GDL EDITOR

USER MANUAL (v0.93)

FUNDED BY: Cambio Healthcare Systems (http://www.cambio.se)

INTRODUCTION

The GDL editor is multiplatform application that allows users to create, edit and run GDL files. GDL is a formal language designed to represent clinical knowledge for decision support. It is designed to be natural language- and reference terminology- agnostic by leveraging the designs of openEHR Reference Model and Archetype Model. This tool provides an editing and testing environment capable of generating forms based on the elements defined in the GDL.

USER INTERFACE

BASIC LAYOUT

The GDL editor is divided in eight tabs:

- **Description:** basic information about the **guideline**.
- **Definitions:** references to the archetypes used in the rules and preconditions.
- **Rule list:** allows managing of all the rules inside the guideline.
- **Preconditions:** a list of conditions that have to be fulfilled before any rule is executed.
- **Terminology**: translations for each one of the terms used in the guidelines.
- **Binding:** mapping of the local codes used in the guideline to external terminologies.
- **GDL:** the output of the editor (in GDL format).
- **HTML:** the output of the editor (in HTML format).
- **Implementation view:** the output of the editor in a rule engine format (JBoss Drools in this case).

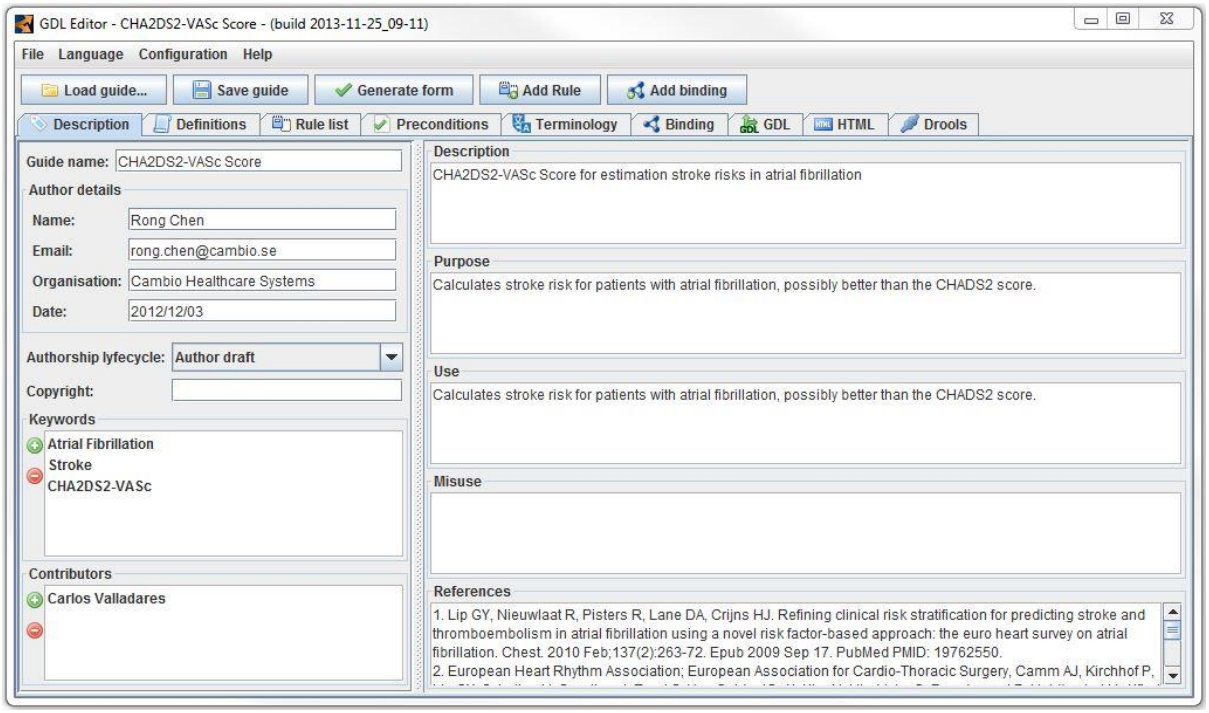

Figure 1: GDL description

MENUBAR

There are three groups of options in the menu bar:

- **File:** basic operations regarding the file system.
  - **Create new guideline**
  - **Load guideline**
  - **Save guideline**
  - **Import**
    - **Import archetype:** Loads an archetype into the editor's repository
    - **Import templat**e: Loads a template into the editor's repository
  - **Export**
    - **Export to object:** Saves the compiled guideline (drools format in this case)
    - **Export to HTML:** Saves the guideline in HTML format

- **Language:** manages the language settings of the guideline and the GDL editor.
  - **Add language:** creates a new language definition for the guideline being edited.

- **Configuration:** change editor settings.
  - **Repositories:** edit the folder location of archetypes and templates (requires reboot).
  - **Current date/time:** set the timestamp the rule engine will use as current time (by default system's clock is used).
  - **Language:** change the editor's language (current version supports only English and Spanish).

DEFINITIONS

Definitions establish a link between the archetype elements and the terms used in our guideline. All definitions of the guideline can be found at the 'Definitions' tab, and can be created either from here or directly from the preconditions /conditions/actions panels.

