A Middleware Approach to Integrate Referent Tracking in EHR Systems.

Shahid MANZOOR, MSc[1], Werner M. CEUSTERS, MD[1], Ron RUDNICKI, MA[1]
[1] Center of Excellence in Bioinformatics and Life Sciences, University at Buffalo, NY, USA

Abstract

The purpose of a Referent Tracking System (RTS) is to manage the representation of particulars in a database and to share this information with Electronic Health Record (EHR) systems. We describe how an implementation of such a RTS can be integrated in an EHR system using middleware technology based on web services. We describe the functional and technical requirements of such an approach and document our experiences with MedtuityEMR, an EHR system that stores patient data in XML.

Introduction

The Referent Tracking (RT) paradigm has been introduced to avoid the ambiguities that arise when using generic terms (from a coding system or terminology) to annotate patient data in Electronic Health Records (EHR)[1]. When, for instance, John consults a physician for a fracture in his left leg. The physician might use SNOMED-CT code 71620000 (‘fracture of femur’) to refer to John’s fracture when annotating for that specific encounter his diagnosis in the EHR. If John later suffers from a second fracture in the same bone, the physician will probably follow the same procedure as before and enter the same code in relation to this new encounter. The problem is that on the basis of these two encounter descriptions one can not determine whether the codes are referring to the same numerical fracture (as for instance would unambiguously be clear if the diagnosis were diabetes since a patient cannot have two different ‘diabetes-es’ in his life) or different fractures. The main reason is that codes from terminological systems or ontologies do not identify uniquely the entities to which they are assigned in the context of clinical record keeping. They rather describe what generic category the entities to which they are assigned belong to.

The RT paradigm resolves these ambiguities by assigning a globally unique ID, called Instance Unique Identifier (IUI), to the particular entities – hereafter called ‘particulars’ – on the side of the patient[1]. In the previous example, both fractures would get different IUIs while still being annotated with SNOMED-CT code 71620000.

We have developed a Referent Tracking System (RTS) to manage the representation of particulars in a database and to share this information with EHR systems[2]. This offers EHR systems, by means of the IUIs communicated to them by the RTS, to unambiguously refer to entities on the side of the patient, while still being able to further characterize the nature of these particulars by using terms and codes from terminologies.

Objectives

In this paper, we describe the design of a middleware application that allows the RTS to communicate with the EHR system *MedtuityEMR*[3], and provide some guidelines on how to build similar applications for other EHR systems. Under our design, the middleware application interprets the patient encounter data which in MedtuityEMR are expressed in XML with the objective to assign the appropriate IUIs to the particulars referred to in the MedtuityEMR data. Our focus here is on mechanisms that can be used to assess whether a particular referred to in an EHR has already been assigned a IUI in the RTS and this in order to maintain the global uniqueness of the identifier.

Materials and methods

Referent Tracking System

The RTS is a server which runs as a standalone application inside an apache tomcat HTTP web servers at port 8080. The server can communicate simultaneously with multiple EHR clients running at remote locations. The system is intended to be hosted by a health institute which serves as the hub for other health institutes (clients). The data representation in the RTS is based on the templates defined in the RT paradigm (Table 1).[4]

MedtuityEMR

MedtuityEMR is an EHR application developed by Medtuity Inc. In addition to clinical documentation, the features of MedtuityEHR include patient tracking, document management, messaging, reporting, and prescriptions.

MedtuityEMR has specially designed controls for quickly entering encounter information. Complicated encounters can be accurately documented through mouse-clicks only by means of templates relevant to 18 medical specialties such as Urology, Surgery, ENT, and Neurology. There are more than 1000 of these templates for conditions and treatments such as fractures, dental pain, head injury, and so forth.

RT Template	RDFS Class
Description	
$A_i = < IUI_p, IUI_a, t_{ap}>$	ParticularRepresentatio n
Act of assignment of IUIp to a particular at time tap by the particular referred to by author IUIa	
$N_i=< IUI_a, t_a, nt_j, n_i, IUI_p, t_r>$	PtoN
The particular referred to by IUI_a asserts at time t_a that n_i is the name of the nametype nt_j assigned to the particular referred to by IUI_p at t_r.	
$Co_i=<IUI_a\ t_a, cbs, IUI_p, co, t_r>$	PtoCo
The particular referred to by IUI_a asserts at time t_a that it is annotated by concept code *co* from terminology system *cbs* at t_r.	
$R_i = <IUI_a, t_a, r, o, P, t_r>$	PtoP
The particular referred to by IUI_a asserts at time t_a that the relationship r from ontology o obtains between the particulars referred to in the set of IUIs P at time tr.	

Table 1: RT templates description

Once a template has been filled out, MedtuityEMR generates a structured progress note which then is stored in compressed XML. An example is the control shown in Figure 1 taken from MedtuityEMR's 'fracture-femur' template which allows the clinician to enter data about the strength with which the patient can move the ankle.

