Leveraging ADL Archetypes by transforming them to AML Archetypes

A THESIS
SUBMITTED TO THE FACULTY OF
GRADUATE SCHOOL
UNIVERSITY OF MINNESOTA

BY

DEEPAK KUMAR SHARMA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE

AUGUST 2015

Acknowledgements

The Graduate School at the University of Minnesota has been a pleasant learning experience for me, not only at the academic level but also personal level. First and foremost, I would like to thank my thesis advisory committee, Dr. Christopher G. Chute, MD, Dr. P.H. at Johns Hopkins, Dr. Claudia Neuhauser, PhD. at University of Minnesota and Dr. Guoqian Jiang, MD, PhD. at Mayo Clinic. They have played a very important role in successfully completing this Master's with their valuable feedback, guidance and sharing of their experience, knowledge and expertise. Their guidance has been crucial in selecting most appropriate topics to study and help strengthen my grasp of the concepts of healthcare informatics.

I would also like to acknowledge and thank Harold Solbrig at Mayo Clinic, who has been a mentor to me, not only for this effort but also for last 12 years, since I joined Mayo Clinic. He has been very instrumental to help me understand and implement the technical aspects of the AML specifications and tooling. I would like to express my gratitude to all collaborators and colleagues at Mayo Clinic for their support and providing me this opportunity to pursue the Master's degree. I am also thankful to the members of the Clinical Information Modeling Initiative (CIMI) Modeling Task Force, including Dr. Stan Huff, Thomas Beale and Robert Lario.

I am also grateful to my wife, Seema, and my kids, Vamika and Viraaj. This would not have been possible without their love, encouragement, patience and support.

Dedication

Dedicated to my mother, Usha Rani Sharma, and father, Ishwar Dayal Sharma

Abstract

The Clinical Information Modeling Initiative (CIMI) has developed the Archetype Modeling Language (AML) specifications, which is now an Object Management Group (OMG) standard. The AML is for modeling archetypes using the Unified Modeling Language (UML). The development of the AML specifications is part of one of the goals for CIMI - to deliver a shared repository of clinical models that is open and free to use. The AML is an attractive option to create, reuse and extend archetypes and the ability to share these archetypes greatly improves interoperability.

AML is new standard with lot of promises and benefits, but lacks support of any tooling to get started with creating AML archetypes easily. The ADL archetypes are built using a proprietary format and hence lack an easy gateway to Model-Driven Architecture. The author has created maps for transforming existing archetypes in the OpenEHR's Archetype Definition Language (ADL) to AML workspace. These proven mappings bridge the gap between ADL and AML by providing seamless transition and leverage the ADL archetypes to the AML modeling workspace. This thesis is about these mappings and their implementation.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADL	Archetype Definition Language (ADL), version 1.5, July 2014, OpenEHR. http://www.openehr.org/releases/trunk/architecture/am/adl1.5.pdf
AML	Archetype Modeling Language (A UML Profile for Modeling Archetypes), OMG RFP, http://www.omg.org/cgi-bin/doc.cgi?health/2012-7-1
AOM	Archetype Object Model (AOM), version 1.5, September 2014, OpenEHR, https://github.com/openEHR/specifications/blob/master/publishing/architecture/am/aom1.5.pdf
BMM	Basic Meta-Model, https://github.com/openEHR/bmm
CIMI	Clinical Information Modeling Initiative (CIMI). http://www.opencimi.org/
CTS2	Common Terminology Services 2 (CTS2) 1.2, Object Management Group Specifications, April 2015, http://www.omg.org/spec/CTS2/1.2/
HL7	Health Level 7, http://www.hl7.org
JSON	JavaScript Object Notation (JSON), http://json.org/
LOINC	Logical Observation Identifiers Names and Codes (LOINC), https://loinc.org/
MDR	ISO/IEC 11179, Information Technology, -- Metadata registries, http://metadata-standards.org/11179/
NIEM	OMG UML Profile for NIEM Version 1.0, http://www.omg.org/spec/NIEM-UML/1.0/
OCL	Object Constraint Language, Object Management Group (OMG), http://www.omg.org/spec/OCL/
ODIN	Object Data Instance Notation (ODIN), version 1.5.1, OpenEHR, http://www.openehr.org/releases/trunk/architecture/syntaxes/ODIN.pdf

OMG	Object Management Group, http://www.omg.org/
RDF	Resource Description Framework, World Wide Web Consortium (W3C), http://www.w3.org/RDF/
SNOMED	Systematized Nomenclature of Medicine (SNOMED), The International Health Terminology Standards Development Organization (IHTSDO). http://www.ihtsdo.org/snomed-ct
UML	OMG Unified Modeling Language, Version 2.5. Object Management Group (OMG) Specification, Beta 2, Sept 2013. http://www.omg.org/spec/UML/2.5/Beta2/
XMI	OMG XML Metadata Interchange (XMI) Format http://www.omg.org/spec/XMI/
XML	W3C Extensible Markup Language (XML), http://www.w3.org/XML/

1 Introduction

1.1 Motivation

The Archetype Modeling Language (AML) specifications [1] are greatly influenced by the OpenEHR's [7] Archetype Definition Language (ADL) and its ADL Object Model (AOM). This is due to the fact that the ADL has been able to get the most traction among many other efforts to create standards, models and frameworks for building and managing clinical archetypes. The ADL archetypes are created in a proprietary format, Object Data Instance Notation (ODIN), while the AML archetypes are created in the OMG Unified Modeling Language (UML).