To create a new definition from the 'Definitions' tab (Figure 2), just drag and drop (or double click) the definitions (on the right side) you want to insert. All editable components of each definition will be displayed as a link (blue and underlined). To change its value, just click on it. To comment/uncomment a definition you will have to click on the first button (green if active, gray if inactive) right next to the name of the definition. To delete a definition, click on the second button (red).

> *Commenting definitions, conditions, actions and rules is useful for debugging the guideline, but keep in mind that commented content is not saved*

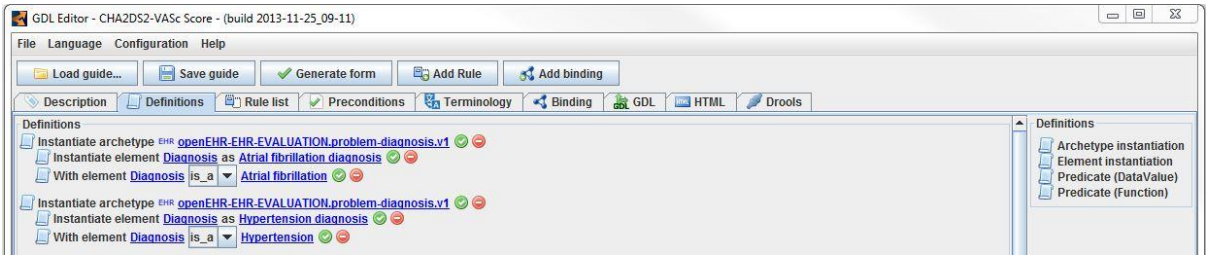

Figure 2: Managing definitions

GDL currently supports four types of definitions:

- **Archetype instantiation:** creates a reference to an archetype or a template. For each instantiation we will have to define three parameters:
    - **Domain:** there are three possible values (EHR / CDS / ANY) (1, Figure 3). See the GDL specification for more information about each one of them.
    - **Archetype/Template:** a list with the registered archetypes and templates will be displayed (2).

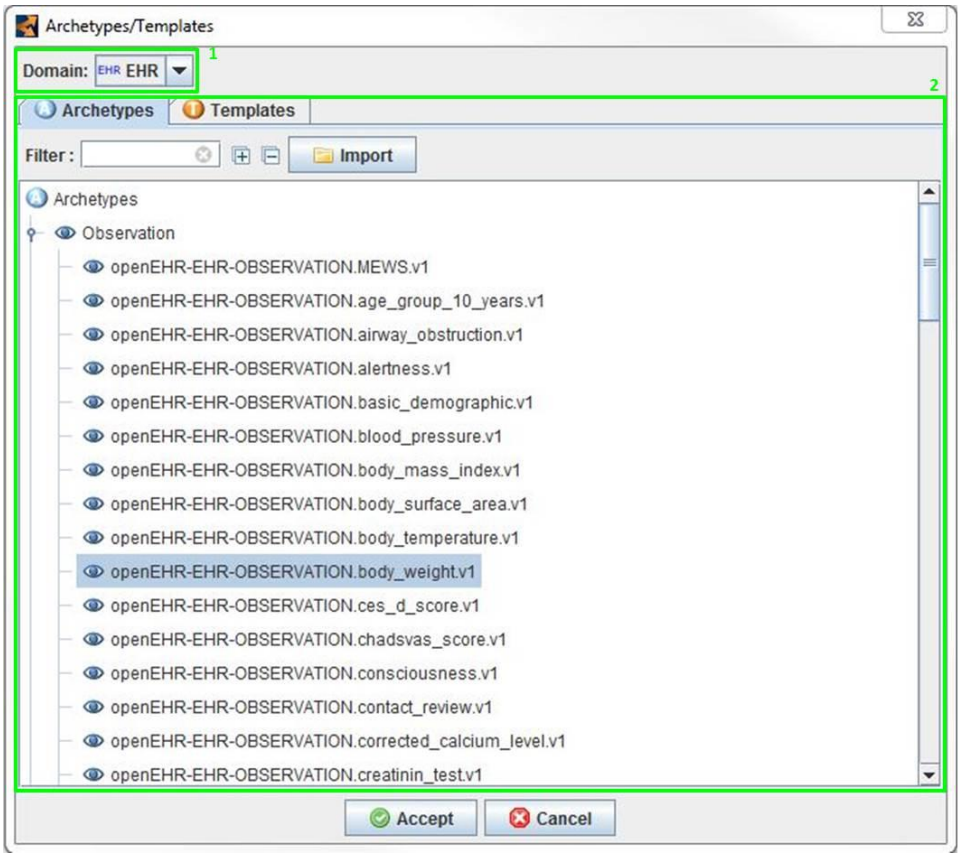

Figure 3: Choose archetype dialog

- **Element instantiation:** creates a reference to an element inside the archetype or template. It has to be placed inside an archetype instantiation.

