Background

The *Concept Constraint Designer*[1] (CDD) is to be a tool that provides an easy to use graphical editing environment for creating *Concept Constraint Definitions* (CCD) based on the MLHIM Reference Model.

The MLHIM Reference Model is formally expressed in the MLHIM Reference Manual and on the MLHIM XMind template. The MLHIM reference implementation is expressed as XML Schemas.

A CCD is a set of constraints expressed in XML Schema based on the MLHIM reference implementation; that describes a specific healthcare or administrative concept.

These CCDs are therefore portable descriptions of a concept that can be exchanged between applications whose core source code is based on the MLHIM reference model. This approach is known as multi-level modeling and allows applications to semantically exchange information irregardless of their database design or type of persistence model in general.

Goal

The goal of the CDD is to provide an easy to use graphical interface, similar to mind-mapping software, by a healthcare domain expert and export it as a CCD (XML Schema concept description).

Current Status

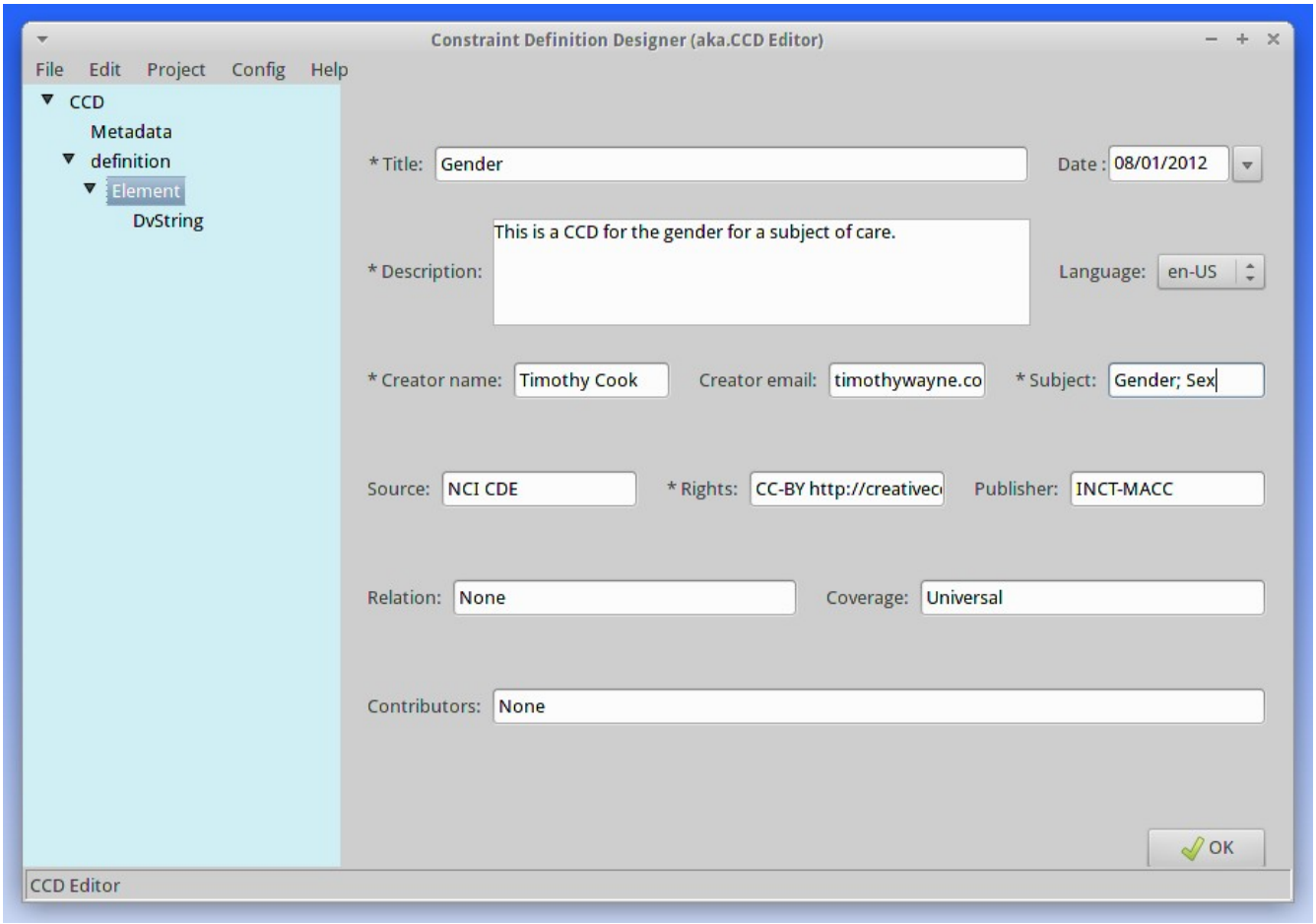
At this point (see the date in the page footer), the CDD is a XMind template that represents the MLHIM reference model. Along with a beginning CDD application that can generate a basic CCD skeleton.

