

# **TATRC / Web3D Medical Group Task 2**

## **Report on FMA/SNOMED -> X3D metadata**

**Final deliverable 9/30/07**

**Nicholas F. Polys, Ph.D. & Andrew Ray**

The overall goal of this contract was to provide a method for semantic medical data to be incorporated into an X3D scenegraph via the use of the Metadata nodeset. The knowledge bases targeted by the contract were FMA and SNOMED. Our work consisted of five phases or sections that covered: obtaining FMA and SNOMED information, finding an appropriate representations of that information in X3D Metadata, creating transformation mechanisms for the information, evaluating their similarities, and finally and showing examples of the metadata in use. This report is broken up into five different sections detailing how each of these areas was completed.

Our work shows that it is possible to integrate both FMA and SNOMED information individually into an X3D scenegraph with a lossless transformation. It is theoretically possible that both FMA and SNOMED transformation could be combined together in one single mapping tool handling both data sources, but with current resources this was outside the scope of this contract. In addition, the separation of tools helps to illustrate and clarify their differences regarding structure and terminology.

The results of this project may constitute Recommended Practice for the Web3D Consortium's Medical Working Group and inform the development of semantically-integrated interactive 3D applications (i.e. anatomy browsers and imaging (DICOM) tool vendors). A common scenario would be to use these conventions to semantically 'tag' segmented anatomical structures for storage or delivery as an X3D environment.

The general nature of the conversion is to have a MetadataSet contain items with multiple attributes and to use a metadata string or integer for the information. The toolset we provide is based on the following simple premises, used as conventions:

1. MetadataSets refer to their sibling Transform node, where the object's shape geometry may be specified. A sibling Group node may be instantiated for parts or subdivisions of the referent object. This allows larger containing structures or anatomical systems to be easily accessible programmatically and additional detail accessible when needed
2. The MetadataSet is instantiated with its source specified as the reference field (e.g. FMA, SNOMED); its children are typically MetadataStrings specifying its attributes and its relationships to other entities in the source ontology
3. Unique identifier names of source entities (integers) are prepended with an 'm'. This allows result data to conform to the Web3D scene-graph identifier convention (DEF); to cross-reference corresponding entities in the scene-graph or to programmatically access named nodes, one must remove this first character ('m') and compare it with a MetadataSet's name="" attribute.
4. If source information is of type Integer, a MetadataInteger is instantiated
5. If source information is of type Boolean, a MetadataString is instantiated with a value of true or false.

The contents of the deliverable tarball implementing these conventions are listed in Appendix section

A. The tarball is available at : <http://snoid.sv.vt.edu/~anray2/x3dMetadata.tar.gz>

## Section 1. Obtain working copies of FMA / SNOMED

### 1.a FMA

We obtained information about FMA by going to their website, following their instructions of downloading the mysql data, installing the data into a local mysql database, installing Protege (an ontology viewer / editor), and installing PHPMyAdmin for investigating the mysql database via web pages and forms. The website for FMA and a set of instructions to duplicate our work on localhosts can be found in section A of the appendix.

An example PHPMyAdmin viewing account to the FMA mysql can be found online at:

<http://snoid.sv.vt.edu/npolysWWW/phpMyAdmin/index.php>  
user: web3d password: web3d

### 1.b SNOMED

Currently, there is very little information publicly accessible concerning SNOMED. The main source of information is <http://www.snomed.org> but it is currently in transition. One of the websites linked from the main site contains the XML Schema and an example SNOMED XML file. This can be found here:

<http://www.ihtsdo.org/our-standards/technical-documents/#c586>.

In the future more information and examples will hopefully be provided.

The main difficulty in achieving these tasks dealt with finding information about SNOMED and the system configuration of Protégé for FMA. Once this was finished, we moved to the next step of transforming the information from these packages into a format that could be instantiated into a X3D scenegraph in a lossless manner.

## Section 2. Determine a mapping between FMA -> X3D

FMA has information at many different levels. They are both theoretical and practical. At a theoretical level FMA is composed of body that has many different parts and each part has a parent / child relationship and several different pieces of information associated with it. A way to map this into X3D metadata would be to have a metadata set for each part, have metadata strings and integers to contain information about part, and to provide the names of the other parts that it is associated with. This will allow for the relationships and information in FMA to be represented.

The format of the information of FMA comes in two different forms. The first is raw SQL information (selected using a 'produce xml' option). The second are the forms that Protege can export. Protege itself allows for browsing of the FMA and exporting in several different.