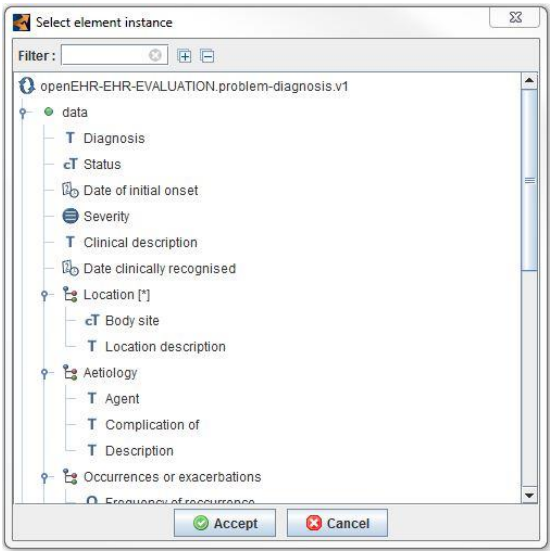

Figure 4: Choose element instance dialog

- **Predicate (DataValue):** defines a constraint for the archetype instance. It has to be placed inside an archetype instantiation.
- **Predicate (Function):** adds constrains to elements defined by using aggregate functions.

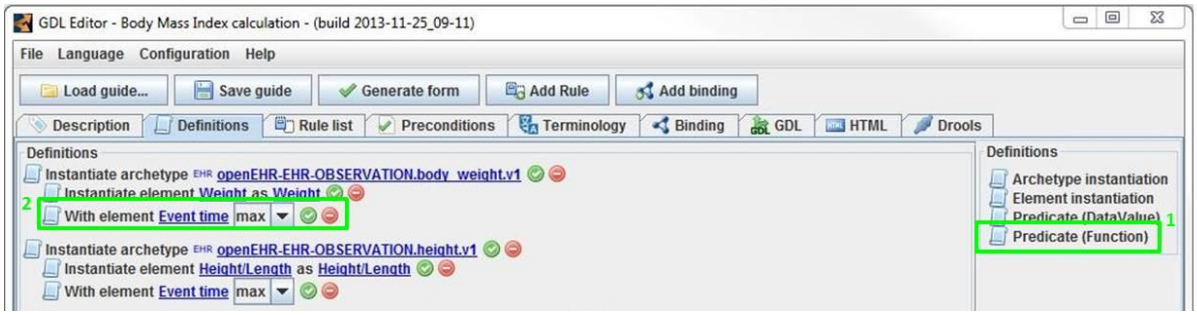

Figure 5: Predicate function

> *Definitions, conditions and actions can be added by dragging them into the main panel or simply by double clicking on them.*

RULE LIST

In this tab we will be able to manage all the rules in the guideline. Each rule contains a set of conditions and actions (see RULE EDITING). To enter a rule, just click on its name.

The managing of rules is very similar to the definitions. To add a new rule use the 'Add rule button' situated on the editor's toolbar. To edit a rule's name use the pencil icon.

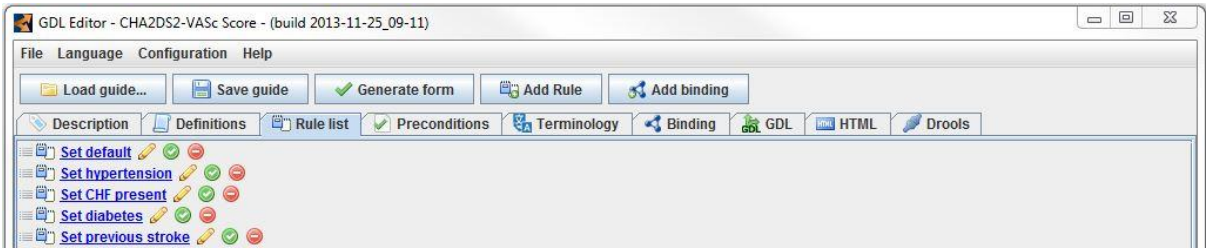

Figure 6: Rule list

RULE EDITING

When a rule is opened, the rule editor will be displayed. The upper part shows the conditions needed for the rule to execute, the bottom part contains the actions that will take place once the rule is activated (see Figure 8). Most of the actions and conditions will refer to an element instance which can be previously defined on the DEFINITIONS section or directly created from the rule editor. In the second case, when selecting an element instance form a condition or action, a dialog for selecting/defining element instances will be displayed. This dialog will allow us to select an already defined element instance (1, Figure 7), an element instance from an already defined archetype instance (2) or add a new archetype instance (3).

