the proc. of the International Conference on Health Informatics (HEALTHINF 2009). pp. 105-110. 2009.

[5] Liu. S., Personal Health Records: Stus-Quo and Future Pespectives, in Ubiquitous Health and Medical Informatics". Ed. S. Mohammed and J. Fiaidhi, IGI-Global, pages 43-63. 2010.

[6] Puustjärvi J. and Puustjärvi L., The role of medicinal ontologies in querying and exchanging pharmaceutical information. International Journal of Electronic Healthcare, Vol. 5, No.1 pp. 1 – 13. 2009.

[7] Puustjärvi, J. and Puustjärvi L., Towards Semantic Exchange of Clinical Documents, International Journal on Advances in Life Sciences (IJALS). Vol. 1, No.2&3 pp. 69 – 76. 2009.

[8] Continuity of Care Record (CCR) Standard. Available at: http://www.ccrstandard.com/

[9] CCD, What Is the HL7 Continuity of Care Document? Available at: http://www.neotool.com/blog/2007/02/15/what-is-hl7-continuity-of-care-document/

[10] XML Path Language (XPath). Available at: http://www.w3.org/TR/xpath.

[11] XQuery 1.0: An XML Query Language. Available at: http://www.w3.org/TR/xquery/

[12] SQL tutorial. Available at:
http://www.w3schools.com/sql/default.asp

[13] Gruber, M. and Thomas R., Toward principles for the design of ontologies used for knowledge sharing. Padua workshop on Formal Ontology, 1993.

[14] Davies, J., Fensel, D., and Harmelen, F. Towards the semantic web: ontology driven knowledge management. West Sussex: John Wiley & Sons.2002.

[15] OWL – WEB OntologyLanguage. Available at: http://www.w3.org/TR/owl-features/.

[16] Trevena L., Davey H., Barratt A., Butow P., and Caldwell P., A systematic review on communicating with patients about evidence. Journal of Evaluation in Clinical Practice 2006; 12,(1): 13-23.

[17] Mettler M., Kemper D., Information Therapy: Health Education One Person at a Time. Health Promotion Practice 2004 4(3) 214-217.

[18] RDF – Resource Description Language. Available at: http://www.w3.org/RDF/

[19] Introduction to Relational Databases. Available at:
http://www.databasejournal.com/sqletc/article.php/1469521/Introduction-to-Relational-Databases.htm

[20] Database Triggers. Available at:
http://download.oracle.com/docs/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm

 ISBN: 978-1-61208-119-9