American Journal of Bioinformatics Research 2012, 2(4): 33-39
DOI: 10.5923/j.bioinformatics.20120204.01

Service Oriented Tools for Medical Records Management and Versioning

A. S. Ellouze[1,*], A. Jmal[1], R. Bouaziz[2]

[1]Computer Science Department, Higher Institute of Technological Studies, Sfax, 88A 3099, Tunisia
[2]Computer Science Department, Faculty of Economic Sciences and Management, Sfax, 88A 3099, Tunisia

Abstract The technological impact of the Internet gave birth to telemedicine that provides both the digitalization of patient's medical records, and the coordination between different healthcare actors. Work in this domain has shown that the standard XML is one of solid supports for the specification of the structure and of the semantic of clinical information; especially XML also ensures publishing medical records through Web. However, evolutions in time of XML schema of these records as well as interpenetration of temporal aspects in medical information are not sufficiently handled. In this paper, we address the problem of modeling, managing and implementing temporal medical data. We define efficient mechanisms for the management of multi-version medical records. For each structure evolution, our approach for schema versioning of XML medical records consists in creating a new schema version and in preserving old ones and their corresponding data. We present an algorithm to generate and to maintain schema versions of documents. We show the feasibility of our approach by the development of an application based on service oriented tools to publish services of versioned electronic medical records through the web

Keywords Electronic Medical Record, Temporal XML, Version Management, Schema Versioning, Service Oriented Architecture

1. Introduction

Using standard XML (eXtensible Markup Language) in information systems becomes more and more imperative, because it facilitates the interoperability and the cooperation between these systems, especially by publishing documents and services through the Web. Several applications also require data historization. So, some works[1,2,3] have studied the temporal behaviour of XML documents, and defined the concepts of temporal XML documents, and non-destructive update operations.

Beyond historization of data, schema describing XML documents can themselves evolve in time. Historization of this schema allows to examine and to manage data according to current schema versions during the considered periods [4, 5 and 6]. By storing the successive versions of a document in an incremental fashion, XML repositories will be able to achieve: (1) the efficient preservation of critical information and (2) the ability to support historical queries on the evolution of documents and their contents.

These theoretical and technological issues satisfy the requirements of medical information systems. The telemedicine constitutes a typical example of these systems; it requi-res the patient's medical records digitalization as well as remote coordination between different healthcare actors.

Several studies showed that the use of XML documents is an effective solution for the exchange and the publishing of medical services[7,8]. Besides, these documents are characterized by a high degree of evolution in the time, not only for their data, but also for their structures.

In this paper, we propose an approach based on a temporal extension of XML documents, introducing four temporal dimensions: three for data historization (time of data validity in the real world, time of current data in database and time of event triggering the data validity), and one for schemas version historization (time of version application). This approach aims at bringing a chronological visualization of data evolutions of medical records and of structure evolution of these records. It is based on a multiversion medical data model in compliance with XML Schema, and on management mechanisms for medical records having multi-version structures. To show the feasibility of our approach, we have designed and implemented a service oriented application, baptized TOMRMS (Temporal On-line Management of patient's medical Records affected by Multiple Sclerosis) for the ASHMS (Association of protection of handicapped persons in Sfax-Tunisia), ensuring medical record management by means of versioning their

* Corresponding author:
samet_afef@yahoo.fr (A. S. Ellouze)
Published online at http://journal.sapub.org/ bioinformatics

XML schema [9]. We also intend to offer a Web interface ensuring temporal management of medical records of patients affected by Multiple Sclerosis is "MS". This application offers to voluntary doctors the opportunity to help the ASHMS to ensure medical follow-up suited to patients affected by MS. Besides, it offers to these patients historical queries, which concern the evolution of their medical records. Implementation of TOM-RMS is achieved according to an oriented service architecture service, under the J2EE environment.