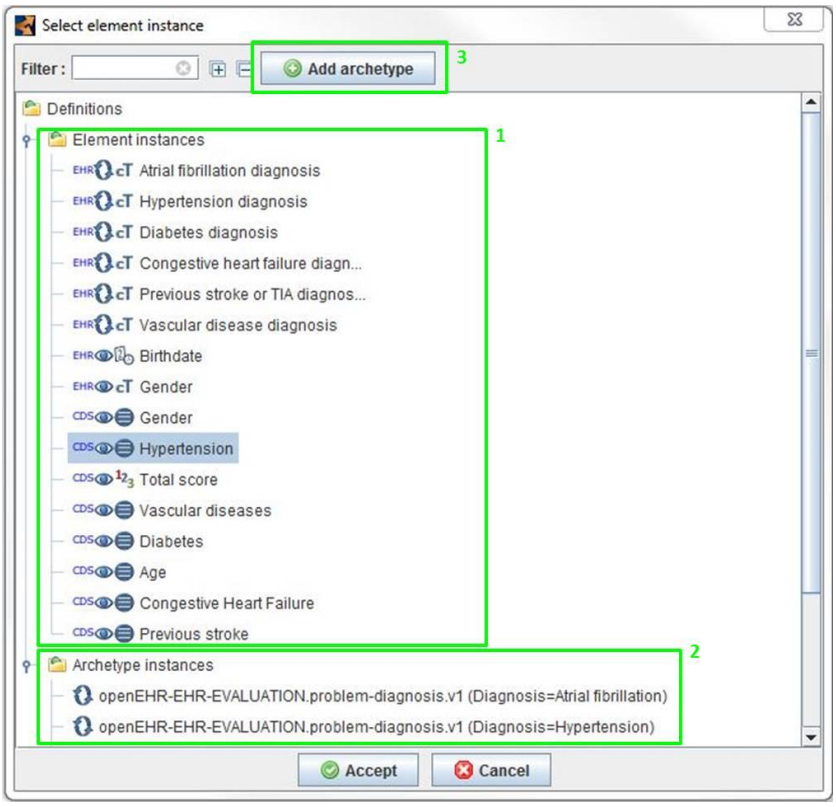

Figure 7: Select element instance

Editing conditions and actions is very similar to the definitions. The current version of the GDL editor supports six types of conditions:

- **Compare (DataValue):** compares the value of one element instance with a data value (constant).

- **Compare (NullValue):** compares the null value of an element instance with an openEHR NULL_FLAVOUR code.
- **Compare (Element):** compares the value of an element instance with the value of another element instance.
- **Compare (Attribute):** compares the attribute of an element instance with a constant or an expression (see EXPRESSION EDITOR).
- **Element is initialized:** checks whether if the element instance has or has no value assigned.
- **Or operator:** performs logical disjunction between two conditions.

Currently there are four types of actions supported:

- **Set (DataValue):** initializes the element instance with the data value selected.
- **Set (NullValue):** removes the value from the element instance and sets the NULL_FLAVOUR code selected.
- **Set (Element):** copies the value of one element instance to another.
- **Set (Attribute):** sets the value of an attribute using a constant or expression (see EXPRESSION EDITOR).

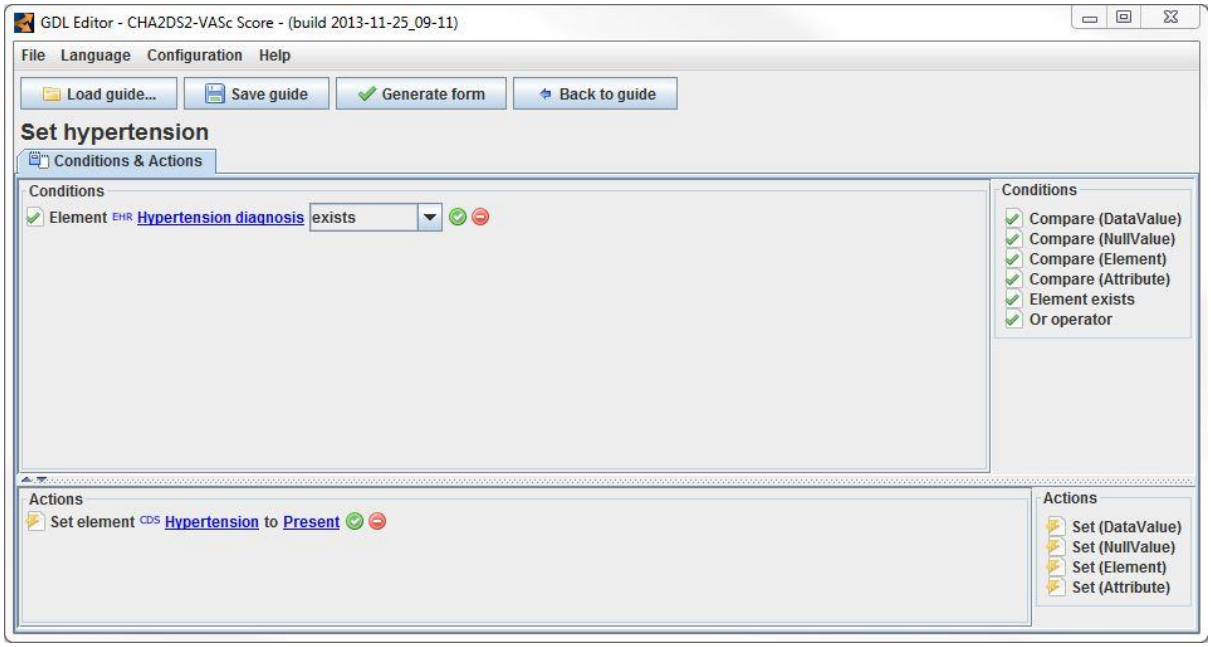

Figure 8: Rule editor

It's important to note that we will only be able to carry out actions on the element instances that correspond to an archetype instance in the CDS domain. This means that the rule engine is not able to make changes directly on the EHR elements.

> *Element names can be edited directly by right-clicking on its name.*

EXPRESSION EDITOR

Attributes of element archetypes can be compared to expressions containing other attributes or constant values. GDL supports a basic set of arithmetic operators (see Arithmetic operators on GDL Specifications). The expression editor is divided in two parts, the editing panel (above) and the viewing panel (bottom), any changes made on the editing panel will be displayed in the viewing part, if the expression is correct. Elements can be added to the expression manually or using the assistant (right side).