There are some useful open-source tools (mainly contributed by the Ocean Informatics [8]) available at the OpenEHR to aid creation, review, manage and export ADL archetypes. The archetypes are based on the reference models and having these reference models in yet another proprietary format, Basic Meta-Model (BMM) is a prerequisite for working with ADL archetypes. This is an additional step for modelers, to transform their reference models into BMMs first, before they can start creating ADL archetypes. Even though the modelers can export ADL archetypes into some other useful formats using the OpenEHR modeling tools, the maintenance of ADL archetypes is tied to the OpenEHR tools only, e.g., the archetype editor - ADL Workbench IDE (AWB) [6].

AML offers several advantages just by having archetypes in UML format. UML is an OMG standard and is non-proprietary. UML is familiar to modelers around the world and provides an important gateway to the Model-Driven Architecture [19], which ODIN/BMM don't. There are numerous UML based tools already created and may be used directly to utilize AML models in various ways. It is likely that a new or existing reference model is created in UML and that eliminates the need to transform the reference model into a proprietary format. In other words, AML works directly with the reference models in UML.

There are many benefits of creating mappings between ADL and AML and developing transformation tool to realize these mappings. Modelers around the world have created many rich sets of archetypes using the OpenEHR's ADL modeling tools. These archetypes are very valuable as they are reviewed and vetted archetypes and currently being used throughout the world. Their transformation to equivalent AML archetypes will make them available to the new potential AML modelers, who can re-use and extend them without having to create them from scratch. Their AML transformation would also provide an easier way to use them in applications that are based on model-driven architecture. The implementation artifacts of this transform could be embedded to develop innovative user-friendly interfaces to make AML modeling easier.

1.2 Background

1.2.1 Constraints Based Models

An *archetype* is a collection of constraints on a given reference model and a collection of archetypes is called an *archetype library*. The archetype modeling is the *constraints based* modeling where a reference model, a model with all the classes, becomes the most abstract level of exchange. The classes in the reference model are specialized by defining constraints to narrow them down. The constraints, which do not affect the model class in any way, may be defined on the cardinality of the attributes, the values and the value ranges that can be assigned. Any instance of the narrowed down view will be valid at the recipient's end of the *shared* reference model. The shared information between two systems now contains the reference model class and the set of constraints *about* it. The constraints based modeling approach makes more sense in an environment where a common shared set of model classes need to be 'constraint down' based on the requirements. An archetype can be viewed as a funnel (Figure 1) that filters the targeted instances among all possible instances of the reference model.