The remainder of the paper is organized as follows: in the next section we review previous efforts in temporal/clinical XML data and we locate our work. An approach of medical record schema versioning is presented in the third section. The fourth section presents an algorithm that describes our solution of medical record version management. Functional and technical description of our application is presented in this section, which shows also, through interfaces, experiments as well as obtained results. Last section concludes this paper and presents aimed perspectives.

2. Related Work

Temporal database management systems offer capabilities of storage, updating and querying current, future and past data. In this context, two temporal dimensions are usually considered: valid time and transaction time[10]. Valid time is the time during which the fact is true in the real world. Transaction time is the system time during which a fact is recorded in the database as current data. More recently, this temporal extension of data management interested XML documents. Supporting valid time on the Web was studied in [11] by Grandi and Mandreoli who proposed a new <valid> markup tag for XML/HTML documents. Rather than temporal queries, the focus of their work was the visualization of temporal information, which they showed can be achieved via browsers and XSLT[12]. Besides, several data model extensions and XML query languages are proposed, to ensure temporal query and management of information. For instance, extensions to XPath were proposed by Amagasa [13], Dyerson[14], and Mendelzon et al.[15], whereas Gao and Snodgrass proposed the τXQuery language that adds various temporal constructs to XQuery[16]. Our approach allows supporting temporal representations and queries without any extension of XML and its query language. It proceeds to a temporal presentation of the clinical information, similar to the XML-based presentation for scientific data, proposed in[17]. But, to ensure the perpetuity of this presentation, we propose in addition an extension for medical records of a versioned schema. Our versioning Strategy is inspired from works of Brahmia et al and Bouaziz et al[5,6]. Authors proposed an approach ensuring multi-temporal XML databases management in a schemas multi-version environment.

The use of XML in order to exchange medical data as well as the integration of temporal dimension in the management of XML documents of these data are more and more imperative[18]. Valid time is the temporal dimension usually considered for clinical data[19]. For both conceptual and relational models, many approaches handle the formalization and the management of valid time semantics for this kind of data. An approach of valid time management for temporal semi-structured multimedia clinical data is proposed in[20]. It is based on the use of a Multimedia Temporal Graph Model "MTGM"[18]. MTGM is a graphic model representing semi-structured medical data having temporal and multimedia characteristics. The MTGM is then converted into an XML document. However, medical record schema versioning is not sufficiently studied. In[7,8], we find approaches allowing temporal dimension modelling for medical records. Evolution in time of XML schema and interpenetration of temporal aspects in medical information were handled collectively, to our knowledge, only in[9]. In this paper, we presented an approach for the management of medical records. These records and their schema versions are modeled through temporal and multimedia and multi-version XML documents. XML data management includes creation, modification and deletion of XML document. XML documents modification includes addition, modification and deletion of XML elements.

Besides, change detection mechanisms between schemas versions must be used. XyDiff algorithm[21,22] adapts LaDiff algorithm[23] to the specificities of XML documents, while X-Diff[24] detects the changes between XML documents based on an unordered model that is applicable to published relational data. DeltaXML[25] and Microsoft XMLDiff[26] are commercialised as change detection tools that enable the generation of differences between the XML schema versions and their representation in an XML format. We are inspired by these tools to detect changes between medical records schema versions.

3. Schemas Versioning

3.1. Basic Principles

Management of medical information systems having services exchanged through Web requires two appropriate components in order: (1) to maintain medical record schema versions under the Web and (2) to formulate queries containing temporal specifications.