> *The expression viewer is used as an indicator for expression correctness. If empty, it means the text inserted cannot be parsed.*

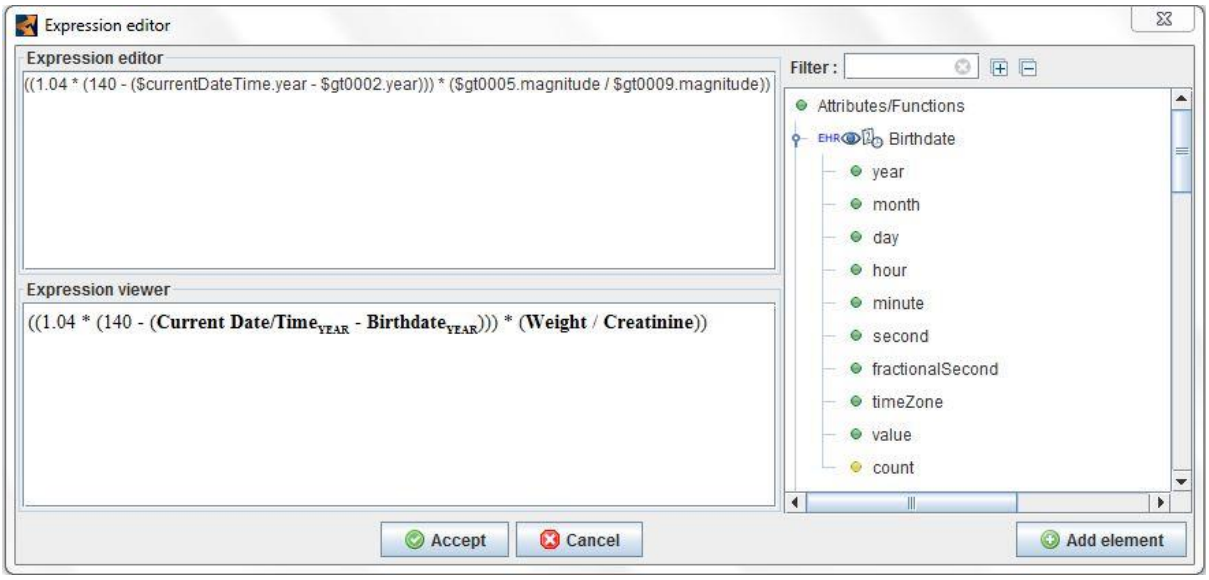

Figure 9: Expression editor

PRECONDITIONS

Preconditions are managed the same way conditions on the RULE EDITING. This section defines which facts must be matched before the guideline can execute.

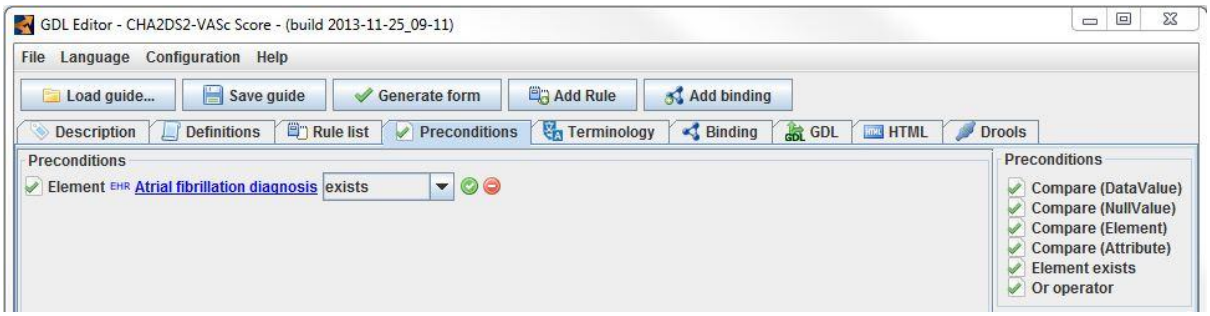

Figure 10: Editing preconditions

TERMINOLOGY

The terminology editor allows us to edit the different terms found on the GDL. GT codes are created automatically when we add element instances or create new rules, but they will have to be created manually when creating bindings.

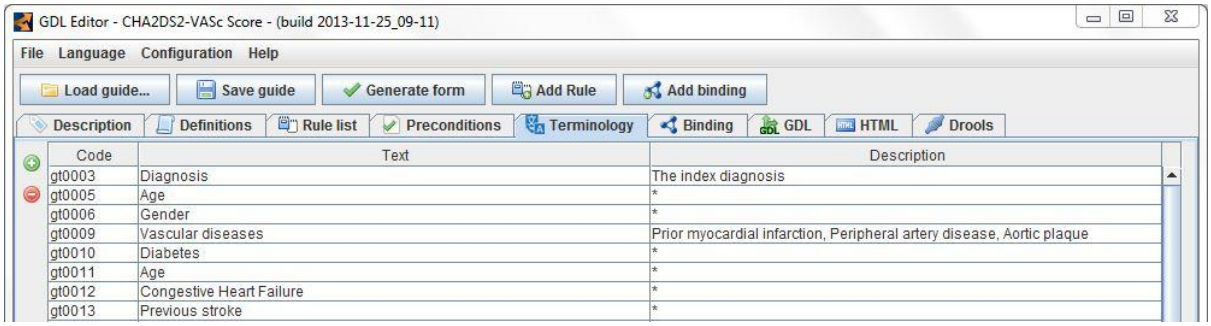

Figure 11: Terminology editing

To change the language of the terminology used we use the menu option *Language*. If we want to add a new language to the terminology, we can use the option *Add language…*