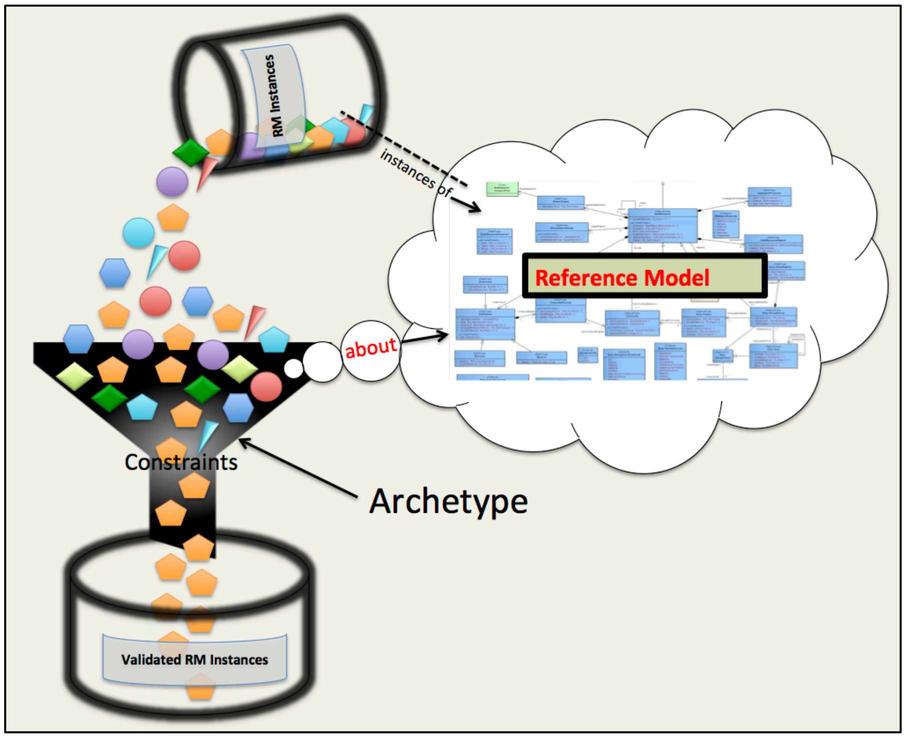


Figure 1: Constraint Based Modeling

The shared archetypes strive to solve the problem of *interoperability* – to preserve information and its semantics during exchange. The efforts of the standardization of terms and creating 'IsoSemantic' models (where instances intend to mean the same thing) are some other ways to improve the interoperability, but they are proved to be partially effective and almost always need additional transformational steps to extract and convey the intended meaning of the exchanged information.

An archetype defines a domain model that talks about the information model but still stands separate from it. This separation is the key feature of archetypes that provides flexibility for a user application, which employs an archetype library as an external resource instead of having constraints embedded in its implementation.

1.2.2 The Clinical Information Modeling Initiative

Since 2011, mainly four organizations – the Health Level 7 (HL7), Intermountain HealthCare (IHC), OpenEHR and Mayo Clinic have joined efforts (along with other collaborating organizations) in developing converging specifications under the umbrella of the Clinical Information Modeling Initiative (CIMI). The CIMI work is influenced and guided by the grounds gained with the HL7 Detailed Clinical Models [11], the OpenEHR ADL Archetypes, the Meta Data Repository (MDR) - ISO/IEC 11179 specifications [2], and Mayo Clinic's rich experience of developing vocabulary management solutions like CTS2 and LexEVS [5].

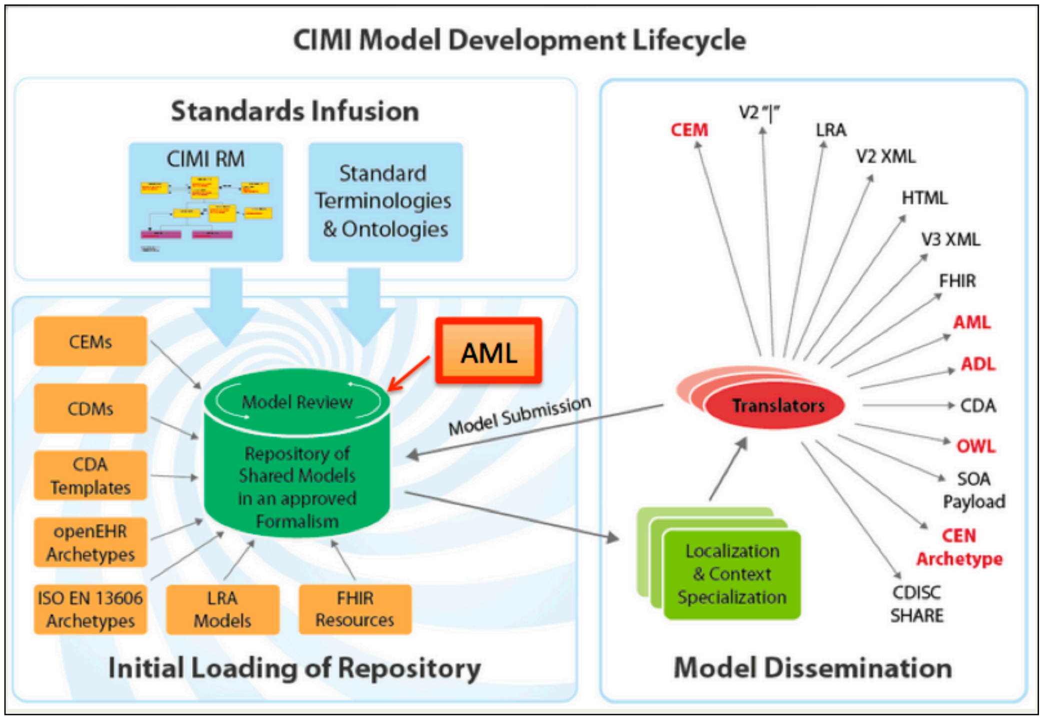


Figure 2: The CIMI Model Repository

CIMI is independent of any standards group and ensure that the models that are created are <u>open and free to use</u>. CIMI's strategic goal is to be able to share data, applications, reports, alerts, protocols, and decision support modules with anyone in the world. CIMI offers single formalism with two representations – ADL and AML, and formal bindings to two standard terminologies – SNOMED CT and LOINC.

The shared clinical model repository of CIMI enables developers of healthcare information systems to re-use existing reviewed models. The Initial set of models come from the existing OpenEHR's ADL archetypes at Clinical Knowledge Manager (CKM) [9] and the IHC Clinical Element Models (CEM) [10] converted to ADL. As of August

2015, they are being reviewed and transformed into archetypes, with the CIMI Reference Model (CIMI RM) as the reference model, for their load into the CIMI shared Model repository in near future.