The need to have several schema versions of a medical record is motivated by the following reasons:

Healthcare treatment techniques are always in evolution. A medical record defined according to traditional means can not be able to support information supplied with recent ones. For example, a medical image supplied with the standard DICOM[27] describing healthcare of a patient must be recorded under a new medical record schema version, if this record served previously for recording only radiological pictures

Patient's clinical states can include progressive symptoms which appear successively. Every new symptom has its own

characteristics. It is the case of patients affected by multiple sclerosis. Such disease is characterized by its progression and its evolution. Schema of the medical record that describes this disease can always change. Therefore, historization of progressive symptoms requires management of several schema versions of the medical record; each one describes an episode of the disease according to an interval application time

The approach proposed in this paper for the definition of medical records available on Web and providing complete historization of patient's therapeutic history is based on the following principles:

Medical record model is described according to XML Schema

The considered XML environment is multi-version and temporal where every medical record can have several XML schema versions. An XML schema of a medical record is considered as a logical entity which evolves in the time through different versions. Each of these versions has one of the following temporal formats: snapshot (static), transaction temporal (TT), validity temporal (VT) or bi-temporal (BT). With the TT, a medical record is stamped by begin transaction time and end transaction time allowing to retrieve any state (right or wrong) taken in the past by this document in the database; but it is not possible to indicate phase shifts of time with regard to the real world. With VT, a medical record is stamped by begin valid time and end valid time allowing to represent all the states of this medical record according to the history of the real world. This allows retroactive and post-active actions, but errors corrections are made in a destructive manner. Finally, BT format uses both transaction time and valid time; it associates the advantages of the two previous formats

Data incorporated into XML schema can then be time stamped in the following system temporal attributes: "Transaction_Start_Time", "Transaction_End_Time", "Valid_Start_Time" and "Valid_End_Time". Besides, an "Event_Time" attribute can be used to indicate occurrence time of the event having triggered the clinical information.

Medical record having an XML document form, which is valid to an XML schema version, must have the same temporal format of its schema

XML schema versioning is temporal: any XML schema version has a temporal interval [version start time – version end time]. Version end time remains unknown until the definition of a new version of XML schema; during this period, it is valued in the special symbol "UC" which means "until changed"

XML schema versioning is individual: schema of every medical record has its own versions, which are managed independently of XML schema versions of other medical records

XML schema of a medical record has always one current version, and zero, one or several past versions. When a new XML schema version is declared, it takes the place of the current version (if it exists) which becomes a past version

Versions of medical record XML schema are numbered according to an increasing order (for example, Version_V1.xsd, Version_V2.xsd, Version_V3.xsd, etc.). Current XML version has always the upper number

An XML schema version can be derived from any previous version: the XML database administrator can use any old schema version to define a new one

The new medical record XML schema version cannot be identical to the last XML schema version of this record (it must be different in its structure and\or its format); otherwise, it cannot be accepted

Management of XML data includes creation, modification and deletion of XML documents. Modification of XML documents includes addition, modification and deletion of XML elements. In a snapshot or validity temporal XML document, operations of modification and deletion of XML elements are destructive. However, these operations are not destructive in a transaction-time or bitemporal XML document. The definition of every operation is presented below

The designer has to update the XML schema of medical records, when they evolve in time, through several versions. Figure 1 represents a diagrammatically of our approach. It shows the transition from the current XML schema version number i of a medical record "r" to the schema version number i+1 of "r". The version i becomes a past version, and the version i+1 the current version.

By default, doctors have to define documents which are valid with regard to a specified XML schema version. Medical record changes must be valid to the XML schema version corresponding to this record. After every modification, the record must always be valid to its schema version. As for patients, they can query XML documents relative to their medical records, whatever the schema version of their records and without appointing it.

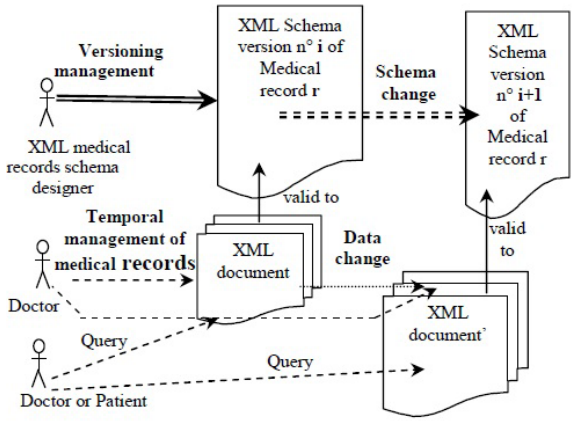

Figure 1. Temporal Management of medical records in a multi-version schema environnement