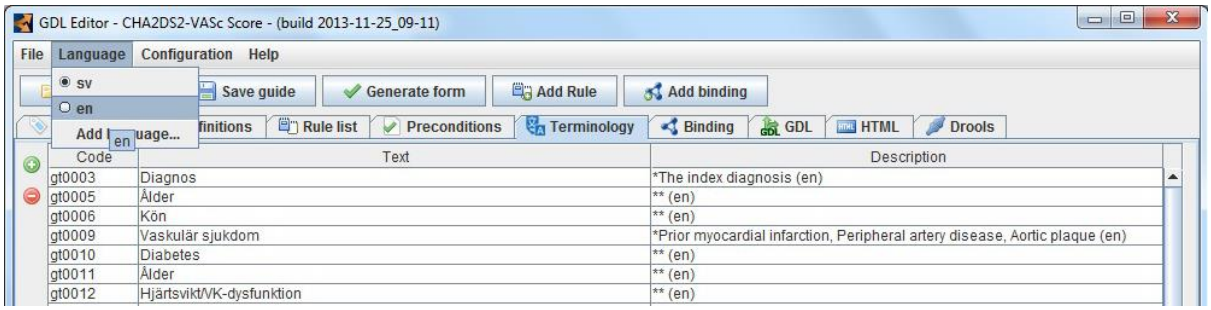

Figure 12: Changing the language of the terminology

BINDING

Terminology binding works the same way the rest of the DEFINITIONS. Clicking on the button '*Add terminology*' we will be able to create a new tab with the selected terminology.

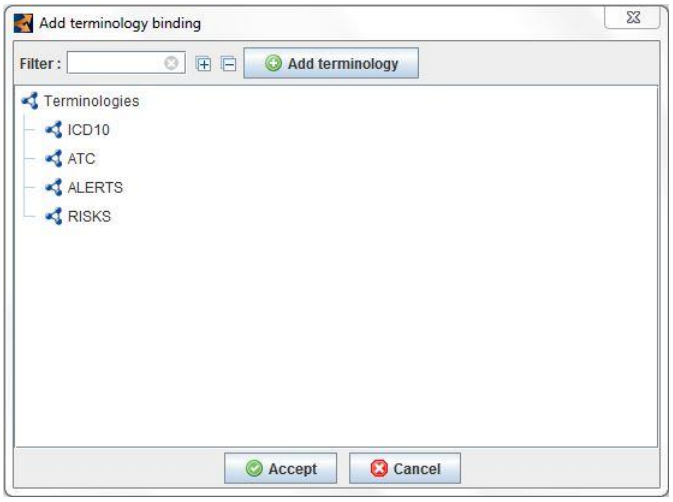

Figure 53: Add terminology binding dialog

When the guideline is executed, any code contained here and referenced with an *is_a* operator will be translated to its bound terminologies for resolution. Each row in the binding table contains three:

- **Local terms:** the codes defined in the local terminology of the guideline (see TERMINOLOGY). They can be selected by clicking on the field.
- **Terminology codes:** the codes of the terminology we are binding, separated by comma. Can be inserted manually or using the terminology viewer by double clicking on the magnifier icon.
- **URI:** an uniform resource identifier that points to a post-coordination expression (under development).

