Constraint Definition Designer

CDD

Constraint Definition Designer

Release 2.3.0

Timothy W. Cook

2012-08-01

THIS DOCUMENT IS UNDER REWRITE AT THIS TIME!!!!
Please refer to the MLHIM2 2.3.0 specifications documents for the model specific information. This guide is to just give a general idea of how to use the Xmind template. Also, there is a short section on how to use the CDD tool to create the skeleton CCD with metadata. Please refer questions to the mlhim-owners mailing list: https://launchpad.net/~mlhim-owners

Contents

CDD Tool – ccd.py

The source code for this tool is in the 'src' directory of the CDD package. The packaged tool can also be found on the HKCR website at: http://www.hkcr.net/tools

At the time of this writing (2012-08-01) the tool can accept the metadata for a CCD and create a shell CCD (XML Schema).

Start the program with: $python cdd.py

In the tree in the left panel, select metadata and fill in the form. These metadata items are a subset of the Dublin Core Metadata Initiative (DCMI) elements for resource descriptions. For details see: http://dublincore.org/

Most fo the items are self explanatory. The Subject should be made up of semi-colon spearated entries of MeSH terms. http://www.nlm.nih.gov/mesh/

Please send all questions to the MLHIM Owners list at: https://launchpad.net/~mlhim-owners

Regards,

Tim Cook

Constraint Definitions

Constraint Definitions (CDs) are used to add knowledge model information to applications built using a specific Reference Model. In this document we will be focusing on the Multi-Level Healthcare Information Model (MLHIM). In MLHIM we are using the mind-mapping tool XMind, to create the conceptual outline of the expert knowledge model. This visual model is managed via the Healthcare Knowledge Component Repository (HKCR). HKCR is an application developed on the Plone content management system (CMS). It provides all of the facilities of an enterprise grade CMS with the addition of conversion, editing and validation specifically for MLHIM constraint definitions.

The visual representation, when complete, is converted to a constraint definition model (CDM) inside the HKCR. This CDM is expressed as a XML Schema defining cardinality and existence constraints against the reference model. The CDM is a data definition for a specific concept. It is therefore, generically called a Concept Constraint Definition (CCD).

Creating the Visual CCD

Xmind is a mind map tool that can be used in many different ways. Often used as a brainstorming tool to keep track of linked concepts. In the specific case of creating a CCD. We have a template CCD-x.y.xmt (where x.y represents the latest version). This template is a functional approximation of the MLHIM Reference Model.

From the XMind menu select: File->New->New From Template (or use the shortcut keys CTL+ALT+N) and choose your copy of the CCD template file.

At this point you may save your CCD and name it as you wish. We are going to be doing a demonstration of converting an openEHR archetype[1] to MLHIM CCD. Since we are using the openEHR-EHR-OBSERVATION.apgar.v1 we will name our CCD the same. So we end up with the filename openEHR-EHR-OBSERVATION.apgar.v1.xmind

Exploring the Template

Once the template is open you will see along the bottom of the main window, a list of tabs.

CCD

- This page has the basic components of the CCD. This is where you will actually be doing the CCD layout.

DATATYPES

- This page contains definitions of the reference model classes in the datatypes package.

DATASTRUCTURES

- This page contains definitions of the reference model classes in the datastructures package.

EHR

- This page contains definitions of the reference model classes in the ehr package.

COMMON

- This page contains definitions of the reference model classes in the common package.

DEMOGRAPHIC

- This page contains definitions of the reference model classes in the demographic package.

LEGEND

- This page is the legend for color coding and icons used throughout the template. See the notes box for more information about the selected node.

CCD Sheet

This sheet has the CCD main idea with two child nodes; definition and metadata. The basic concept is to complete the definition as a structured concept.

---

[1] We recommend using the Archetype Workbench while examining the ADL

Defining the Concept

Using our example archetype openEHR-EHR-OBSERVATION.apgar.v1 we see that the definition component is an OBSERVATION. So, go to the EHR tab and select the OBSERVATION class and press CTL+C to copy it to the clipboard. Go back to the CCD tab and select definition. Press CTL+V to paste the OBSERVATION. It will now be a child node of definition. Click on the expansion symbol (plus sign in a circle) on OBSERVATION and should now have a display like Figure 0.1.

