DICOM metadata as RDF

Michael Brunnbauer

Bonubase GmbH
Wingertstraße 58
68809 Neulußheim
brunni@netestate.de

Abstract: DICOM (Digital Imaging and Communications in Medicine) is an industry standard for medical imaging. We present an OWL ontology for DICOM metadata and dicom2rdf - a tool to extract RDF metadata out of DICOM files (Media type application/dicom, filename extension .dcm). We discuss benefits, use cases, limitations and possible enhancements.

1 The DICOM standard

DICOM objects consist of sets of attribute-value pairs that allow nesting (the values can be other DICOM objects). There are several thousand official attributes, an extension mechanism for private attributes and 27 data types called value representations (VR) for the values [1]. The data type for each official attribute is fixed.

The entity an attribute applies to is determined by its context: Attributes are organized into macros and modules of several attributes. An attribute can be used in several macros or modules.

Modules are organized into information object definitions (IOD), describing one or more real world objects called information entities (IE) like patient, study, series or image. The standard defines the IE (real world entity) a module describes in every IOD.

1.1 DICOM sequences

Sequence attributes in DICOM allow nesting of data. They create attributes containing sequences of one or more sequence items, effectively nesting DICOM objects. A sequence item is a set of attribute/value-pairs and does not necessarily correspond to an information object definition. It can also represent some kind of real world entity that is only implicitly defined by DICOM or some kind of abstract entity created by the particular organization of information in DICOM.

1.2 Official and private attributes

Official attributes are identified by a group and element number (16bit unsigned integers usually in hexadecimal notation). Additionally, every official attribute has a unique keyword:

Group/element number	Keyword	Name
(0010,0010)	PatientName	Patient Name

All official attributes have an even group number. Odd group numbers are used for private extensions. In order to organize private extensions without collisions, the following approach is used: The element numbers for each group are organized into 256 slices of 256 attributes. The first 16 slices are reserved. The first slice is used to store the implementor identification code for the 240 usable slices. Attribute gggg,0010 defines the implementor for the first usable slice gggg,1000-10FF (gggg is the group number) etc.

When creating data, software using a private extension searches for an unused slice N, stores its implementor identification code with the attribute gggg,N and maps its private attributes to the attribute slice N. As long as the implementor identification code is unique, the private attribute can be uniquely identified by this code, the group number and the attribute number within the slice (the first byte of the element number).

2 Modelling approach and related work

The modelling approach is a 1:1 mapping from the DICOM data model using real world entities from the DICOM ERM instead of abstract data entities where possible. We chose this approach as a compromise between a very easy mapping using only abstract data entities and a very complex mapping using a clean knowledge representation and existing ontologies.

The ontology is generated automatically from data extracted out of the DICOM standard documents and seems to be the first with a usable public RDF namespace.

In a similar approach, Kahn et al. have shown how a DICOM ontology could be modelled using the CT Image IOD as example [2].

The Stanford Center for Biomedical Informatics Research has started a project for a DICOM ontology at least 3 years ago but nothing seems to have been published and the current status is unclear [3].

The German company SOHARD is developing a product called SeDI (Semantic DICOM), which provides a SPARQL endpoint for a DICOM PACS (Picture Archiving and Communication System) using on the fly query transformation into DICOM C-Find

or Move requests. The query and result transformation uses an ontology that has not been published yet [4].

3 Generating the OWL ontology from the standard

This section highlights important design decisions for the ontology and how the ontology was generated automatically.

The attribute dictionary from the pydicom Python module [5] contains a list of DICOM attributes with group/element number, name, keyword and data type (VR). We used it to generate our own dictionary adding some missing attributes. This dictionary is used to generate URIs, labels and ranges for the attributes.

3.1 Official attribute URIs

The official attribute URIs use two equivalent variants of fragment identifier:

```
#Tag.<group>.<element>
```

or (owl:sameAs)

```
#<keyword>
```

<group> and <element> is the upper case hexadecimal group and element number (4 hex digits each) of the attribute. <keyword> is the unique keyword assigned by DICOM to the attribute, if available.

3.2 Private attribute URIs

Although private data elements are not yet part of the ontology, we created a URI scheme for them so the dicom2rdf tool can extract metadata using private extensions: Private attributes with odd group number and element number > 0x1000 use these fragment identifiers:

```
#PTag.<implementor>.<group>.<nelement>
```

<nelement> is the element number & 0xff (& = logical and). <implementor> is the normalized identification code (value of attribute <group>,<element1> - where <element1> is the element number right-shifted by 8 bits). The normalized form of the implementor identification code is produced by removing all whitespace, encoding it with UTF8 and escaping anything not a letter, digit or '_.-/' using a $xx escape (Python function urllib.quote() with % replaced by $ in the result).