1.2.3 Role of UML Profiles

The AML specifications, guided by the OpenEHR Foundation's ADL and its ADL Object Model (AOM), target to fulfill the requirement of representing the Archetype Models (AMs) by accurately following the ADL and AOM specifications. AML intends to support primarily two types of communities: the UML modelers, who may or may not be clinical domain modelers, and the clinical domain model experts, who are currently using ADL.

The AML specifications are organized, as a set of three UML profiles - Reference Model Profile (RMP), Terminology Binding Profile (TBP) and Constraint Model Profile (CMP). A UML Profile is an extension mechanism to customize an existing metamodel (with constructs like the stereotypes, tagged values, and constraints) to a specific domain, platform or process. A UML Profile does not change metamodel in anyway, and it just adds new 'constraints' to it and hence creates a restricted form of metamodel.

A UML profile is a package of stereotypes, but can also have the enumerations, primitive types, data types, and constraints. A UML Profile can import other UML Profiles. A UML Profile can be applied or removed from a metamodel dynamically. The UML Profile constraints are evaluated when it is applied. A UML Profile can restrict availability of the UML elements, when applied in a strict mode. This can guide modelers to create models using only the 'valid' UML elements filtered by the strictly applied UML Profile (Figure 3).

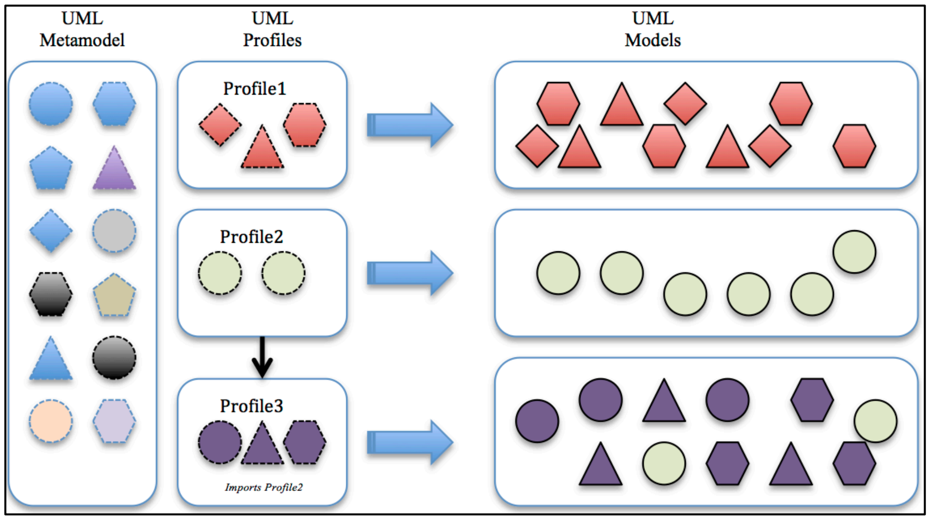


Figure 3: The UML Profiles guide the resulting UML model for their content

A Reference Model (RM), an important part of the archetype modeling, is a model on which archetypes are based; hence first step, for a modeler, is to make sure the RM has necessary model elements. The process of writing constraints on the RM elements should be guided by constraint specifications and the terminology binding specifications (for using the external terminologies to provide meaning to model elements). The three AML profiles strive to help accomplish these objectives.

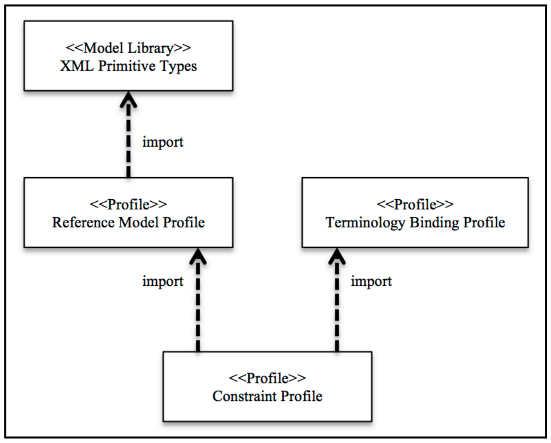


Figure 4: The AML Profile dependencies

Figure 4 shows the dependencies among the three UML profiles of AML. The UML XML Primitive Types library represents the data types of XML Schema as defined by the National Information Exchange Model (NIEM) UML Profile.

1.2.4 Problem Statement

In mid 2014, soon after I started working with the Clinical Information Modeling Initiative (CIMI) Modeling Task Force for the development of the AML specifications, I realized the absence of any tool to move between the ADL and UML representations of archetypes. The AML, now being the newly minted OMG standard, does not have any authoring environment or tool to create and manage AML archetypes. The advantages of having archetypes in AML over ADL, made it one of the supported archetype formalism by CIMI, but lack of any tool prevents or makes it harder for the UML modelers to get started with the AML archetypes.