### 2.a FMA via SQL Queries

MySQL can deliver database query results as XML. The transformation can therefore be done through an XSLT file. We implemented on and include it in the xslt folder in the tarball (FMA.xslt). The program xsltproc can be used to apply the stylesheet to output files.

An example FMA SQLrecord:

```
<FMA>
  <frame>26212</frame>
  <frame_type>6</frame_type>
  <slot>2002</slot>
  <facet>0</facet>
  <is_template>0</is_template>
  <value_index>0</value_index>
  <value_type>3</value_type>
  <short_value>Liver</short_value>
</FMA>
```

An example of how this can be transformed into target X3D metadata is as follows:

```
<X3D profile="Immersive" version="3.1" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
<head>
  <meta content="X3D metadata description of a FMA mysql xml document." name="description"/>
  <meta content="Written by a xslt template created by Andrew Ray." name="creator"/>
  <meta content="Program created on 9/23/2007." name="created"/>
  <meta content="Copyright (c) of generating author" name="rights"/>
  <meta content="FMA, SNOMED." name="subject"/>
</head><Scene>
<MetadataSet DEF="m123456" name="FMA-XML" reference="FMA">
  <MetadataInteger name="frame" value="26212"/>
  <MetadataInteger name="frame_type" value="6"/>
  <MetadataInteger name="slot" value="2002"/>
  <MetadataInteger name="facet" value="0"/>
  <MetadataInteger name="is_template" value="0"/>
  <MetadataInteger name="value_index" value="0"/>
  <MetadataInteger name="value_type" value="3"/>
  <MetadataString name="short_value" value="Liver"/>
</MetadataSet>
</Scene>
</X3D>
```

## 2.b FMA information via Protege

Another form of information that we have used is to produce X3D metadata in HTML output from Protege. Protege allows for the export of information in several different forms; however the only complete export format that contains all of the information in the database is the 'HTML output' option. All of the other options (RDF / OWL / XML are a few) require database lookups to completely fill in all of the information. An example Protege HTML output page can be found here or in the tarball provided with the report:

<http://snoid.sv.vt.edu/~anray2/parser/Upper+lobe+of+lung.html>

Part of the work for providing a pathway for FMA metadata to be incorporated into X3D was to build a digital library of FMA information. This was develop a C++ program to convert these html files into X3D metadata. From a standards prospective, an XSLT file is preferable to a C++ implementation, but due to the nature of the source data in this case (HTML), C++ was the best solution for transforming to X3D metadata. Instructions on how obtain, build and run this program are in Appendix section B.

An example of the metadata produced by this program is below.

```

<X3D profile="Immersive" version="3.1" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
<Scene>
<Transform DEF="Body">
<Group>
<MetadataSet DEF="m7334" name=" Upper_lobe_of_lung" reference="FMA">
  <MetadataString name="has inherent 3-D shape" value="true"/>
  <MetadataSet name="regional part" reference="FMA">
    <MetadataString name="SetName" reference="FMA" value="Bronchopulmonary_segment"/>
    <MetadataString name="SetName" reference="FMA" value="Bronchopulmonary_segment"/>
    <MetadataString name="SetName" reference="FMA" value="Bronchopulmonary_segment"/>
  </MetadataSet>
  <MetadataString name="regional part of" value="blank"/>
  <MetadataSet name="constitutional part" reference="FMA">
    <MetadataString name="SetName" reference="FMA" value="Parenchyma_of_lobe_of_lung"/>
    <MetadataString name="SetName" reference="FMA" value="Lobar_bronchial_tree"/>
    <MetadataString name="SetName" reference="FMA" value="Pleura_of_lobe_of_lung"/>
    <MetadataString name="SetName" reference="FMA" value="Neural_network_of_lobe_of_lung"/>
    <MetadataString name="SetName" reference="FMA" value="Vasculature_of_lobe_of_lung"/>
  </MetadataSet>
  <MetadataString name="dimension" value="3-dimension"/>
  <MetadataString name="has dimension" value="true"/>
  <MetadataString name="has boundary" value="true"/>
  <MetadataInteger name="FMAID" value="7334"/>
  <MetadataString name="Preferred name" value="Upper_lobe_of_lung"/>
  <MetadataSet name="Synonym" reference="FMA">
    <MetadataString name="SetName" reference="FMA" value="Lobus_superior"/>
    <MetadataString name="SetName" reference="FMA" value="Superior_lobe_of_lung"/>
  </MetadataSet>
  <MetadataSet name="part" reference="FMA">
    <MetadataString name="SetName" reference="FMA" value="Parenchyma_of_lobe_of_lung"/>
    <MetadataString name="SetName" reference="FMA" value="Lobar_bronchial_tree"/>
    <MetadataString name="SetName" reference="FMA" value="Pleura_of_lobe_of_lung"/>
    <MetadataString name="SetName" reference="FMA" value="Bronchopulmonary_segment"/>
    <MetadataString name="SetName" reference="FMA" value="Bronchopulmonary_segment"/>
    <MetadataString name="SetName" reference="FMA" value="Bronchopulmonary_segment"/>
  </MetadataSet>
  <MetadataString name="part of" value="Lobular_organ"/>
  <MetadataString name="Non-English equivalent" value="Lobus_superior_pulmonis"/>
  <MetadataString name="constitutional part of" value="blank"/>
</MetadataSet>
</Group>
</Transform>
</Scene>
</X3D>