An XML schema version of a medical record is simply an XML document which is a set of XML elements. Each element is composed of several sub-elements and each element or sub-element is characterized by a set of attributes. So, schema change operations can handle with elements of XML schema (or sub-elements of XML schema) and/or with attributes of elements of XML schema (or attributes of the

sub-elements of XML schema). Transition from an XML schema current version to a new XML schema version is achieved by applying a set of schema change operations.

According to this approach, the creation of a new XML schema version does not affect the old XML schema versions and their underlying XML documents. The complete history of medical record XML schema and data changes is then preserved. When a new XML schema version is created, there is neither conversion of previous XML schema versions, nor revalidation of previous XML documents. Only, the attribute time of the version end concerning the previous schema version, which is valued to "UC" will be modified and will receive the moment which precedes time of the beginning of the new version. Schema changes do not affect, therefore, the underlying data.

For each medical record, new XML documents are created and modified according to the current XML schema version of this record; they always have to remain valid with regard to this version. So, change operations of data or XML documents relative to medical records (addition, modification or deletion of document elements) have no effect on the corresponding XML schema. Modification of an XML document cannot lead to a new valid document with regard to a different XML schema version; this operation must let any modified document valid with regard to its XML schema version.

3.2. Schema Change Operation Specification

We distinguish six main schema change operations:

Addition, deletion and modification of an XML schema element

Addition, deletion and modification of an attribute of an XML schema element

In Table 1 we present notations of schema change operations. We suppose that the following schema change operations are applied to $SV_{i,r}$ and produce $SV_{i+1,r}$, as result.

Table 1. the notation for XML schema change operations in medical records

Expression	Description
SVi,r	An XML schema version number i of record "r"
ei,r	An element e of SVi,r
Ei,r	The set of elements of SVi,r; Ei,r = {ei,r}
ai,r	An attribute a of ei,r
Att(ei,r)	The set of attributes of the element ei,r; Att(ei,r) ={ai,r}

Let us take the example of the operations below described, to evolve from VSi,r (Record_Vi.xsd) to the version VSi+1,r (Record_Vi+1.xsd):

The addition of a new element in a medical record XML schema allows managing data relative to a new symptom arisen for a patient. The addition of a new element, called "e", in the XML schema of a medical record can be formalized as follows:

AddElem(VSi,r,e) $\rightarrow$ VSi+1,r such that

$$Ei+1,r = Ei,r \cup \{e\} \quad (1)$$

The deletion (non destructive) of an XML schema element allows deleting metadata having no meaning anymore in a patient medical record. The deletion of an element, called "e", from an XML schema of a medical record is formalized as follows:

DelElem(SVi,r,e) $\rightarrow$SVi+1,r such that

$$Ei+1,r = Ei,r / \{e\} \quad (2)$$

The modification of an element "e" is conditioned by the membership of this element in the XML schema current version of the concerned medical record. The modification consists in one of the following operations: Addition of a new sub-element "s" to "e", deletion of a sub-element "s" from "e", renaming "e", modification of type of element "e". It can be formalized as follows:

ModifElem(SVi,r,e) $\rightarrow$SVi+1,r such that

$$Ei+1,r = Ei,r/ \{ei,r\} \cup \{ei+1,r\} \quad (3)$$

The addition of an attribute "a" to "e" cannot be achieved if there is an existing attribute of "e" with the same name of "a" in the current XML Schema version. It can be formalized as follows:

AddAtt(VSi,r,e,a) $\rightarrow$ VSi+1,r such that

$$Ei+1,r = Ei,r / \{ei,r\} \cup \{ei+1,r\} \wedge [Att(ei+1,r)=Att(ei,r) \cup \{ai+1,r\}] \quad (4)$$