The manual creation new AML archetypes or migration of existing ADL archetypes into AML's UML format is a process which is tedious, time-consuming, error-prone and quickly becomes outright confusing even to the expert UML modelers. The clinical modelers who have been using ADL to model, can't take their models into AML workspace as no such export mechanism exist in ADL Modeling environment. The gap between ADL and AML workspaces exist – ADL cannot get benefits of UML representation of AML, while AML lacks tooling and hence requires manual creation of new and existing archetypes, which is not practical.

As mentioned earlier, there are rich sets of established ADL archetypes that could be used immediately with a solution that can bridge the gap between ADL and AML representations of archetypes.

1.2.5 Method and Goals

A possible solution to the problem, stated in previous section, demands identification of a set of lossless mappings between ADL and AML to close the gap and let modelers move between ADL and AML seamlessly. An implementation of these mappings can be good first step to let users create AML archetypes and import existing ones.

Even though the AML specifications constitute relatively simpler stereotypes, it does not stop a modeler to employ them in a way, which might be different from its intended use. A way of creating new AML archetypes, in addition to the imported and transformed one, needs to be part of such implementation that will avoid the deviation from intended usage of the AML specification.

The mappings and their well-implemented artifacts can fuel the development of other tools that create higher-level graphical user-interface (GUI) applications to create and manage new AML archetypes and a way to import/export existing ADL or AML archetypes.

So the goals of this project were set - to develop lossless mappings between ADL and AML and use them to programmatically serialize the existing ADL archetypes into their AML counterpart. The development of the solution to the stated problem triggered from my efforts to develop a temporary utility that minimizes tedious (and error-prone) manual steps of creating AML archetypes in a UML editor. But I quickly realize that a well-developed solution could become a valuable resource to develop additional libraries and GUI tools in near future. One could also use it to check and validate if a resulting AML objects satisfy the AML archetype requirements. The process of identifying the mappings, ADL to AML, could help review and figure out the limitations and gaps in AML specifications itself.

The approach to implement a solution included the following steps for me:

- Have a clearer understanding of both formats – ADL and AML
- Identify lossless mappings between ADL and AML constructs
- Implement the mappings to create a library with a programming interface

The completion of these steps and the resulting implementation library, in its current state, encapsulates my work for this thesis.

The first step in my approach was the most challenging – to equip myself with really good understanding of both formats, ADL and AML, and the knowledge of the reasons for the alternatives chosen to represent objects in both representations. I gained this knowledge over time working with CIMI Modeling Task Force assignments, discussions, literature research and valuable guidance from my advisers and mentors.

The experience and knowledge gained over last decade, while working on various projects in the biomedical informatics domain at Mayo Clinic, helped immensely.

This knowledge gained was instrumental in creating *lossless* maps between ADL and AML. I deduced these mappings, in parallel, as I progressed with CIMI modeling tasks for development of the AML specifications. The efforts also involved getting conversant with various modeling tools and solutions involving the ADL and AOM, which also have evolved in parallel with the AML specifications.

An archetype definition is mainly divided into three sections –Metadata, Constraints Definition and Terminology Binding. Following the same, the mappings too are divided into three parts describing counterparts from ADL and AML archetype constructs (Figure 5). The details of the identified mappings are described in section 2.

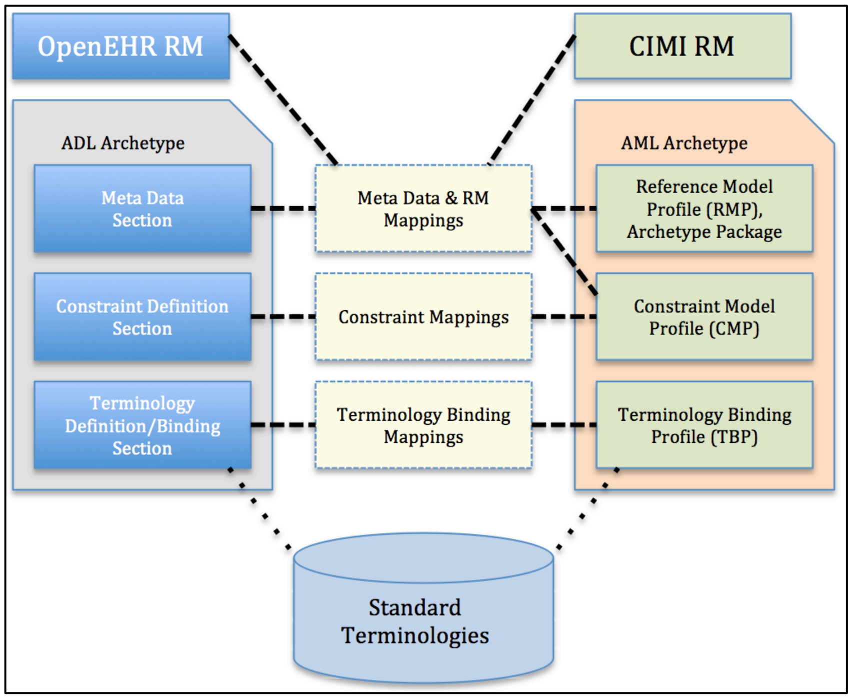

OpenEHR RM
CIMI RM
ADL Archetype
Meta Data Section
Constraint Definition Section
Terminology Definition/Binding Section
Meta Data & RM Mappings
Constraint Mappings
Terminology Binding Mappings
AML Archetype
Reference Model Profile (RMP), Archetype Package
Constraint Model Profile (CMP)
Terminology Binding Profile (TBP)
Standard Terminologies