The Xmind template is used by the domain experts in this way:

1) Save a copy of the template to a new file.
2) Copy and paste the elements on the sheets into the CCD->description node.
3) Set/define all the constraints as needed for a given concept.

1 See the CDD Users Manual for detailed information.

At this point the domain expert has something that looks like the design_example.xmind file. The CDD (cdd.py) can produce the CCD skeleton with the metadata.



The 'tree' in this graphic is not correct so ignore the Element->DvString portions.

Once the metadata is entered the user will click the Ok button. The metadata is saved in a structure (a tree, dictionary of dictionaries?) and pickled to a file that the user chooses the filename with a .cdd extension. This is a Project file, so the Project->New menu selection should clear the metadata form. The Configuration form should have a button to store the metadata information as default metadata information. It should not be saved every time w/o user confirmation. Help->Users Guide should launch docs/cdd-manual.pdf

The left panel is now called 'the tree' and the right panel is called 'the form'. When specifying an action/dialog, etc. I will use names from the wxPython Demo application.

For button background colors use the associated color of the page in the Xmind template. Example; Cluster and Element buttons will be Orange and CareEntry, AdminEntry, etc will be Purple. Leave the form background as a light grey.

The 'definition' selection in the tree should behave as follows:

The form should have five radio buttons labeled:

Care Entry

Admin Entry

Demographic Entry

Cluster

Element

(the entire area of the label should be active for user selection)

Use an Ok button to confirm selection.

When the user confirms the definition selection, clear the form and fill it with the **Form Type** based on the selection. Also add the selection as a child of definition in the tree.

