Guide Definition Language (GDL)

Revision	0.1
Editors	Rong Chen[a], Iago Corbal[a]
Date of issue	18 December 2012

[a] Cambio Healthcare Systems

1 Introduction

1.1 Background

Expressing and sharing computerized clinical decision support (CDS) content across languages and technical platforms has been an evasive goal. Lack of commonly shared clinical information models and flexibly support for various terminology resources have been identified as two main challenges of sharing detailed clinical rules between sites.

1.2 Purpose

This document contains the design specifications of the guide definition language (GDL). GDL is a formal language designed to represent clinical knowledge for computerized decision support. GDL is designed to be natural language- and reference terminology- agnostic by leveraging the designs of openEHR Reference Model and Archetype Model.

1.3 Scope

The scope of the GDL is to support expressing of clear-cut clinical knowledge for single-decision making. Discrete GDL guides, artifacts writing in self-contain document in GDL format, can be combined together to support complex decision making process. It does not cover the process aspects of the clinical guideline.

1.4 Related Documents

- openEHR Reference Model Data Types Information Model
- openEHR Reference Model Data Structures Information Model
- openEHR Reference Model EHR Information Model
- openEHR Reference Model Common Information Model
- openEHR Archetype Model Archetype Object Model (AOM)
- openEHR Archetype Model Archetype Definition Language (ADL)

1.5 Nomenclature

In this document, the term ‘attribute’ denotes any stored property of a type defined in an object model, including primitive attributes and any kind of relationship such as an association or aggregation.

1.6 Requirements

1. It must be possible to express CDS rules using archetypes both as input and output for the rule execution.
2. It must be natural language-agnostic and can support multiple language translations in single rule without changing the rule definitions.
3. It must be reference terminology-agnostic so different terminologies can be used to support reasoning.

4. It should be straight-forward to convert the CDS rules to main-stream general-purpose rule languages for execution.
5. It must be able to express medical knowledge as first-order predicate rules.
6. There must be sufficient meta-information about the CDS rules, e.g. authorship, purpose, versions and relevant references.
7. It must be possible to reuse CDS rules written in the new format in different clinical context.
8. It should be possible to group a set of related CDS rules in order to support complex decision making process.

1.7 Design Principles

In response to the above mentioned requirements, the following principles are applied in the GDL design.

1.7.1 Archetypes both as Input and Output of Rules

This is achieved by creating bindings between data elements defined by archetypes and variables used by the CDS rules. Each CDS variable is uniquely identified in the context of the rule and bond to a specific data element defined by an archetype using Archetype ID and a path. Once defined, the variable can be used in the expressions of ***when*** and ***then*** statements as input or output of the rule execution.

1.7.2 Natural Language Neutrality

Several design ideas from openEHR archetype formalism are used to achieve natural language neutrality. First of all, all language-dependent meta-information about the purpose, use, misuse, authorship and references of the guide are grouped together under ***translations*** and indexed by ISO language codes inside GDL guide. Secondly, all language-dependent labels and descriptions, e.g. the name of a varialbe, are defined in the ***terminology definition*** section of the guide and indexed by ISO language codes. Thirdly, unique identifiers for variables and rules are used in rule expressions instead of their names, which are language dependent.

1.7.3 Rule Language Neutrality

GDL only uses a set of common rule language features, like “when” and “then “. The expressions in the when and then statements support common arithmetic calucations, logic operator and functions.

1.7.4 Grouping and Reuse of Rules

A GDL guide may constain several rules that related to each other. Each guide is self-containing and should be reusable across different clinical context. Different guides can be chained together to support complex decision support. This is achieved by selecting output of a guide, a specific element of an archetype, as input of anther guide.

1.7.5 Meta-information of the CDS rules

Authoring information, lifecycle state and various meta-information are supported by reuse RESOURCE_DESCRIPTION class.

2. Guide Object Model

2.1 Design Background

The underpinning of GDL design is the use of openEHR archetypes and templates both as input and output of CDS rules. This is the key to achieve natural language- and reference terminology-independence. Because of that, openEHR design specifications take a major part of the GDL design. In other words, the GDL design is aimed to make substantial reuse of existing openEHR specifications. In areas where existing openEHR design is not sufficient, additional designs are introduced.