3.3 Attribute ranges

The ranges of the attributes are based on their DICOM value representations (VR):

- Textual VRs (AE,CS,LO,SH,LT,ST,UT,PN) should be represented by plain literals. FF should be replaced by \r\n\r\n and ESC should be replaced by the unicode replacement character to enable RDF/XML serialization. PN (Person Name) values consist of several delimited components and should not be decomposed for the attributes of the ontology with VR PN. More popular attributes like foaf:familyName and foaf:givenName can be used for the components if the attribute applies to the corresponding person.

- Integer VRs (IS,SL,SS,UL,US,AT) should be represented by xsd:long (even if the VR defines them as int or short. This approach simplifies SPARQL querying).

- Decimal/Floating point VRs (DS,FL,OF,FD) should be represented by xsd:double (even if the VR defines them as float to simplify SPARQL querying).

- VRs representing time (DA,TM,DT,AS) are represented as xsd:date, xsd:time, xsd:dateTime and xsd:duration, respectively. DICOM has a notion of approximate dateTime or time: You start with the year/hour and leave out more specific elements if they are unknown. The information about what is not known is sacrificed with this representation in favor of usability and the potential performance benefit of xsd:dateTime and xsd:time.

- The UID VR (UI) should be represented by the corresponding urn:oid: URI.

- The sequence VR (SQ) should be represented by a List from the Collection ontology (http://purl.org/co/List).

- The unknown VR (UN) should be represented by one of the above types if the application knows the VR implicitly or by a plain literal if the VR is not known and the actual value is within the ASCII range.

- The OB and OW VR should be represented by individuals (not literals) with no restrictions on modelling.

Values of Attributes that can have multiple values (value multiplicity!=1) are represented by a List from the Collection ontology so those attributes have range http://purl.org/co/List. The list items have the type corresponding to the VR from the list above.

3.4 Attribute domains

As explained in 1., the entity an attribute applies to is defined by its context. If we know the information object definition (IOD) of the current data object, we can determine the

relevant information entity (patient, study, series, image, etc.) for the attributes in that data object. The relevant IOD for a data object is identified by the „SOP Class UID“ attribute (0008,0016).

We have created a parser to extract macro, module and information object definitions from a text version of part PS 3.3 of the DICOM standard [6]. This information is used to determine the relevant IE when extracting data with dicom2rdf and to define the attribute domains in the ontology.

Therefore, the domains for many attributes are information entity classes like patient, study, series, image or unions of several of these classes.

The URIs of information entity classes use

```
#IE.<IE>
```

fragment identifiers where <IE> is the name of the information entity with whitespaces removed.

Unfortunately, sequence items do not have to be information objects. In this case, the entities to which the attributes in the sequence apply are not clearly defined by the standard.

To resolve this messy situation and create a clean knowledge representation for information within sequence items is a gigantic task nobody has undertaken yet. As a first step, we decided to treat sequence items as independent entities and the attributes within a sequence item apply to the sequence item (unless the sequence item happens to be another information object).

We have given every sequence attribute a corresponding class of possible sequence items with the following fragment identifier:

```
#SequenceItem.<keyword>
```

or (owl:sameAs)

```
#SequenceItem.Tag.<group>.<element>
```

or

```
#SequenceItem.PTag.<implementor>.<group>.<nelement>
```

The part after "SequenceItem." is identical with the attribute fragment identifier of the sequence attribute.

The object of a sequence attribute should be a http://purl.org/co/List with the sequence items as list members.

3.5 The Ontology

You can find the ontology at http://purl.org/healthcarevocab/v1

It currently has 1593 classes, 2529 object properties, 4504 data properties and no individuals. It has been validated to be OWL DL compliant with the Pellet OWL 2 reasoner version 2.3.1 using the "lint" command.

4 dicom2rdf

dicom2rdf can be downloaded at https://github.com/Bonubase/dicom2rdf and has a BSD license.

The distribution also contains the parsers used to extract information out of the DICOM standard documents and the tool used to generate the OWL ontology.

DICOM test files can be obtained at:

http://www.dclunie.com/images/FluoroWithDisplayShutter.dcm.zip

http://www.dclunie.com/images/charset/korean_agfa_infinitt_2008-3.dcm.bz2

http://www.osirix-viewer.com/datasets/

RDF/XML files generated with dicom2rdf from these DICOM files can be found at:

http://www.netestate.de/dicom/

4.1 Entity URIs

DICOM allows entities to be identified by OIDs [7]. The OID of a data object is stored with the SOP Instance UID attribute (0008,0018). The corresponding urn:oid URI will be used for the data object if the attribute is present.