Figure 5: Mappings between ADL and AML

The third and last step was to represent these mappings well in the implementation to get a reliable transformed content. The goal of implementation was to deliver an Application Programming Interface (API) library for creating AML objects in memory. The library created not only allows creation of new AML objects, but also imports ADL and transforms them into AML using the mappings established between them. The initial version of library would at least include implementing mappings related to archetype identification and constraints (which constitute the *core* of an archetype).

At the time of writing this thesis, the released version of the library included implementation of the CMP mappings only, and based on the AML specifications submitted in November 2014. The mappings described in this thesis have been updated to the latest AML 2015 specifications and the implementation library is being updated for the RMP and TBP mappings.

2 ADL to AML Transformation

The three sections of an ADL archetype – Metadata, Constraints and Terminology, directly correspond to the three AML profiles. I have established these mappings between these ADL sections and AML profiles that are described in detail here.

2.1 Reference model mappings

Even though the modelers are free to create and use their own reference models, the AML specifications include a reference model – the CIMI Reference Model (CIMI RM). This is the reference model for the CIMI archetypes and the inclusion of CIMI RM helps in leveraging existing CIMI archetypes and provides consistent computational framework. One could start writing new archetypes with CIMI RM, reusing existing ones, developing translational tools, and creating platform specific implementation of the CIMI models.

I have contributed to the development of CIMI RM (which is provided with the released AML specification documents) and have used it as *the reference model* for implementing AML archetypes. The ADL archetypes, the source of these mappings, use the OpenEHR Reference Model as their reference model. This required additional task of making sure the classes in the OpenEHR RM map correctly to CIMI RM. Since CIMI

RM, inspired from OpenEHR RM, is practically a subset of OpenEHR RM makes the task of mappings RM classes trivial.

The AML Reference Model Profile (RMP) specifications guided me to create mappings related the RM and the RM class for an AML archetype. The AOM class ARCHETYPE_HRID (Figure 7) class represents ADL archetype identification elements and the RMP from the AML specifications (Figure 6) shows the AML stereotypes and tags used to reference a RM class. The first set of mappings related to archetype identification are listed in Table 1.