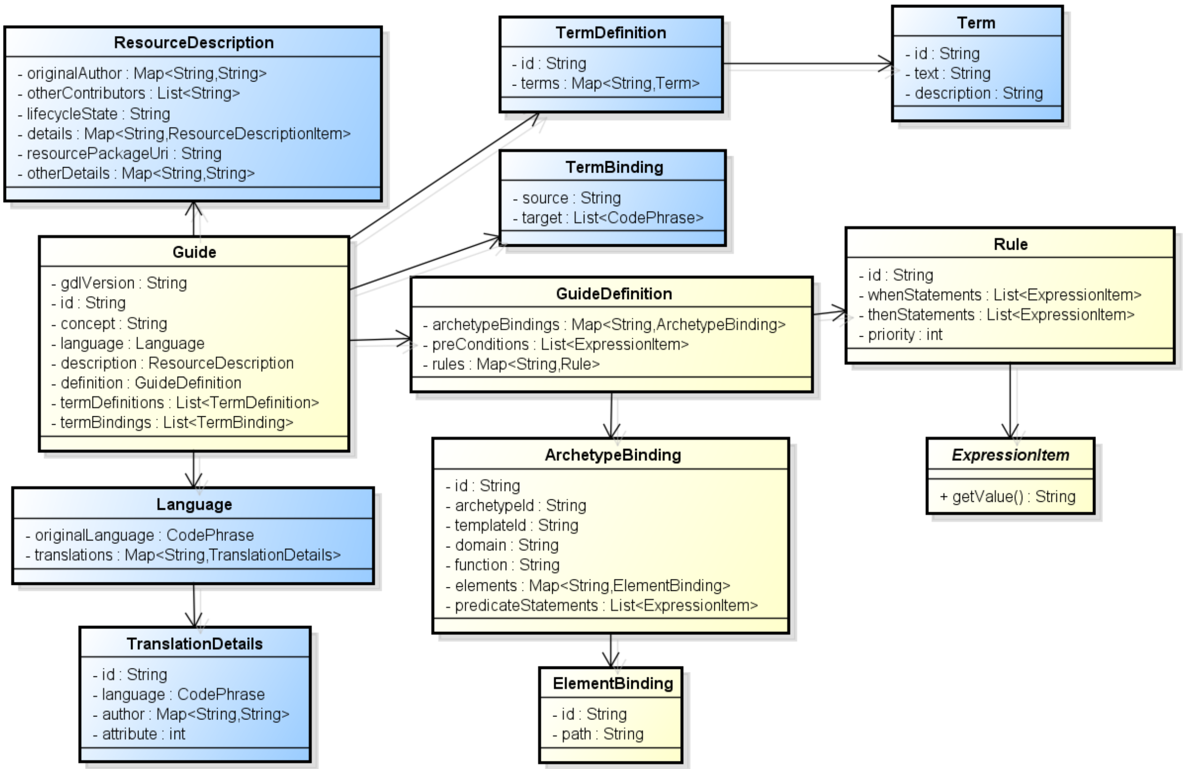
2.2. Packages Structure

The Guide Object Model, the object model of the GDL consists of two packages, the guide package and the expressions package.

3 Guide Package

3.1 Overview

The overview of the guide package is illustrated in figure 1. Note that classes in blue color are loosely based on the original design from the openEHR specifications.



ResourceDescription
- originalAuthor : Map<String,String>
- otherContributors : List<String>
- lifecycleState : String
- details : Map<String,ResourceDescriptionItem>
- resourcePackageUri : String
- otherDetails : Map<String,String>
TermDefinition
- id : String
- terms : Map<String,Term>
Term
- id : String
- text : String
- description : String
TermBinding
- source : String
- target : List<CodePhrase>
Guide
- gdlVersion : String
- id : String
- concept : String
- language : Language
- description : ResourceDescription
- definition : GuideDefinition
- termDefinitions : List<TermDefinition>
- termBindings : List<TermBinding>
Rule
- id : String
- whenStatements : List<ExpressionItem>
- thenStatements : List<ExpressionItem>
- priority : int
GuideDefinition
- archetypeBindings : Map<String,ArchetypeBinding>
- preConditions : List<ExpressionItem>
- rules : Map<String,Rule>
ExpressionItem
+ getValue() : String
Language
- originalLanguage : CodePhrase
- translations : Map<String,TranslationDetails>
ArchetypeBinding
- id : String
- archetypeId : String
- templateId : String
- domain : String
- function : String
- elements : Map<String,ElementBinding>
- predicateStatements : List<ExpressionItem>
TranslationDetails
- id : String
- language : CodePhrase
- author : Map<String,String>
- attribute : int
ElementBinding
- id : String
- path : String