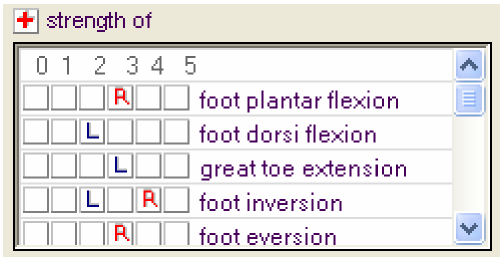

Figure 1: Input control for measuring the strength for flexions of a patient's feet

Listing 1 shows an excerpt of the XML generated on the basis of the input provided in Figure 1 for patient John. The text in italics corresponds to the first two lines in the control. *PtSession* forms the root element of the XML file. The patient's demographic data are in the *PtsInfo* element (in our example just showing last name and date of birth). *PtVisitInfo* contains the encounter description through a hierarchy consisting of *Leveln* (e.g *Level1*) and *Item*. The mapping between the XML elements corresponding to patient data and the GUI controls is captured by the GUID number.

```
<PtSessio  >
  <PtsInfo m_PtLastName="John"
m_PtDOB="01/01/1985 />
  <PtVisitInfo m_PtTimeIn="02/27/2007 02:44 PM">
    …
    <Level1 m_TemplateName="Fracture-femur"
m_TemplateGUID="{1379254  -C66D-4B47-A055-
CEA1A0A53C87 >
        Item m_Text="Examination">

       Level4 m_TemplateName ="">
     <Item m_Text="strength of right foot plantar flexion
is 3/5; strength of left foot dorsi flexion is 2/5; "
m_GUID="{65B2695  -81A1-4291-B26F-
344EBFD2B56B}  />
       Level4>
   ……
        Item>
   </Level1>
 </PtVisitInfo>
</PtSession>
```

Listing 1: An excerpt of the XML generated for the fracture-femur model

Results

Middleware Application

EHR applications can benefit from the RTS by directly calling its services. If an EHR application is complex, such as MedtuityEMR, containing many input screens for different disease templates, then directly accessing the RTS services would require programming changes in almost all parts of the EHR application. Therefore, we designed a middleware application which provides a bridge between the RTS and MedtuityEMR. As MedtuityEMR saves the patient encounters as XML, we have exploited this design and use the same XML for the communication between the RTS and MedtuityEMR. The middleware component identifies particulars by iterating through the XML and calls the services of the RTS on behalf of MedtuityEMR to annotate the particulars with IUIs. This approach keeps the MedtuityEMR application and the RTS integration specific implementations at separate places. The design of the middleware application is shown in Figure 2; the arrows show the data flow between the components.

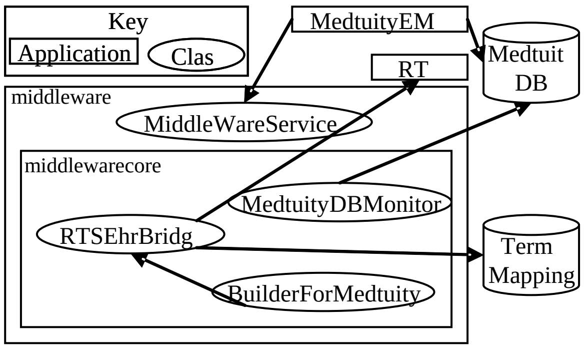

Figure 2: The Architecture of the middle ware application

It runs as a standalone application and provides its interface to MedtuityEMR via web services following several communication scenarios which each require a distinct level of integration. One scenario is to monitor the MedtuityEMR database for new transactions related to patient encounters. As soon as the monitoring component (*MedtuityDBMonitor*) finds newly entered patient data, it forwards them to the *RTSEhrBridge* component for further processing. Another approach is that MedtuityEMR sends actively the XML data via web services to the middleware application just before or after saving the encounter data. After annotating the identified particulars with the appropriate IUIs, the middleware application returns the results to the MedtuityEMR application. This approach, in contrast to the previous one, allows MedtuityEMR to manage the IUIs at the time of documenting the encounter. Of course, both approaches require software changes to be made in MedtuityEMR, the latter more drastically than the former.

The middleware application is designed as standalone application as well as a java library so that EHR applications can embed it easily in their programming environments.

Term Mapping Database

The information regarding which particulars are *possibly* referred to when an input control is used in the context of an encounter is stored in the term mapping database. '*Possibly*' here refers to the fact that some particulars may already be listed in the RTS such that, in line with the RT paradigm, the IUI already assigned to them has to be used for further reference. Other particulars might not yet be listed in the RTS, in which case a new IUI has to be created.

Deciding what is the case for a given data element, can be accomplished by looking at the ontological characteristics of the universals (types, kinds) of which the particulars under scrutiny are instances and under what scenario they are referred to in MedtuityEMR. We identified four different cases.