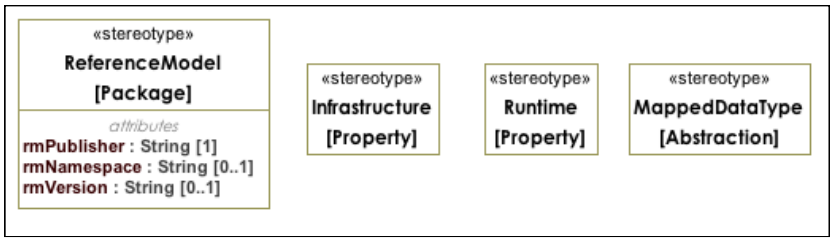


Figure 6: The AML Reference Model Profile

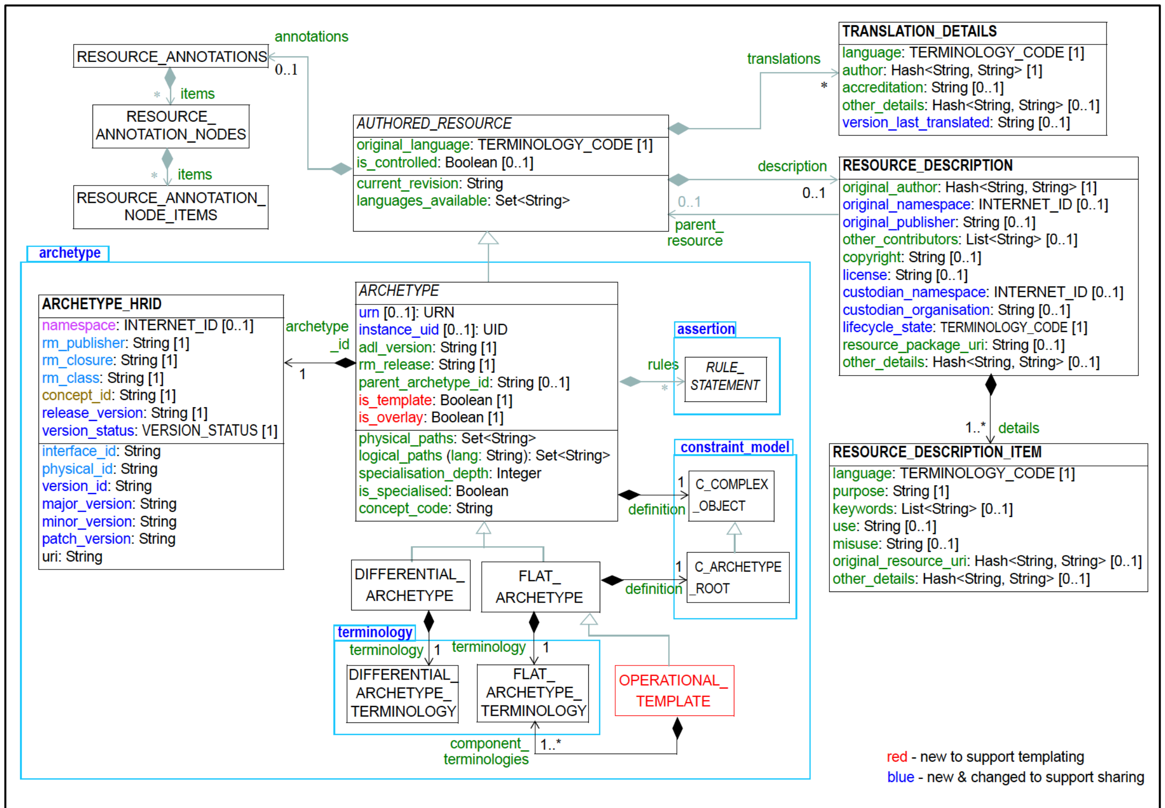
AOM	AML Profile	AML Profile Element
rm_publisher	*RMP*	*<<ReferenceModel>>::rmPublisher*
rm_closure	*CMP*	*<<ArchetypeLibrary>>::rm_package*
rm_class	*RMP*	Root/Defining class of the archetype. The name of the *<<ReferenceModel>>* Class specialized by this archetype
concept_id	*CMP*	*<<Archetype>>* Package name
release_version	*CMP*	*<<Archetype>>::release_version*
version_status	*CMP*	*<<Archetype>>::version_status*
build_count	*CMP*	*<<Archetype>>::build_count*
namespace	*CMP*	*<<Archetype>>* Package URI

Table 1: Archetype Identification mappings

The AML primitive type constraints are defined in terms of the *AML Primitive Types* even when the type is being considered is the Reference Model Type. The *MappedDataType* stereotype specifies the *AML Primitive Type* abstraction for a Reference Model Classifier. For the purpose of this transformation, I used The *AML Primitive types* that are defined by *The UML XML Primitive Types,* and map directly to the CIMI RM Primitive types.

Table 1 also shows the mapping to form namespace and the *physical_id* of an ADL in AML workspace. (See Appendix B on how to identify an ADL archetype). Another important part of mapping is to map hierarchical relationships among archetypes and its constraint elements. Fortunately an AML archetype specializes another AML archetype in exactly the same way the UML classes are specialized. This is another advantage of working with AML's UML representation. This object-oriented nature of UML classes eliminates the need for complicated identification scheme that ADL employs to identify the specializations of constraint elements.

The *ResourceTranslation* stereotype of CMP models the language specific terminology definitions and metadata about the translations in the archetype context. I plan to include implementation of mappings related to *ResourceTranslation*, which maps more or less directly from its ADL counterpart, in future releases.



RESOURCE_ANNOTATIONS
annotations
0..1
items
RESOURCE_ANNOTATION_NODES
items
RESOURCE_ANNOTATION_NODE_ITEMS
AUTHORED_RESOURCE
original_language: TERMINOLOGY_CODE [1]
is_controlled: Boolean [0..1]
current_revision: String
languages_available: Set<String>
translations
TRANSLATION_DETAILS
language: TERMINOLOGY_CODE [1]
author: Hash<String, String> [1]
accreditation: String [0..1]
other_details: Hash<String, String> [0..1]
version_last_translated: String [0..1]
description
0..1
parent_resource
RESOURCE_DESCRIPTION
original_author: Hash<String, String> [1]
original_namespace: INTERNET_ID [0..1]
original_publisher: String [0..1]
other_contributors: List<String> [0..1]
copyright: String [0..1]
license: String [0..1]
custodian_namespace: INTERNET_ID [0..1]
custodian_organisation: String [0..1]
lifecycle_state: TERMINOLOGY_CODE [1]
resource_package_uri: String [0..1]
other_details: Hash<String, String> [0..1]
1..* details
RESOURCE_DESCRIPTION_ITEM
language: TERMINOLOGY_CODE [1]
purpose: String [1]
keywords: List<String> [0..1]
use: String [0..1]
misuse: String [0..1]
original_resource_uri: Hash<String, String> [0..1]
other_details: Hash<String, String> [0..1]
archetype
ARCHETYPE_HRID
namespace: INTERNET_ID [0..1]
rm_publisher: String [1]
rm_closure: String [1]
rm_class: String [1]
concept_id: String [1]
release_version: String [1]
version_status: VERSION_STATUS [1]
interface_id: String
physical_id: String
version_id: String
major_version: String
minor_version: String
patch_version: String
uri: String
archetype_id
1
ARCHETYPE
urn [0..1]: URN
instance_uid [0..1]: UID
adl_version: String [1]
rm_release: String [1]
parent_archetype_id: String [0..1]
is_template: Boolean [1]
is_overlay: Boolean [1]
physical_paths: Set<String>
logical_paths (lang: String): Set<String>
specialisation_depth: Integer
is_specialised: Boolean
concept_code: String
assertion
rules
RULE_STATEMENT
constraint_model
definition
1
C_COMPLEX_OBJECT
C_ARCHETYPE_ROOT
DIFFERENTIAL_ARCHETYPE
FLAT_ARCHETYPE
terminology
1
DIFFERENTIAL_ARCHETYPE_TERMINOLOGY
FLAT_ARCHETYPE_TERMINOLOGY
OPERATIONAL_TEMPLATE
component_terminologies
1..*
red - new to support templating
blue - new & changed to support sharing