The deletion of an attribute "a" from "e" cannot be performed if there is no existing attribute of "e" with the same name of a in the current XML Schema version. Its formalization is as follows:

DelAtt(SVi,r,e,a) $\rightarrow$ SVi+1,r such that

$$Ei+1,r = Ei,r /\{ei,r\} \cup \{ei+1,r\} \wedge [Att(ei+1,r)=Att(ei,r) / \{ai,r\}] \quad (5)$$

The modification of an attribute "a" of "e" cannot be performed if there is no existing attribute of "e" with the same name of "a" in the current XML Schema version. This operation can modify any attribute "a" of "e" (i.e. its name or its type). It can be formalized as follows:

ModifAtt(SVi,r,e,a) $\rightarrow$ SVi+1,r such that

$$Ei+1,r = Ei,r /\{ei,r\} \cup \{ei+1,r\} \wedge [Att(ei+1,r)=Att(ei,r)/ \{ai,r\} \cup \{ai+1,r\}] \quad (6)$$

This operation can modify any attribute "a" of "e" (i.e. its name or its type).

4. Technical and Functional Description of Schema Version Management Services of Medical Records

To ensure schema evolution and versioning of medical records, we use an appropriate XML document, called "VM_Document". This document includes all descriptive schema versions related to all existing medical records, as illustrated in Figure 2. It constitutes, therefore, an access point to all XML schema versions, as well as to all medical records valid to every schema version. It becomes possible and easy, with this document, to formulate complex and temporal queries by using XPATH and XQUERY languages.

We structure "VM_Document" with <Schema_name>

tags. These tags declare the set of medical information system XML schemas, as for example <Medical_Records_Structure>, <Diseases_Records_Structure> and <Doctors_Records_Structure>. We are limited in this paper to the schema formulation of <Medical_Records_Structure>. This last one contains tags below described.

<Schema_Version> tags, those declare versions of this schema. In each of these versions, medical records that are valid according to a particular version are attached.

<Version_URL>, <Version_Number>, <Version_Start_Time>, <Version_End_Time> and <Medical_Records> tags that are used under every <Schema_Version> tag. They declare respectively the URL containing the XML version description (defined through XML Schema language), number of the version, its application begin time, its application end time and all medical records which are valid according to this version.

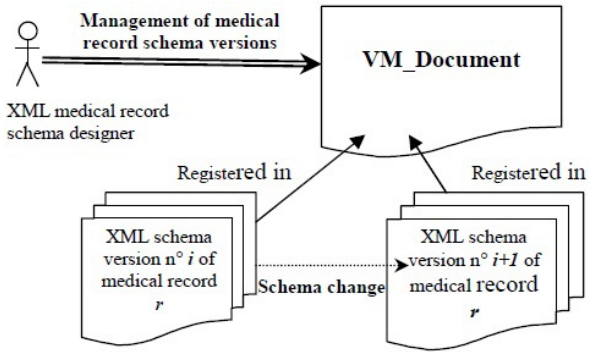

Figure 2. Management of medical records schema versions in global XML document:"VM_Document"

<Medical_Record> tags that are declared under <Medical_Records> tag, at the rate of a tag for every medical record, which is valid according to the considered schema version. A patient can have several medical records among which each one will be defined under the appropriate <Schema_Version> element.

<Record _Number> and <URL_File> tags, those declare respectively patient's identifier and URL of its bi-temporal XML document. Indeed, patient's medical record, valid to a considered schema version, can be historized through several XML bi-temporal documents. Patient's identification allows us to extract from "VM_Document", all bi-temporal XML documents which describe a medical record, as well as all corresponding schema versions.

Extract from "VM_Document", given in Figure 3, illustrates this structure.

To show the feasibility of our versioning approach of medical record schema, we have developed an application, baptized TOM-RMS, based on web service architecture.