Case 1 involves particulars which exist throughout the life of a particular patient, examples being the patient himself, most body parts (e.g. his brain), and some disease (e.g. his diabetes) and some conditions such as his blood pressure. Whenever these particulars are first observed they are assigned an IUI, and that IUI is to be used in all future EHR statements made about them. This case encompasses also particulars which do not necessarily exist throughout a patient's life time, but which are assumed to ***still*** exist when they are referred to in the context of a new observation. Thus a patient can indeed loose his left foot, but if a clinician states to have measured the strength of a patient's left plantar flexion, then his left foot must exist.

Some particulars start to exist at t1 and disappear at t2, such as, hopefully, a fracture of the femur, or the flexion of his foot. Furthermore, John may have more than one femur fracture in his life and, without doubt, will flex his left foot quite often, each flexion being a new particular. However, in the context of, for instance, a follow-up encounter, some particulars ***can not*** be the same as observed during a previous encounter, while others ***may be*** the very same particulars as observed before. This leads to three further cases.

Case 2 involves particulars which ***may be*** re-observed but the context of the encounter is such that it can be decided upon automatically whether or not a new or existing one is observed. As an example, if John breaks his left leg and therefore visits a clinician at t1 for treatment, then the EMR application would record that *John (#IUI-1) has femur fracture (#IUI-2) in his left leg (#IU1-3)*. For every ***follow up*** visit (t2….ti) for ***that*** particular fracture, #IUI-3 must be used. If John later breaks again his left femur then a new IUI must be assigned, and that this is the case can be derived from the context that a ***new*** visit is entered, and not a ***follow-up*** visit.

Case 3 involves particulars which ***can not*** be re-observed during a new encounter, a foot flexion being an example, a measurement act being another one. Here we have primarily processes which have a life-time that is shorter than the duration of an encounter.

Table 2: Particulars involved in the registration that '*the strength of right foot plantar flexion is 3/5*'

Particular	Case
P1: John's act of right foot plantar flexion	3
P2: the act of giving counterforce to P1	3
P3: the assessment that the equality of forces with which P1 and P2 are applied justifies a score of 3/5	3
P4: the person who performed P3	1
P5: John's right foot plantar muscle group	1
P6: the disposition of John's right plantar muscle group to plantar flex with a certain strength	1
P7: John	1
P8: John's femur fracture	2

Case 4 involves particulars which ***may be*** re-observed and the context of the encounter is such that it can not be decided upon automatically whether a new or existing one is observed. This case applies also under a scenario when the integration of an EHR with the middleware component is very loosely such that the EHR cannot process clarification requests from the middleware component.

The practical consequence of the distinction drawn is that for particulars in case 1, a new IUI is to be assigned the 1st time they are observed, and that IUI is to be retrieved afterwards. In case 2, the EHR application can inform the middleware component whether a new IUI is to be assigned. In case 3, the RTS would create automatically a new IUI without any further questions to be asked. In case 4, the clinician has for each observation to provide the information whether or not a new particular is involved

As an example, the data-entry control in the state shown in figure 1 would make MedtuityEMR store the string '*strength of right foot plantar flexion is 3/5*' in John's EHR. Therefore, the *Term Mapping Database*, which can be viewed as an application ontology for MedtuityEMR, contains the information on how this string is to be interpreted in terms of the underlying particulars that must exist in order for the string to be a true statement. That information is derived on the basis of an ontological analysis carried out a priori.[5] Table 2 shows the results of this analysis, together with the classification of the particulars according to the 4 cases identified above. The *Term Mapping Database* contains such an analysis for each data control used in MedtuityEMR.

Web Services

The *Web Services* provides an interface to the remote clients.[6] The web services are the remote procedures that could be invoked from any programming environment. The web services forward all the clients' requests to the bridge component.

The middlewar core component

The *middlewarecore* component receives the MedtuityEMR patient's encounter XML either by monitoring the database or the XML is sent by MedtuityEMR through web services. It is composed of two software components i.e. *BuilderForMedtuity* and *RTSEhrBridge*.

The *BuilderForMedtuity* class is a parser for MedtuityEMR's XML structures. It extracts the EHR statements (such as *strength of right foot plantar flexion is 3/5*) by iterating through the XML source.

The *RTSEhrBridge* class first retrieves the configuration of involved particulars for each statement (as in Table 2) from the *Term Mapping Database*. Based on this information as well as on the encounter context information (whether a new visit or a follow-up is being documented), it decides whether IUIs for the particulars are first to be searched for in the RTS, or are to be created directly.