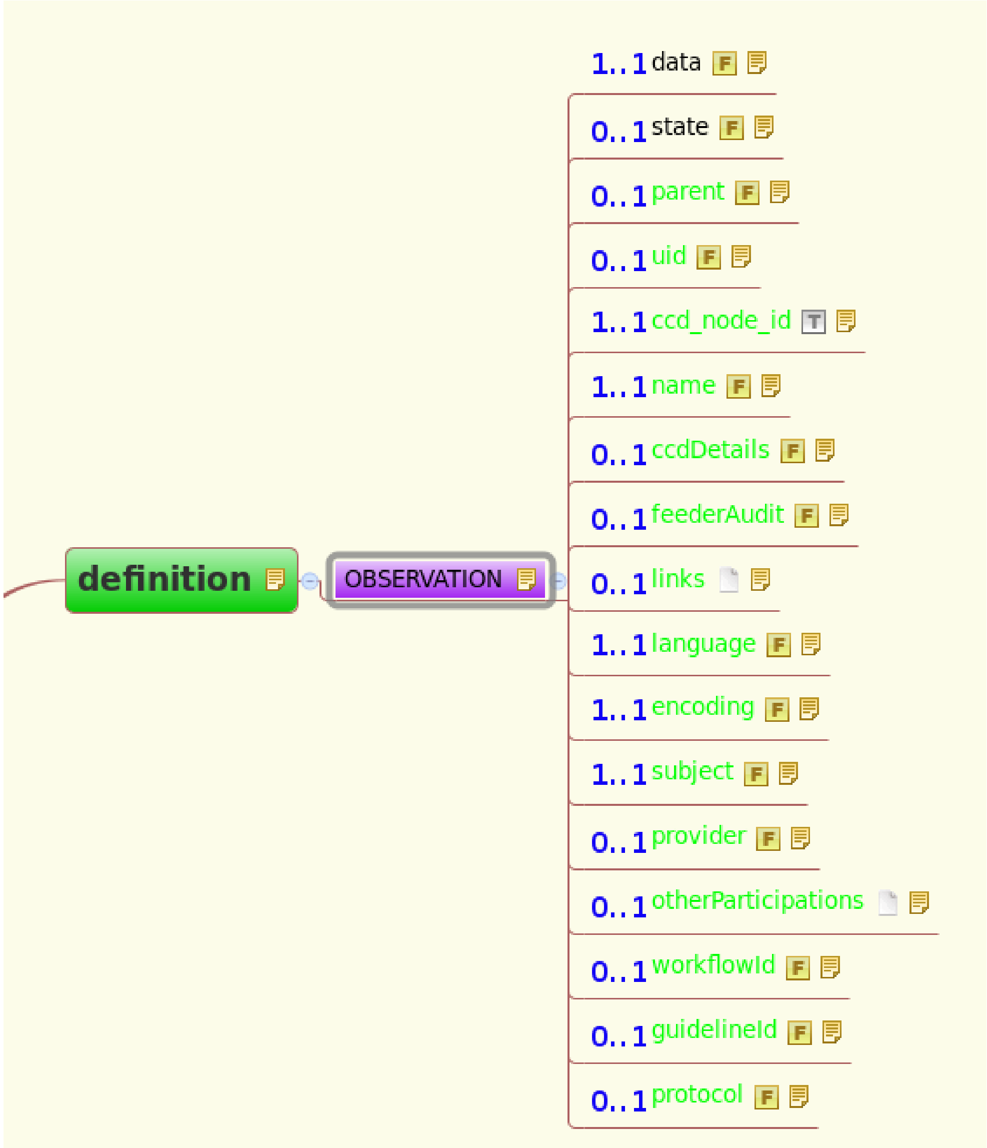

Figure 1: Definition w/OBSERVATION

Notice that the data attribute is required by the reference model. In the archetype definition it calls for a HISTORY. Go to the DATASTRUCTURES sheet and select the HISTORY class and copy it. On the CCD sheet, select the OBSERVATION.data attribute and paste the HISTORY class.

The archetype HISTORY.events attribute is constrained to 1..* (from its original 0..1) of course this should not be possible. But the reference model is actually refering to zero to one List of EVENTS. This is an inconsistenency throughout the models. But the meaning is to require at least one EVENT with no upper bound on the number of EVENTS. For our purposes, in conversion, we will change the 0..1 to 1..* by clicking on the icon and selecting one2many.png. So now copy paste a POINT_EVENT to HISTORY.events. Now you may notice that in the ADL, there is a POINT_EVENT.offset defined. This is confusing because EVENT.offset in the reference model is defined as a class function, not an attribute. But the ADL leaves out required attributes in the reference model in openEHR. In MLHIM we use the approach of being explicit as opposed to implicit. We will therefore define the HISTORY.origin attribute as required and trust that the implementation will correctly calculate the POINT_EVENT.offset. The ADL specifies that the returned DV_DURATION must be 1 minute. However, that is not in conformance with the reference model and since this should be a calculated value.