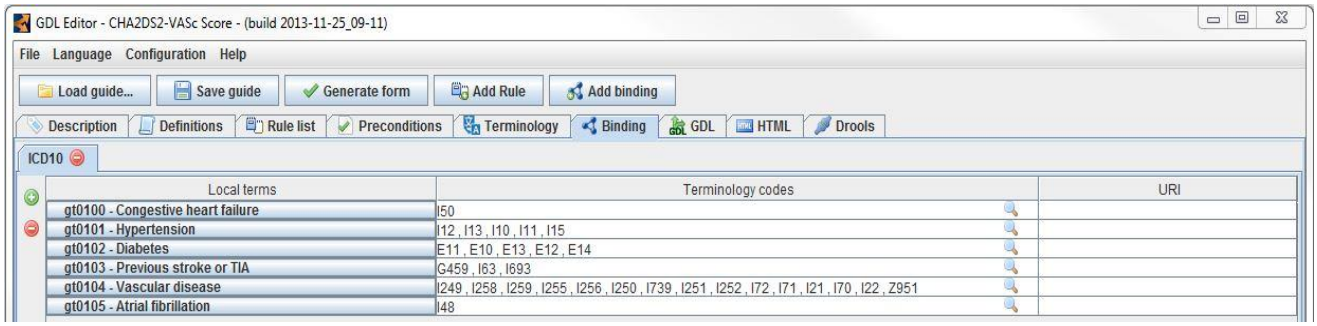

Figure 14: Binding of terminologies

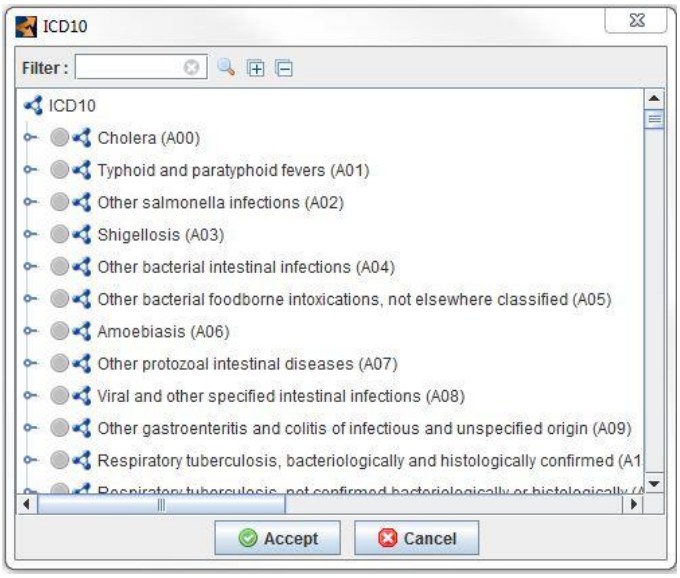

Figure 15: External terminology selection

GDL

The GDL viewer shows the result guideline in GDL format. When displayed, all the previous sections will be converted into formal language. This is the same format used when saving the guideline to a *.gdl* file. The GDL viewer also allows editing the GDL code.

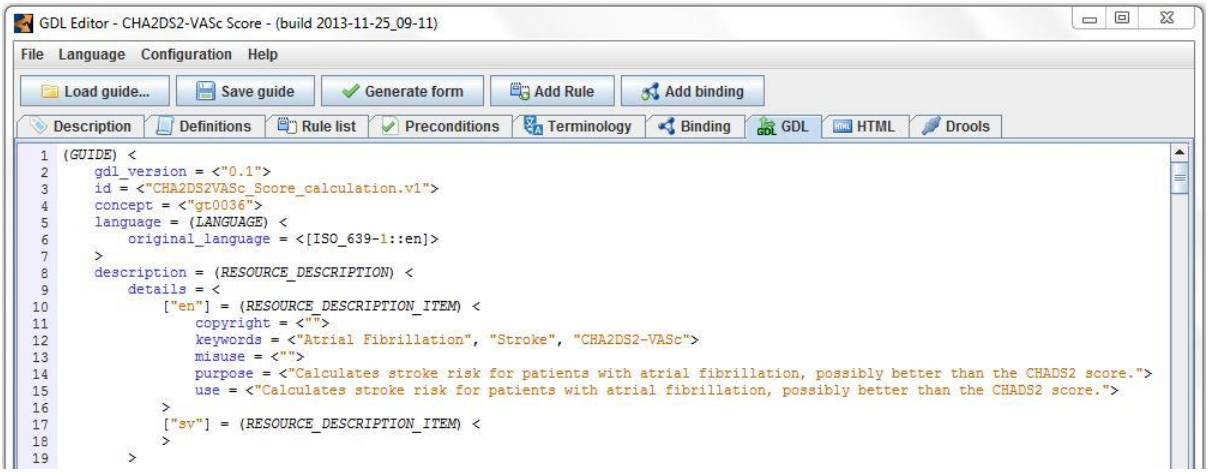

Figure 16: GDL view

If the editor finds any problems generating the GDL language, it will show us a dialog with a message, asking whether to ignore the problem or not. Most of these problems are due to empty values inside conditions and actions. To avoid these types of errors, make sure all the elements are properly filled before loading the GDL viewer.

Figure 17: Error serializing GDL guideline

HTML VIEW

This tab allows visualization of the GDL guideline in HTML format. You can use the tool *File>Export>Export to HTML* to save the content of the GDL in this format.

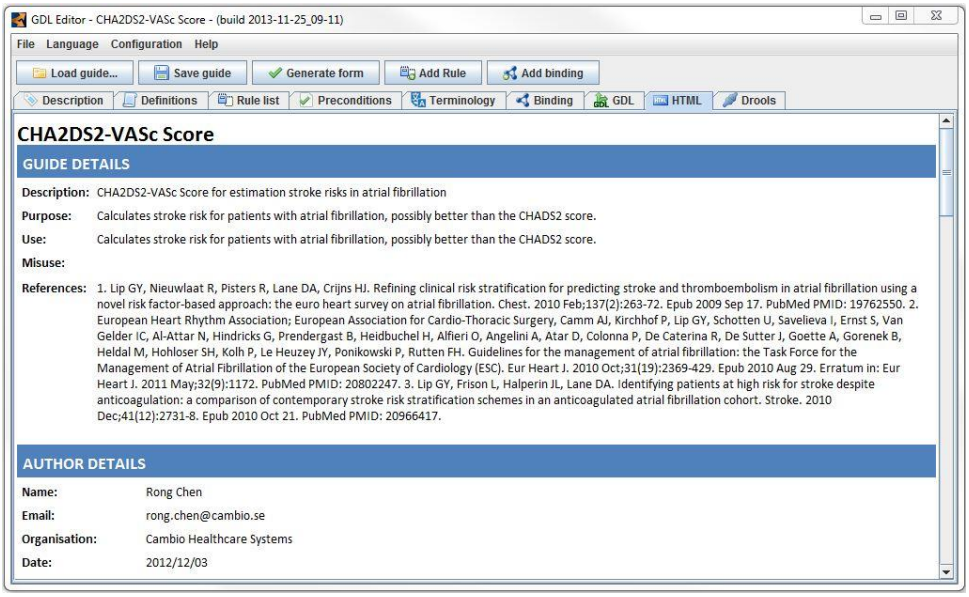

Figure 18: HTML view

IMPLEMENTATION VIEW

The editor supports several implementation views. Current version only displays Drools format, but can be extended with other rule engine languages. When generating a form, the implementation format will be compiled and used for rule execution.