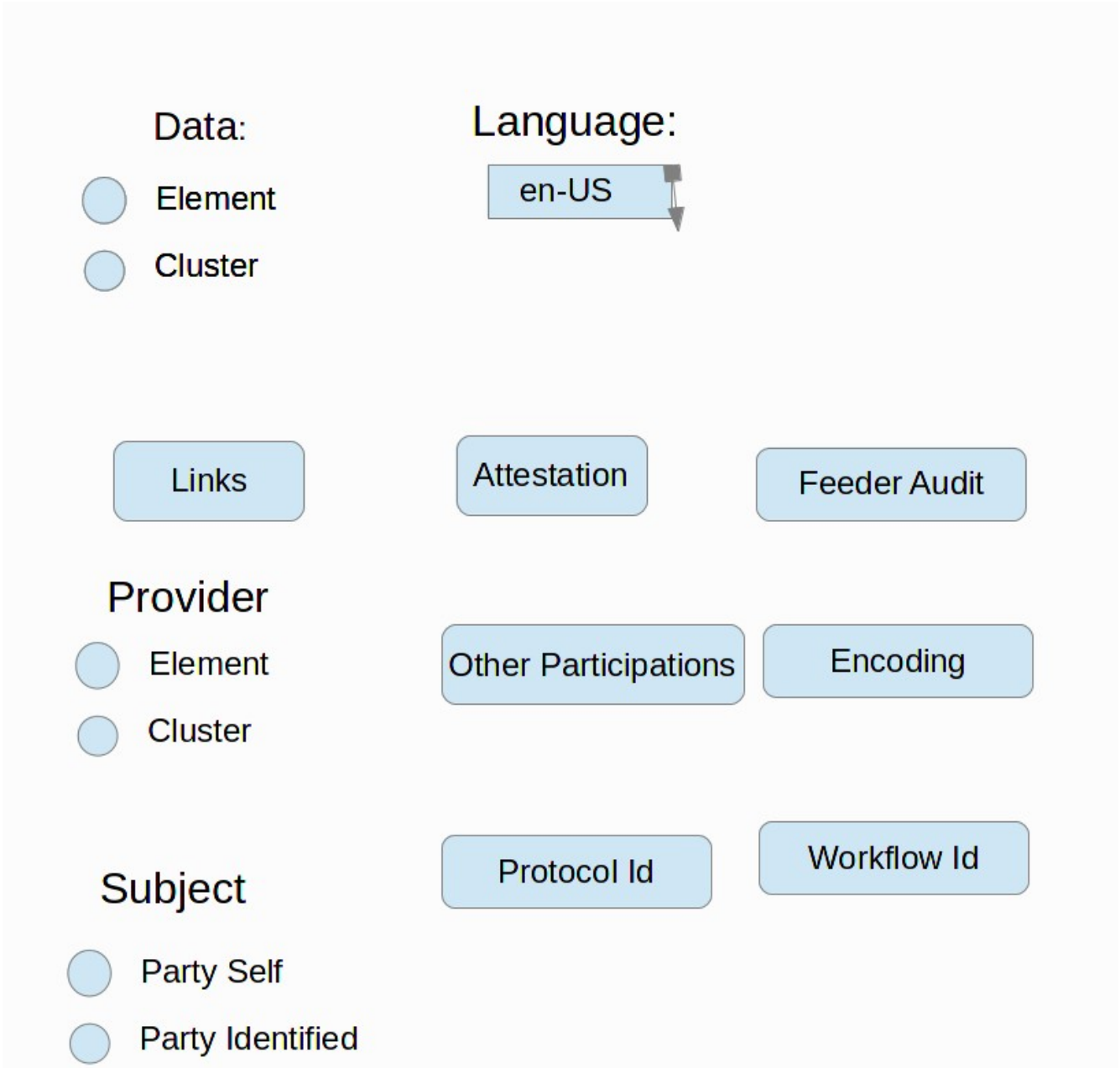
Form Types:

Element - a SingleChoiceDialog with all of the datatypes as options.

Cluster – Radio buttons for Cluster and Element

xxxEntry – These are the more complex forms that require all of the attributes from the class (complexType elements) to be on the form. In the example I'll walk through each of these.

In the example we add a CareEntry child of definition and open a Care Entry form. Each form in the form area will be a notebook tab. The tab name should be the kind of form it is; i.e. CareEntry, DvCodedString, etc. This form should look something like this:


Data:
Element
Cluster
Language:
en-US
Links
Attestation
Feeder Audit
Provider
Element
Cluster
Other Participations
Encoding
Subject
Party Self
Party Identified
Protocol Id
Workflow Id

Language is just a pulldown and allows selecting the language. It is mandatory but should default to whatever was selected in the metadata.

The Links button opens a new tab (DvURI form) with a TextCtrl labeled 'data_name'. A section labeled 'dv' will have an EditableListBox labeled'enumeration', A TextCtrl labeled 'pattern', two TextCtrls for min_length and max_length.

The other buttons work in a similar manner. Check the 'type' for each one to see what kind of form needs to be opened (built) in each new tab.

For Data, if the Element button is selected it should open a tab with a datatype select box. Then when a datatype is selected you can reuse that tab for the datatype form.

Once you have created a CCD, you may upload it to HKCR to the CCD Queue
http://www.hkcr.net/ccd-queue

If you have any questions please contact us via the MLHIMOwners mailing list:

https://launchpad.net/~mlhim-owners