```

### Section 3. Determine a mapping between SNOMED -> X3D

SNOMED is much simpler to convert into X3D metadata than FMA. SNOMED already comes described by way of xml tools, so converting it to X3D metadata is simply transforming 'styling' the XML file. This is done via a XSLT. The instructions for how to obtain and run this file are in Appendix section C.

We generated documentation for the SNOMED CT Schema with xnsdoc:

<http://snoid.sv.vt.edu/~anray2/snoMed/>

The complete input and output for SNOMED knowledgebase is rather extensive so only a small part will be used to explain the strategy for creating X3D metadata out of SNOMED information. In general, the SNOMED data is grouped into different elements that have several different attributes in each tag. The general strategy for converting this into X3D metadata was to create an overarching metadata set that contained the same structure of the existing format by using sets with the attribute name that holds the old elements name. After this, all of the attributes are incorporated as metadata strings with name fields being the attribute and the value field being the value of the attribute.

An example of SNOMED input:

```
<concepts>
  <concept conceptId="369445005" conceptStatus="0" fullySpecifiedName="Chronic proctocolitis (disorder)"
    ctv3id="XUU7a" snomedId="D5-45285" isPrimitive="1">
    <descriptions>
      <description descriptionId=" 774149015 term="Chronic proctocolitis"
        descriptionType="3" initialCapitalStatus="0" languageCode="en"
        descriptionStatus="0">
        <history releaseVersion="20020131" changeType="0" reason="null"/>
      </description>
    </descriptions>
    <relationshipSet>
      <relationship relationshipId="2398318025" relationshipType="24611200006"
        conceptId2="3853150009" characteristicType="1" refinability="1"/>
      <relationshipGroup>
        <relationship relationshipId="1082140022"
          relationshipType="116676008" conceptId2="23583003"
          characteristicType="0" refinability="2"/>
        <relationship relationshipId="1082141021" relationshipType="363698007"
          conceptId2="34402009" characteristicType="0" refinability="2"/>
      </relationshipGroup>
    </relationshipSet>
  </concept>
</concepts>
```