OIDs for information entities are identified by several attributes. The current version of dicom2rdf searches for 0020,000D for a study, 0020,000E for a series and 0020,0052 for a frame of reference.

If no urn:oid URI can be found for the data object or information entity, a unique hash URI relative to the generated RDF/XML file will be generated.

4.2 Possible improvements

There is room for improvement by identifying more inverse functional properties in the DICOM standard and by using additional properties and classes to create a cleaner knowledge representation.

5 Discussion

"A data model is not an ontology" may be an argument against our approach from a KR point of view.

Although we invested some time to be able to connect attributes to the real world entities they refer to (instead of simply connecting them to the data object), there are still many topics that could be solved better, for example by going through the official DICOM attributes one by one, in other words: by further investing large amounts of time. In order to further improve the ontology as such, the 1:1 mapping between DICOM and ontology attributes should be further reduced.

However, we think that the transition between data model and ontology is fluent. Our approach should be seen as a starting point offering immediate benefits.

What are those benefits ?

DICOM offers only limited options to query data in a PACS (Picture Archiving and Communication System). Search is limited to certain information entities and attributes, for which the PACS may or may not use a fast search index. In general, DICOM queries reflect the necessities and workflows of the institutions where the DICOM data is generated: hospitals and doctor's offices.

By contrast, RDF data in a triple or quad store can be queried with the SPARQL query language in almost any conceivable way - including graph patterns, aggregates and transitive queries. Matching of single values uses indexes and more advanced triple/quad stores also support indexes for value comparisons or full text search. DICOM metadata can also be easily mixed and queried with other data in a triple store.

An obvious use case would be research in a big archive of DICOM data.

In general, RDF and SPARQL are a good choice when heterogenous sources of (ragged) data have to be queried in an ad hoc way.

The DICOM standard is big and complex so it is no surprise that attributes are often used in a context where they are not allowed. It is quite obvious that a relational database is not a good fit for DICOM - especially when private extensions are considered.

There are limits to the scalability of certain SPARQL queries [8] but those limits also apply to other powerful query languages like SQL. There is no general purpose

technology that scales for all query problems that can be made scaleable at all. In such cases, specialized algorithms have to be implemented anyway.

The benefits of RDF pose a danger in some respect when it comes to privacy. RDF is a Web technology so data may easily end up to be publicly available when it should not be. The use of anonymizing tools, access control and strong security is vital in every application handling medical data.

5.1 SPARQL example

To illustrate the benefits, here is a SPARQL example query that is not possible as standard DICOM query. It calculates the average specific absorption rate for MR images and the number of images generated with equipment from General Electric between 1990 and 2010 (assuming that the queried set of images is representative for the scope of the query).

The DICOM data object and the information entities are connected with dcterms:subject.

```
PREFIX dicom: <http://purl.org/healthcarevocab/v1#>
PREFIX dcterms: <http://purl.org/dc/terms/>
select avg(?sar) count(*) where {
?image dicom:SAR ?sar.
?image dicom:AcquisitionDate ?date
FILTER(
?date >=
"1990-01-01"^^<http://www.w3.org/2001/XMLSchema#date> &&
?date <
"2010-01-01"^^<http://www.w3.org/2001/XMLSchema#date>
)
?io dcterms:subject ?image.
?io a <urn:oid:1.2.840.10008.5.1.4.1.1.4>.
?io dcterms:subject ?equipment.
?equipment a dicom:IE.Equipment.
?equipment dicom:Manufacturer "GE Medical Systems".
}
```

References

[1] Digital Imaging and Communications in Medicine (DICOM), Part 5: Data Structures and Encoding, Section 6.2 http://medical.nema.org/Dicom/2011/11_05pu.pdf

[2] Informatics in Radiology: An Information Model of the DICOM Standard http://radiographics.rsna.org/content/31/1/295.long

[3] DICOM Ontology (DO) Project http://bmir.stanford.edu/projects/view.php/dicom_ontology_do_project

[4] Semantic DICOM http://www.semantic-dicom.org/

[5] pydicom Python module http://code.google.com/p/pydicom/

[6] Digital Imaging and Communications in Medicine (DICOM), Part 3: Information Object Definitions http://medical.nema.org/Dicom/2011/11_03pu.pdf
[7] Digital Imaging and Communications in Medicine (DICOM), Part 5: Data Structures and Encoding, Section 9 http://medical.nema.org/Dicom/2011/11_05pu.pdf
[8] Semantics and Complexity of SPARQL
http://users.dcc.uchile.cl/~cgutierr/papers/sparql.pdf
The Complexity of Evaluating Path Expressions in SPARQL
http://www.theoinf.uni-bayreuth.de/download/pods12submission.pdf