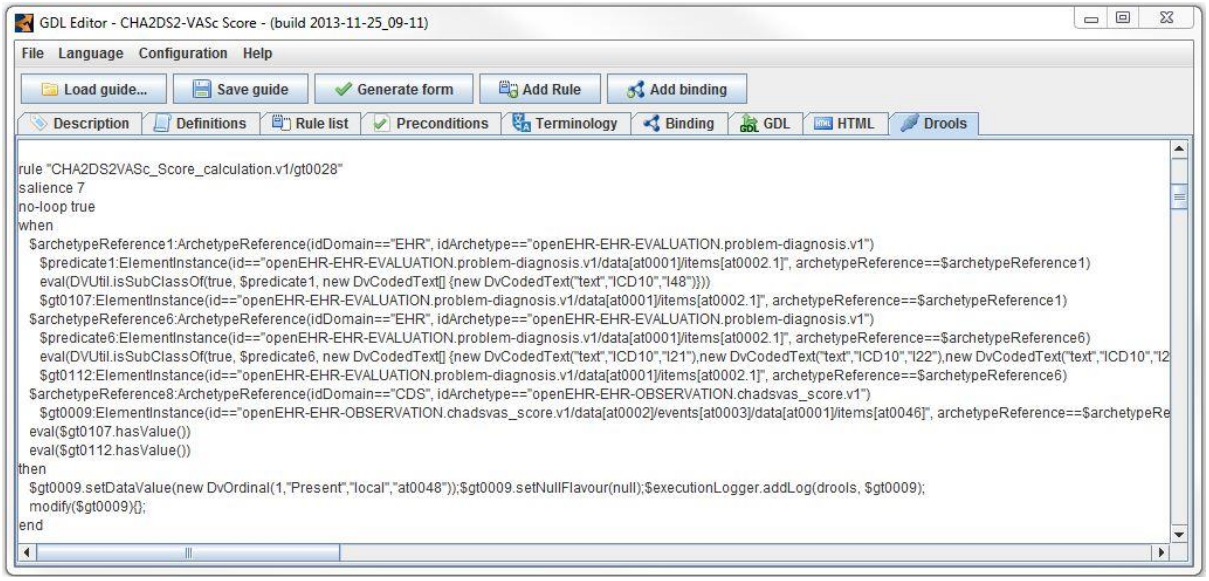

Figure 19: Drools implementation view

FORM GENERATOR

GDL editor contains a utility that allows guideline testing. The form generator will create a set of input fields based on the EHR/ANY content defined on the guideline. This data will be used to execute the rules in the guideline and present a result with the calculated values.

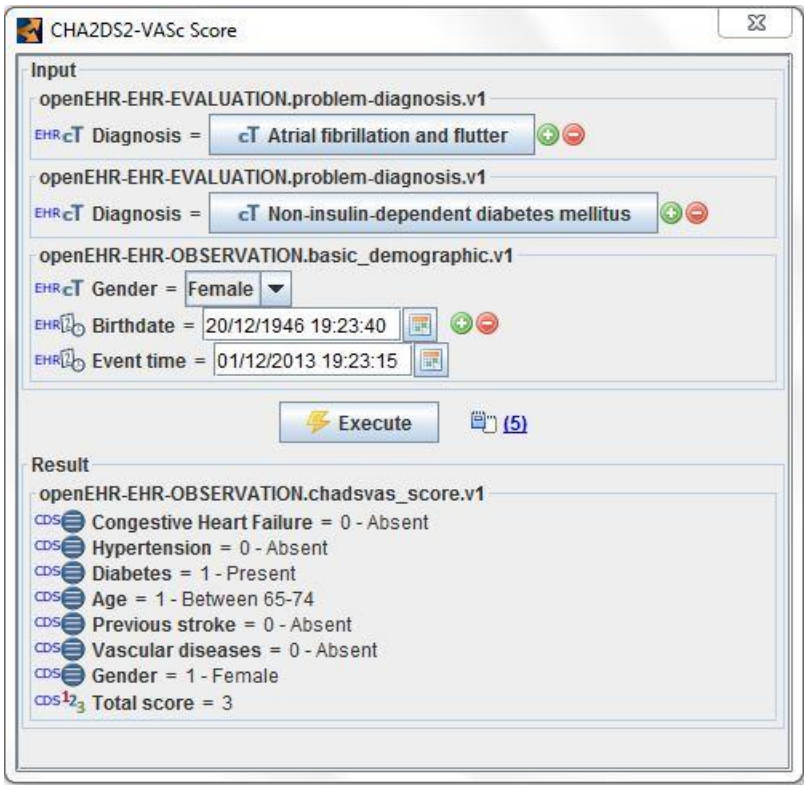

Figure 20: Form generator