We are motivated to implement TOM-RMS with web service architecture thanks to the following advantages: (1) interoperability, (2) simplicity and (3) modularity. For that purpose, we used JAXB "Java Architecture for XML Binding". TOM-RMS offers web services to ensure: (1) authentication by creating access accounts, (2) versioning management of medical records schema by adding new schema versions, each one is time-stamped with "Version_Start_Time" and "Version_End_Time", (3) graphic treelike visualization of schema versions by ensuring a graphic presentation of schema versions elements. (4) XML schema version generation (5) dynamic data entry forms generation, according to an appropriate schema version, and (6) dynamic generation of the XML document associated with the chosen schema version.

We propose a recursive algorithm, illustrated in Figure 4, called "genDocGlobal", for the generation and the insertion of elements in "VM_Document". Difference between two schema versions is calculated by the function "detectChange" which is based on tools proposed by XMLDiff of Microsoft [26].

```
<Medical_Records _Structure>
 <Schema_Version><!—Current schema version--!>
  <Version_URL>Version_V4.xsd</Version_URL>
  <Version_Number>4</Version_Number>
  <Version_Start_Time>2009-9-10</Version_Start_Time>
  <Version_End_Time>UC</Version_End_Time>
  <Medical_Records>
        <Medical_Record>
              <Record_Number>2312</Record_Number>
               <!-- 2312 is an identifier of a patient --!>
              <URL_File>2312 _1_4.xml</URL_File>
        </Medical_Record>
        <Medical_Record>
              <Record_Number>6</Record_Number>
              <URL_File>6_1_4.xml</URL_File>
              <URL_File>6_2_4.xml</URL_File>
<! --Patient 6 has two bi-temporal XML documents valid to
version 4 --!>
        </Medical_Record>
  </Medical_Records>
 </Schema_Version>
 <Schema_Version ><!--Previous schema version --!>
  <Version_URL>Version_V3.xsd</Version_URL>
  <Version_Number>3</Version_Number>
  <Version_Start_Time>2009-7-10</Version_Start_Time>
  <Version_End_Time>2009-9-9</Version_Start_Time>
  < Medical_Records >
  <Medical_Record>
              <Record_Number>6</Record_Number>
              <URL_File>6_1_3.xml</URL_File>
  </Medical_Record >
       <!--Patient 6 has two different schema versions of its
medical records --!>
  </Medical_Records >
 </Schema_Version>
</Medical_Records_Structure>
```

Figure 3. Extract from "VM_Document"

Our algorithm proceeds through the following steps:

Step 1: Calculation of the number of versions existing in "VM_Document"

Step 2: Creation of a new schema in "VM_Document", when number of versions is equal to zero

Step 3: Check existence of a current version, when number of versions is higher than zero

Step 4: Creation of a new current version in "VM_Document", when Num_Version is equal to zero

Step 5: Change detection between the new XML schema version, referenced by URL_New_Schema, and the version referenced by Num_Version. If there is no schema versions change, algorithm retrieves the appropriate <Medical_Record> tag that corresponds to the considered

patient. If there is no <Medical_Record> tag for the patient, it creates a new one. Otherwise, it adds <URL_File> tag to the <Medical_Record> tag. If there is a schema versions change, the algorithm executes step 6.