Go to the DATATYPES sheet and select and copy the DV_DATETIME class. Now select (on the CCD sheet) HISTORY.origin and paste the DV_DATETIME. Also paste the DV_DATETIME in the POINT_EVENT.time attribute. Before we go furhter with the ADL. There are two more required attributes in HISTORY. The ccd_node_id and name. The ccd_node_id is inherited from the LOCATABLE class and it will always be completed by the HKCR upon publication of the CCD. The HISTORY.name attribute does require a DV_TEXT class. So from the DATATYPES sheet, copy and paste a DV_TEXT here.

Now according to the ADL we will copy/paste an ITEM_LIST from the DATASTRUCTURES sheet to the POINT_EVENT.data attribute. The ADL sets the cardinality to 1..6. However, the model explicitly expresses six ELEMENTS. So we will copy ELEMENT from the DATASTRUCTURES sheet and paste six copies of it into the ITEM_LIST.items attribute. These ELEMENTS represent one set of measurements taken at one point in time.

- Respiratory effort
- Heart Rate
- Muscle tone
- Reflex irritability
- Skin colour
- Total

Each of the first five ELEMENT data values (dv attribute) may contain an integer 0, 1 or 2. The sixth ELEMENT.dv contains the sum of the first five. Each of these set of choices are created as an OntologyEntry. We will model these choices as a hash: {’Absent’:0,’Weak or irregular’:1,’Normal’:2} this will be the contents of the OntologyEntry. The name will be the comment from the ADL; Respiratory Effort. See Figure 0.2

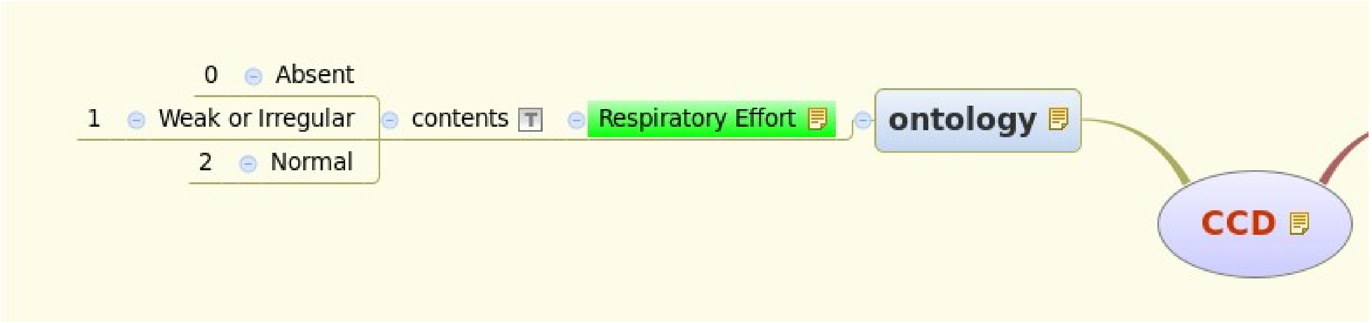

Figure 2: Ontology Entry

Now we will create a hyperlink from the first ELEMENT.dv to this OntologyEntry. We will also change the cardinality of ELEMENT.dv to 1..1. Repeat this process for each of the other four conditions. Remove the hyperlink from the last ELEMENT.dv and enter into the notes that this element contains the total of the previous five.

Now collapse the tree back to the POINT_EVENT. Select PIONT_EVENT and copy it using CTL+C. Select HISTORY.events and paste four copies of POINT_EVENT. You should now have five POINT_EVENT entries in the HISTORY.events attribute. For each POINT_EVENT.name attribute create a child node for the intervals; 1 Minute, 2 Minutes, 3 Minutes, 5 Minutes, 10 Minutes. Note that additional POINT_EVENTS are allowed and the application should allow for them.

On the OBSERVATION.protocol attribute change the cardinality to 1..1 and add an ITEM_LIST. Set the ITEM_LIST.items attribute to 0..* and add an ELEMENT. On the ELEMNT.dv attribute add a DV_TEXT.

When these are complete and you are certain that you do not need to make further changes you can delete all of the tabs except the CCD tab. You can also delete the OntologyEntry class from the CCD sheet.