Figure 1 – The Guide Package

3.2 Class Definitions

3.2.1 GUIDE Class

CLASS	GUIDE	
Purpose	Main class of a discrete guide, which defines archetype bindings, rules and meta-information.	
Attributes	**Signature**	**Meaning**
0..1	**gdl_version:** String	the version of the GDL the guide is written in.
1..1	**id**: String	Identification of this guide
1..1	**concept**: String	The normative meaning of the guide as whole. Expressed as a local guide code.
1..1	**language**: Language	Natural language resources of this guide. It includes an original language and optional list of translations.
1..1	**description**: RESOURCE_DESCRIPTION	Resources description of this guide including authorship, use/misuse, life-cycle and references.
1..1	**definition**: GUIDE_DEFINITION	The main definition part of the guide. It consists of archetype bindings and rule definitions.
1..1	**ontology**: GUIDE_ONTOLOGY	The ontology of the guide.

3.2.2 GUIDE_DEFINITION

CLASS	GUIDE_DEFINITION	
Purpose	The definition of the guide. It includes a list of archetype bindings and a list of rule definitions.	
Attributes	**Signature**	**Meaning**
1..1	**archetype_bindings:** List<ARCHETYPE_BINDING>	List of archetype bindings, which define specific elements to be used by rules.
1..1	**rules**: Map<String, Rule>	Map of rules indexed by local gt codes.
0..1	**pre_conditions**: List<EXPRESSION_ITEM>	List of pre-conditions to be met before the guide should be executed.

3.2.3 ARCHETYPE_BINDING

CLASS	ARCHETYPE_BINDING	
Purpose	The binding of list of elements from a selected archetype or template to local gt codes	
Attributes	**Signature**	**Meaning**
1..1	**archetype_id:** String	The ID of the archetype, from where the list of elements are selected.
0..1	**template_id:** String	The ID of an optional template to be used for selecting elements.
0..1	**domain:** String	The space in which the rule variables reside. The value can either be "EHR" meaning the value is retrieved from the EHR, or "CDS: meaning the value is derived in the CDS engine. When missing, the assumption is either "EHR" or "CDS".
0..1	**function**: String	The function to perform when querying the EHR information. Typical function can be "count" or "average".
0..1	**time_range**: String	The time range, with which the archetype instances are retrieved from EHR.
1..1	**Elements:** Map<String, ELEMENT_BINDING>	Map of element binding indexed by local gt codes.
0..1	**predicate_statements:** List<EXPRESSION_ITEM>	List of predicates to be fulfilled before the EHR queries can be performed

3.2.4 ELEMENT_BINDING

CLASS	ELEMENT_BINDING	
Purpose	The binding between a specific element in an archetype and a local variable in the guide.	
Attributes	**Signature**	**Meaning**
1..1	**id:** String	The local gt code of this element
1..1	**path**: String	The path to reach this element in the archetype.

3.2.5 RULE

CLASS	RULE	
Purpose	A single rule defined in a guide	
Attributes	**Signature**	**Meaning**
1..1	**id:** String	The local gt code of this element
1..1	**when_statements**: List<EXPRESSION_ITEM>	List of expressions to be evaluated before the rule can be fired.
1..1	**then_statements**: List<ASSIGNMENT_EXPRESSION	List of expression to generate output of the rule.

4 Expressions Package

4.1 Overview

The overview of the expressions package is illustrated by figure 2.