Step 6: Recursive call for the algorithm to handle the previous schema version in "VM_Document".

```
genDocGlobal (URL_New_Schema, Version_Start_Time,
              Version_End_Time, Num_Version,
              Record_Number)
// n is count of versions in "VM_Document";
//Num_Version is initialized in the program calling
//genDocGlobal with the identifier of the current version, if
//"VM_Document" exists; Otherwise Num_Version is //initialized
with 1;
// Record_Number is a number which identifies a patient;
// URL_New_Schema is the URL of the new XML schema
//version;
Begin
  n ← Number of schema versions in "VM_Document";
  if n = 0 then
    // "VM_Document" is empty
    inserSchema_Version (URL_New_Schema,
    Version_Start_Time,"UC", Record_Number_1_1.xml);
    // inserSchema_Version inserts element <Schema_Version>
    //in "VM_Document". In this particular case, it inserts the first
    //element <Schema_Version> in "VM_Document".
  else
    if Num_Version >= 1 then
      change ← detectChange (URL_New_Schema,
                  Version_V_Num_Version.xsd);
      // detectChange compares XML schema versions related to
      // URL_New_Schema and to Version_V_Num_Version.xsd.
      if change = FALSE then
        B ← findMedical_Record
              (Num_Version , Record_Number)
        // findMedical_Record returns B = True
        // when it retrieves a <Medical_Record> tag that
        //corresponds to Record_Number in
        //Medical_Records> tag, associated to Num_Version.
        if B=True then
          inserURL_File (Num_Version, Record_Number)
          // inserURL_File adds <URL_File> tag to
          //<Medical_Record> tag, that corresponds to
          //Record_Number, in /<Medical_Records> tag,
          //associated to Num_Version.
        else
          inserMedical_Record (Num_Version,
                        Record_Number)
          // inserMedical_Record adds <Medical_Record> tag,
          //that corresponds to Record_Number,
          // in <Medical_Records> tag, associated to Num_Version.
        end if
      else
        genDocGlobal (URL_New_Schema,
        Version_Start_Time, Version_End_Time,
        Num_Version - 1, Record_Number);
        // recursive call for genDocGlobal to handle the previous
        // schema version.
      end if
    else
      inserSchema_Version (URL_Schema_Num_Version,
      Version_Start_Time, "UC", Record_Number_1_n+1.xml);
      // creation of a new current version.
      updateTemporal_Attribute (URL_Schema_Num_Version,
                                Version_End_Time);
      // updateTemporal_Attribute updates end valid time of the
      //previous version (considered previously the current version).
      end if
    end if
End
```

Figure 4. genDocGlobal algorithm

5. Conclusions

Preserving complete therapeutic history of the patient and publishing electronic medical records through the web represent a critical requirement for the modern medical information system. In addition to retrieval and browsing capabilities, the preservation of medical records history evolution also requires the ability of managing of schema versions related to such documents. Actually, these requirements are not sufficiently met and wait for the development of new enabling technology. Therefore in this paper, we have proposed a novel approach to the management of electronic medical record archives through the web. This approach treats medical record XML schema changes as a versioning process instead of a simple evolution of these records. It also ensures the consistency of the XML database since (1) when a new medical record XML schema version is defined, it does not convert its previous XML schema versions and does not revalidate its previous XML documents which are valid to their XML Schema versions (2) XML medical record change operations do not affect the corresponding XML schema and version; (3) after modification of an XML medical record, this latter is still valid to its XML schema version.

The key features of the proposed approach are: (1) the evolution history of medical records is represented in standard XML by adding a temporal attributes and (2) complex temporal queries can be then expressed in XQUERY, without requiring temporal extensions to the XML standards.

Currently, we work on development of a new web service to make statistics on contents of medical records schema versions.

ACKNOWLEDGEMENTS

The authors thank Frikha Faouzi, the president of the association of protection of the handicapped persons in Sfax-Tunisia. We are grateful to the reviewers for their hard work and invaluable insights which will help to greatly improve the paper.

REFERENCES

[1] I. Tatarinov, G. Ives, A.Halevy and, D. Weld, "Updating XML," Proc. of ACM SIGMOD Conference, pp. 413–424, Santa Barbara,California 2001

[2] S. De Capitani, "An authorization model for temporal XML documents," Proc. of SAC'02, pp. 1088–1093, Madrid, Spain 2002

[3] M. G Manukyan and, L. A. Kalinichenko, "Temporal XML," Proc. of ADBIS, pp. 581–590, Vilnius, Lithuania 2001

[4] S. Chien, V. Tsotras and, C. Zaniolo, "Version management of XML documents," Proc. of the Third International Workshop on the Web and Databases, pp. 75–80, Dallas, TX 2000