Although there are several integer examples in this example, all of them are represented as strings in our transformation because if the value is not valid, it may be set to "null." An example of the transformed XML into X3D metadata is:

```

<MetadataSet DEF="m123456" name="SNOMED-XML" reference="SNOMED">
<MetadataSet name="concepts">
  <MetadataSet name="Concept " reference="SNOMED">
    <MetadataString name="conceptId" value="369445005"/>
    <MetadataString name="conceptStatus" value="0"/>
    <MetadataString name="fullySpecifiedName" value="Chronic proctocolitis (disorder)"/>
    <MetadataString name="ctv3id" value="XUU7a"/>
    <MetadataString name="snomedId" value="D5-45285"/>
    <MetadataString name="isPrimitive" value="1"/>
    <MetadataSet name="Description" reference="SNOMED">
      <MetadataString name="descriptionId" value="774149015"/>
      <MetadataString name="term" value="Chronic proctocolitis (disorder)"/>
      <MetadataString name="descriptionType" value="3"/>
      <MetadataString name="initialCapitalStatus" value="0"/>
      <MetadataString name="languageCode" value="en"/>
      <MetadataString name="descriptionStatus" value="0"/>
    </MetadataSet>
    <MetadataSet name="History" reference="SNOMED">
      <MetadataString name="releaseVersion" value="20020131"/>
      <MetadataString name="changeType" value="0"/>
      <MetadataString name="reason" value="null"/>
    </MetadataSet>
    <MetadataSet name="relationshipSet">
      <MetadataSet name="relationship" reference="SNOMED">
        <MetadataString name="relationshipId" value="2398318025"/>
        <MetadataString name="relationshipType" value="246100006"/>
        <MetadataString name="conceptId2" value="385315009"/>
        <MetadataString name="characteristicType" value="1"/>
        <MetadataString name="refinability" value="1"/>
      </MetadataSet>
      <MetadataSet name="relationship" reference="SNOMED">
        <MetadataString name="relationshipId" value="2398319022"/>
        <MetadataString name="relationshipType" value="246100006"/>
        <MetadataString name="conceptId2" value="61751001"/>
        <MetadataString name="characteristicType" value="1"/>
        <MetadataString name="refinability" value="1"/>
      </MetadataSet>
      <MetadataSet name="relationshipGroup">
        <MetadataSet name="relationship" reference="SNOMED">
          <MetadataString name="relationshipId" value="1082140022"/>
          <MetadataString name="relationshipType" value="116676008"/>
          <MetadataString name="conceptId2" value="23583003"/>
          <MetadataString name="characteristicType" value="0"/>
          <MetadataString name="refinability" value="2"/>
        </MetadataSet>
        <MetadataSet name="relationship" reference="SNOMED">
          <MetadataString name="relationshipId" value="1082141021"/>
          <MetadataString name="relationshipType" value="363698007"/>
          <MetadataString name="conceptId2" value="34402009"/>
          <MetadataString name="characteristicType" value="0"/>
          <MetadataString name="refinability" value="2"/>
        </MetadataSet>
      <!--End relationshipGroup tag -->
    </MetadataSet>
  <!--End relationshipSet tag -->
</MetadataSet>
<!--End concept tag -->
</MetadataSet>
<!--End concepts tag -->
</MetadataSet>

```

```
<!--Final closing tag -->
</MetadataSet>
```

The SNOMED XSLT file that converts the mapping is quite complex (>600 loc). For performance and future maintenance, a C++ program would be a much better fit. Due to the original work plan an XSLT file was produced to meet the deliverable, but if the SNOMED standard is changed, it may be simpler to write a C++ program instead of modifying the XSLT file.

## **Section 4. Determine a mapping between FMA / SNOMED**

SNOMED has a much richer design space than FMA does. However, it has the notion of relationships can encompass the part / partOf relationship that FMA uses. Thus, it is theoretically possible to map the two different ontologies. An more thorough investigation into how the two ontologies can correspond can be found here:

<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1839313>

However, this requires foresight into designing the SNOMED data because FMA is completely fixed and cannot be changed, whereas SNOMED can be manipulated when particular data files are being created. If a SNOMED file is created that contains the correct names and identification numbers for specific part relationships in FMA, then it would be possible to integrate both SNOMED and FMA metadata. The other option would be to include a Grouping MetadataSet that would contain a FMA metadataset and a SNOMED metadataset. If this were done then the existing conversion tools could be modified slightly to accommodate this info.

## **Section 5. Example usage of metadata**

There are many possibilities for applying this work in X3D medical applications. Some specifics of the node structure in the delivery scene-graph will depend on the application requirements. The tools and recommended practice we provide can be widely adapted for these needs. For example in X3D, the interface for an anatomy browser could be created with menus driving the visibility or highlights of shapes with a Switch node; referent shapes can also be Inlined. Alternatively, a program could walk the scene-graph, apply some logic, and display specific metadata information as annotations or labels.

We include an example of the FMA and SNOMED information from this report in the following files:

- `body_ifs.x3d` – An IndexedFaceSet from NIST constituting an androgynous human skin
- `med_example.x3d` – An simple MedX3D scene-graph structure with shapes and metadata conforming to our recommended practice; it includes a mix of FMA and SNOMED information for skin, lung, liver, colon
- `viewer2.x3d` – adds simple interactive interface to some of the data from `med_example.x3d` and demonstrates a simplistic anatomy / ontology viewer

# Appendix

## *Section A: contents of the tarball*

<http://snoid.sv.vt.edu/~anray2/x3dMetadata.tar.gz>

```
x3dMetadata
  finalReport.doc
  fmaHtmlParser
    header.xml
    README.txt
    script1
    Serosa+of+right+hemiliver.html
    Upper+lobe+of+lung.html
    parse.cpp
    script0
    script2
    Serosa_of_right_hemiliver.xml
    Upper_lobe_of_lung.xml
  html_out.tar.bz2
  index.html
  README.txt
  xslt
    fmaOut.xml
    FMA.xml
    FMA.xslt
    README.txt
    sct_xml_cps_sample.xml
    snoModified.xml
    snoOut.xml
    SNO.xslt
```

## *Section B: FMA Setup*

This section deals with where to go to obtain the resources necessary to transform FMA information into X3D Metadata. It goes over how to obtain the database, how to import the database into a mysql database, how to install Protege (ontology program that talks with the database), how to get usable data out of Protege, and how to transform this data into X3D Metadata.