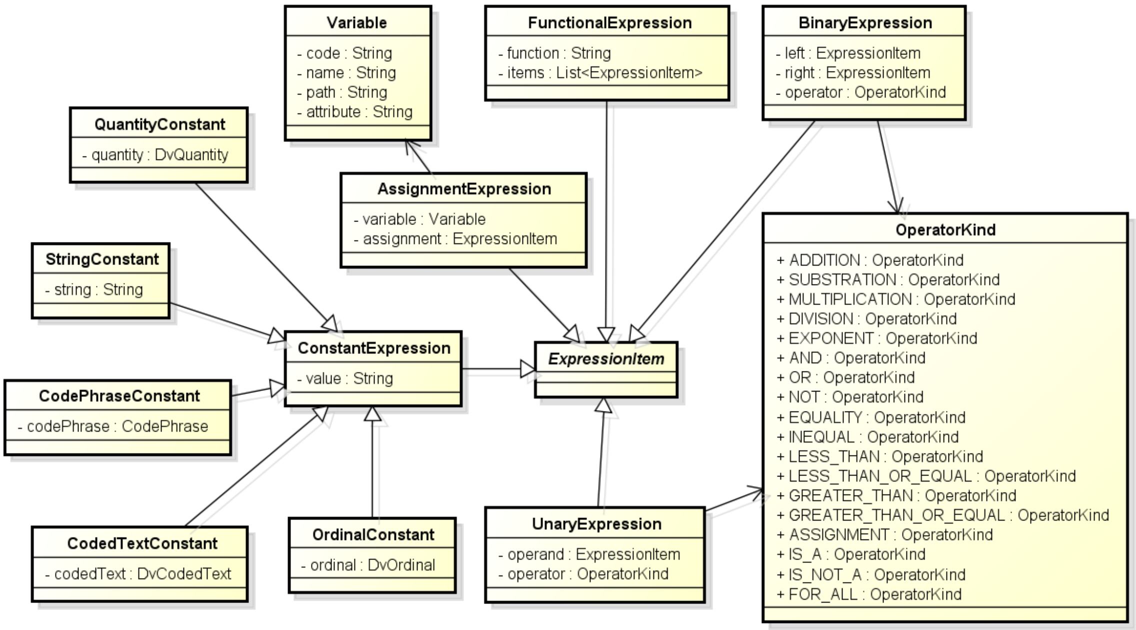


Figure 2 – The Expression Package

4.2 Class Definitions

4.2.1 EXPRESSION_ITEM

CLASS	*EXPRESSION_ITEM* (abstract)	
Purpose	Abstract model of an expression item in the rule.	
Attributes	**Signature**	**Meaning**

4.2.2 UNARY_EXPRESSION

CLASS	UNARY_EXPRESSION	
Purpose	Abstract model of an expression item in the rule.	
Inherit	EXPRESSION_ITEM	
Attributes	**Signature**	**Meaning**
1..1	**operand:** EXPRESSION_ITEM	The operand of this unary expression.
1..1	**operator**: OPERATOR_KIND	The operator of this unary expression.

4.2.3 BINARY_EXPRESSION

CLASS	BINARY _EXPRESSION	
Purpose	Concrete model of a binary expression item.	
Inherit	EXPRESSION_ITEM	
Attributes	**Signature**	**Meaning**
1..1	**left:** EXPRESSION_ITEM	The left operand of this binary expression.
1..1	**right:** EXPRESSION_ITEM	The right operand of this binary expression.
1..1	**operator**: OPERATOR_KIND	The operator of this binary expression.

4.2.4 ASSIGNMENT_EXPRESSION

CLASS	ASSIGNMENT_EXPRESSION	
Purpose	Concrete model of an assignment expression.	
Inherit	EXPRESSION_ITEM	
Attributes	**Signature**	**Meaning**
1..1	**variable:** String	The gt code of the variable to assign the value to.
1..1	**assignment**: EXPRESSION_ITEM	The expression item, from which the value is derived from.

4.2.5 FUNCTIONAL_EXPRESSION

CLASS	FUNCTIONAL_EXPRESSION	
Purpose	Concrete expression models a function.	
Inherit	EXPRESSION_ITEM	
Attributes	**Signature**	**Meaning**
1..1	**function:** String	The name of the function.
1..1	**items**: List<EXPRESSION_ITEM>	A list of parameters to the function.