[5] Z. Brahmia and, R Bouaziz, "An approach for schema versioning in multi-temporal XML databases," Proc. of

ICEIS 2008, Volume DISI, pp. 290-297, Spain, 2008

[6] R. Bouaziz and, Z. Brahmia, "Data manipulation in multi-temporal XML databases supporting schema versioning," Proc. of DaTaX 2009, Russia, 2009

[7] C. Combi and, B. Oliboni, "Managing Valid Time Semantics for Semistructured Multimedia Clinical Data," EDBT 2006 Workshops, LNCS 4254, pp. 375–386, 2006

[8] C. Combi, B. Oliboni and, R. Rossato, "Merging multimedia presentations and semistructured temporal data: a graph-based model and its application to clinical information," International Journal Artificial Intelligence in Medicine, 34(2):89–112, 2005

[9] A. Samet, Z. Brahmia and, R Bouaziz, "An XML-based approach for online management of electronic medical record evolution," Proc. of ACIT Conference, Sanaa, Yemen 2009

[10] C. S. Jensen, C. E. Dyreson and al, "The consensus glossary of temporal database concepts. Temporal Databases Research and Practice," Lecture Notes in Computer Science, vol. 1399, pp. 367– 405, Springer, Berlin, 1998

[11] F. Grandi and, F. Mandreoli, "The valid web: an XML/XSL infrastructure for temporal management of web documents," ADVIS, 2000

[12] XSL Transformations (XSLT), http://www.w3.org/TR/xslt/ Last access date: October 23rd, 2009

[13] T. Amagasa, M. Yoshikawa and, S. Uemura, "A data model for temporal XML documents," DEXA, 2002

[14] C. E. Dyreson, "Observing transaction-time semantics with TTXPath," WISE, 2001

[15] A. O. Mendelzon, F. Rizzolo and, A. Vaisman, "Indexing temporal XML documents," VLDB, 2004

[16] D. Gao and, R. T. Snodgrass, "Temporal slicing in the evaluation of XML queries," VLDB, 2003

[17] P. Buneman, S. Khanna, K. Tajima and, W. Tan, "Archiving scientific data," ACM Trans. Database Syst 29 (1), 2004

[18] C. Combi, B. Oliboni and, R. Rossato, "Merging multimedia presentations and semistructured temporal data: a graph-based model and its application to clinical information," International Journal Artificial Intelligence in Medicine, 34(2):89–112, 2005

[19] C. S. Jensen and, R. Snodgrass, "Temporal data management," IEEE Transactions on Knowledge and Data Engineering, 11(1):36–44, 1999

[20] C. Combi and, B. Oliboni, "Managing Valid Time Semantics for Semistructured Multimedia Clinical Data," EDBT 2006 Workshops, LNCS 4254, pp. 375–386, 2006

[21] A. Marian, S. Abiteboul, G. Cobéna and, L. Mignet, "Changecentric management of versions in an XML warehouse, " VLDB, 2001

[22] G. Cobena, S. Abiteboul and, A. Marian, "Detecting changes in XML documents," ICDE, 2002

[23] S. Chawathe, A. Rajaraman, H. Garcia-Molina and, J. Widom, "Change detection in hierarchically structured information," SIGMOD, 1996

[24] Y. Wang, D. J. DeWitt and, J. Cai, "X-Diff: a fast change detection algorithm for XML documents, " ICDE, 2003

[25] Y. Chen, S. K. Madriaa and, S. Bhowmick, "DiffXML: change detection in XML data, " in: DASFAA, 2004, pp. 289–301

[26] Microsoft XML Diff,

[27] http://apps.gotdotnet.com/xmltools/xmldiff/ Last access date: May 6th, 2009

[28] National Electrical Manufactures Association. Digital Imaging and Communications in Medicine (DICOM).

[29] http://medical.nema.org/dicom/2004.html. Last access date: February 2nd, 2009