To obtain the FMA database, you must agree to the license found here:

[http://sig.biostr.washington.edu/projects/fma/license\\_faq.html](http://sig.biostr.washington.edu/projects/fma/license_faq.html)

Once this is done you have a somewhat involved process for getting the FMA to work. The first part is obtaining the database, next is loading the database into MYSQL. Next is downloading Protege, installing the Protege extensions, and configuring protege to talk to mysql. Lastly, you have to export the FMA project from protege into the html output form.

### **FMA database:**

To obtain dump file: "Please use the username "opensource" and the password "fma2007" when downloading the FMA dump file. We apologize for the inconvenient download procedure, but we are currently investigating better methods."

[http://fma.biostr.washington.edu/latest\\_version](http://fma.biostr.washington.edu/latest_version)

This is simply a file which holds the FMA in a format that mysql can handle. Once you have downloaded the file you can install it into MYSQL using the instructions on this website:

[http://sig.biostr.washington.edu/projects/fm/FMA\\_Release/FMA\\_instructions.html](http://sig.biostr.washington.edu/projects/fm/FMA_Release/FMA_instructions.html)

Once the database is installed and working then you need to install a program called Protege that knows the format for the database (it isn't setup the way you think it is). This program can then output the FMA in a saner format for processing.

### **Protege Instructions:**

[http://sig.biostr.washington.edu/projects/fm/FMA\\_Release/setup-protege.html](http://sig.biostr.washington.edu/projects/fm/FMA_Release/setup-protege.html)

*If these instructions do not work please go to this URL and follow the instructions that the FMA developers provide:*

[http://depts.washington.edu/ventures/UW\\_Technology/Express\\_Licenses/FMA.php](http://depts.washington.edu/ventures/UW_Technology/Express_Licenses/FMA.php)

*As a last resort, here is the FMA homepage, find out how to obtain it from there:*

<http://sig.biostr.washington.edu/projects/fm/FAQs.html>

### **Generating HTML from Protege:**

Once all of this working, the steps to producing X3D metadata are rather simple. The first is start Protege, load the FMA project file, click file, click 'export to format', then click 'HTML'. Specify the directory for the html to be generated and then wait. This process can literally take several days. This process will generate approximately 1.1 gb of data.

A bzip'd tarball that contains the HTML generated from Protege if you wish to save yourself the work of generating it can be found in the release tarball or at the following website. This is a 12 MB download that expands to 1.1 GB.

[http://snoid.sv.vt.edu/~anray2/html\\_out.tar.bz2](http://snoid.sv.vt.edu/~anray2/html_out.tar.bz2)

### **Transforming HTML into X3D Metadata:**

The requirements for this program are a linux system that can compile C++ code, and can run sed on script files. The parser can found in the release tarball in the directory **fmaHtmlParser** or be downloaded at:

<http://snoid.sv.vt.edu/~anray2/parser.tar.gz> .

Building the parser can be done via typing make parse inside of the parser directory. There are two example html files in this directory, and two xml files that show the output of the parser.

The program runs by giving it a single html file to convert. This program will then convert the html file into an xml file named by a sanitized version of the html title (sanitized to only contain the FMA name delimited by '\_'s not spaces).

### **Example:**

```
./parse Liver.html
```

### **Produces:**

```
Liver.xml
```

### ***Section C: Converting SNOMED to X3Dmetadata***

The first step that has to be done is to install a XSLT processor. The one used when creating this program can be found here: <http://xmlsoft.org/XSLT/xsltproc2.html>. Once the processor is installed and working then the next step is to download the SNOMED to X3D metadata XSLT file. The file can be found at <http://snoid.sv.vt.edu/~anray2/xslt/SNO.xslt>. The next step is to strip the following lines out of your XML file:

```
<snomedCt xmlns="urn:snomed-org/sct" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:snomed-org/sctSchema\SnomedCt.xsd">
</snomedCt>
```

This is due to the limitations of xsltproc, it cannot parse this tag. The next necessary addition is to have the xml file begin with <test> and end with </test>. Once this is done then you can transform SNOMED xml files to X3D metadata files by typing:

```
xsltproc SNO.xslt yourxmlfilehere.xml
```