To assess whether particulars are already listed in the RTS, the RTSEhrBridge queries for these particulars by means of statements of the form:

```
getParticularsByPtoPWithPtoCo("IUI-1", null, "rts:co/SNOMED-CT/24176006");
```

In case the particulars are not listed in the RTS, or when the information in the Term Mapping Database states this directly, the *RTSEhrBridge* requests the RTS to create new IUIs for those particulars by means of statements of the form:

```
IUI-2 = rts.createParticular("02/27/2007", "IUI-10");
createPtoCo("IUI-2, "IUI-10", "rts:co/SNOMED-
CT/24176006", "02/27/2007" ,..);
createPtoP("IUI-1", "IUI-10", "has_part", "IUI-2",
"02/27/2007" ,..);
```

The *createParticular* method, in the example above concerning IUI-10 which stands for John, creates a reference to a particular and returns its IUI. The *createPtoCo* associates the MedtuityEMR *Right foot* term with the particular IUI-2. The *createPtoP* method asserts the *has_part* relation between #IUI-1 and IUI-2. The relation information between the particulars IUI-1 and IUI-2 is also found in the *Term Mapping Database*.

After the IUI assignment is done, the *RTSEhrBridge* class returns the IUIs to the *BuilderForMedtuity*. In case the encounter data are sent to the middleware component by MedtuityEMR, *BuilderForMedtuity* would associate the IUIs at the appropriate places in the XML and finally the resultant XML is sent back to MedtuityEMR.

In cases when it can not be determined whether a new or existing particular is observed, for instance under a scenario with less intimate integration or when the clinician is not willing to supply the additional information, the *RTSEhrBridge* class assigns a unique identifier to the particular which is ***not*** an IUI because it doesn't satisfy the requirement of singularity. This identifier would be created in the RTS by means of a statement of the type: 'ID = createIdentifier(tap, iuia)'. Because these identifiers are clearly distinguished from IUIs, it is always possible to supply the missing information later and to replace the identifier accordingly with an appropriate IUI.

Conclusion

The RTS application stores data in RDF and has services to query the data using RDF query languages such as SPARQL.[7] As a consequence, integrating the RTS into an EHR not only eliminates ambiguous references to particulars, but also converts the data into a formal representation which is optimized for automated reasoning. Particulars can be declared to be instances of the universals represented by the classes of a realism-based ontology or annotated with concept codes from terminologies such as SNOMED-CT. For example in our particular scenario (John's femur fracture) some assertions are in the RTS stored in triples of the form:

#IUI-1 *rts:r//OBO_REL/has_part* #IUI-2
#IUI-1 **co** rts:co//SNOMED-CT/116154003
#IUI-2 **co** rts:co//SNOMED-CT/24176006

The first statement represents that particular #IUI-1 enjoys the has_part relation with #IUI-2. The second and third assertions represent that the particulars #IUI-1 and #IUI-2 are respectively annotated with the SNOMED-CT codes for *patient* and *Extrinsic muscles of foot*. This improves interoperability between EHR applications and paves the way for more advanced clinical decision support systems.

Our approach covers all data control templates offered by MedtuityEMR except those with expect free text input.

Although MedtuityEMR allows patient encounters to be documented either as a new visit or a follow-up, the clinicians using the system are not bound by it so that because of this, IUIs cannot always be generated or retrieved. This is however a matter of proper user education, rather than an implementation issue on the side of the RTS.

Although we have thus far applied our technique to MedtuityEMR only, we are keeping our design generic so that it is able to work with other EHR systems as well. This requires for each such EHR system the implementation of a component similar to *BuilderForMedtuity* and to configure the *Term Mapping Database* in such a way that it reflects the type of data stored in the EHR.

Acknowledgements

This work is sponsored by a startup research grant from the New York State Center of Excellence in Bioinformatics & Life Sciences, Buffalo, NY.

References

1. Ceusters W. and Smith B. Tracking Referents in Electronic Health Records. In: Engelbrecht R. et al. (eds.) Medical Informatics Europe, IOS Press, Amsterdam, 2005;:71-76.
2. Manzoor S, Ceusters W, Rudnicki R. A Referent Tracking System for the Semantic Web. First international workshop on health care and life sciences data integration for the semantic web (HCLS-DI2007) May 8, 2007 Banff, Alberta, Canada
3. Medtuity Inc: MedtuityEMR. http://www.medtuity.com/.
4. Ceusters W, Smith B. Strategies for Referent Tracking in Electronic Health Records. J Biomed Inform. 2006 Jun;39(3):362-78.
5. Rudnicki R, Ceusters W, Manzoor S, Smith B. A Case Study in Integrating Referent Tracking into an Electronic Health Record Application (submitted).
6. Booth D, Haas H, McCabe F, Newcomer E, Michael Champion I, Ferris C, Orchard D. Web Services Architecture; W3C Working Group Note 11 February 2004.
7. Prud'hommeaux E and Seaborne A. SPARQL Query Language for RDF. W3C Working Draft 4 October 2006.