FIGURE 8 openehr.am.archetype Package

Figure 7: The AOM Archetype Package [© 2014 OpenEHR Foundation]

2.2 Constraint mappings

The AML CMP enables defining elements of an AML archetype that correspond to AOM's Metadata and the Constraint Definition sections. I used the AML CMP Stereotypes (Figure 8) to implement the following relationships among AML stereotypes. These archetype organizational restrictions, which are now part of the AML specifications, are results of the observations that were made by me and the members of CIMI. I contributed to their formation, while working with the CIMI Modeling Task Force. Their faithful implementation is critical to represent ADL constraints correctly in AML.

- The *ArchetypeLibrary* contains the *Archetype*s and each *ArchetypeLibrary* imports exactly one *ReferenceModel* package.
- All archetypes contained in the *ArchetypeLibrary* constrain classifiers and properties of the same referenced *ReferenceModel*.
- The *ObjectConstraint* stereotype in the CMP is the base stereotype to model a restriction on types, cardinalities, possible values and/or other aspects of a UML *NamedElement*. If the *ObjectConstraint* applies to a UML Property, the UML Property may specify the set of permissible values it can have, type and/or cardinality subject to the UML Property subset/redefinition semantics.
- The *ComplexObjectConstraint* stereotype is a classifier for constraining a Reference Model Classifier, and may constrain the existence, cardinality and/or possible values of any or all of the constrained Reference Model Classifier attributes. When an attribute has a type of another *ComplexObjectConstraint* in another archetype package, the *ArchetypeRoot* classifier represents it. The *ArchetypeRoot* is a specialization of the *ComplexObjectConstraint* in which an external *Archetype* is being 'reused'.
- The *ArchetypeSlot* stereotype is a specialization of the *ObjectConstraint* that allows an archetype to have a composition relationship with any number of archetypes matching some constraint pattern.

Table 2 lists constraints modeling mappings deduced by me between the AOM *Constraint Model* Package elements (Figure 9) and the AML CMP (Figure 8).

AOM	AML CMP
Association between *ARCHETYPE* and *C_COMPLEX_OBJECT*	*ArchetypeDefinition* associates a *ComplexObjectConstraint* supplier with an *Archetype* Client
C_ARCHETYPE_ROOT	*ArchetypeRoot*
ARCHETYPE_SLOT	*ArchetypeSlot*
AUTHORED_RESOURCE, RESOURCE_DESCRIPTION	*AuthoredResource*
C_OBJECT/rm_type_name	*Constrains* (when RM Classifier is constrained)
ARCHETYPE_CONSTRAINT/parent	*Constrains* (when parent archetype is constrained)
RESOURCE_ANNOTATIONS	*ResourceAnnotationNodeItem*
TRANSLATION_DETAILS, RESOURCE_DESCRIPTION_ITEM	*ResourceTranslation*
C_COMPLEX_OBJECT	*ComplexObjectConstraint*
C_ATTRIBUTE/rm_attr_name	Reference Model attribute that is being constrained
C_ATTRIBUTE properties	Owned Properties of *ComplexObjectConstraint*

Table 2: The Archetype Constraints mappings

2.3 Terminology binding mappings

The AML TBP binds model elements to their *meaning* by associating them to either locally defined (within the archetype) terms or the terms of an external (likely a standard) terminology. The AML TBP (Figure 11) is the UML equivalent to the ADL's Terminology Section. Figure 10 shows an example of populated AOM objects with terminology binding values for the term identifiers declared in the example archetype. Table 3 describes the mappings deduced between AOM terminology binding section and the AML TBP.

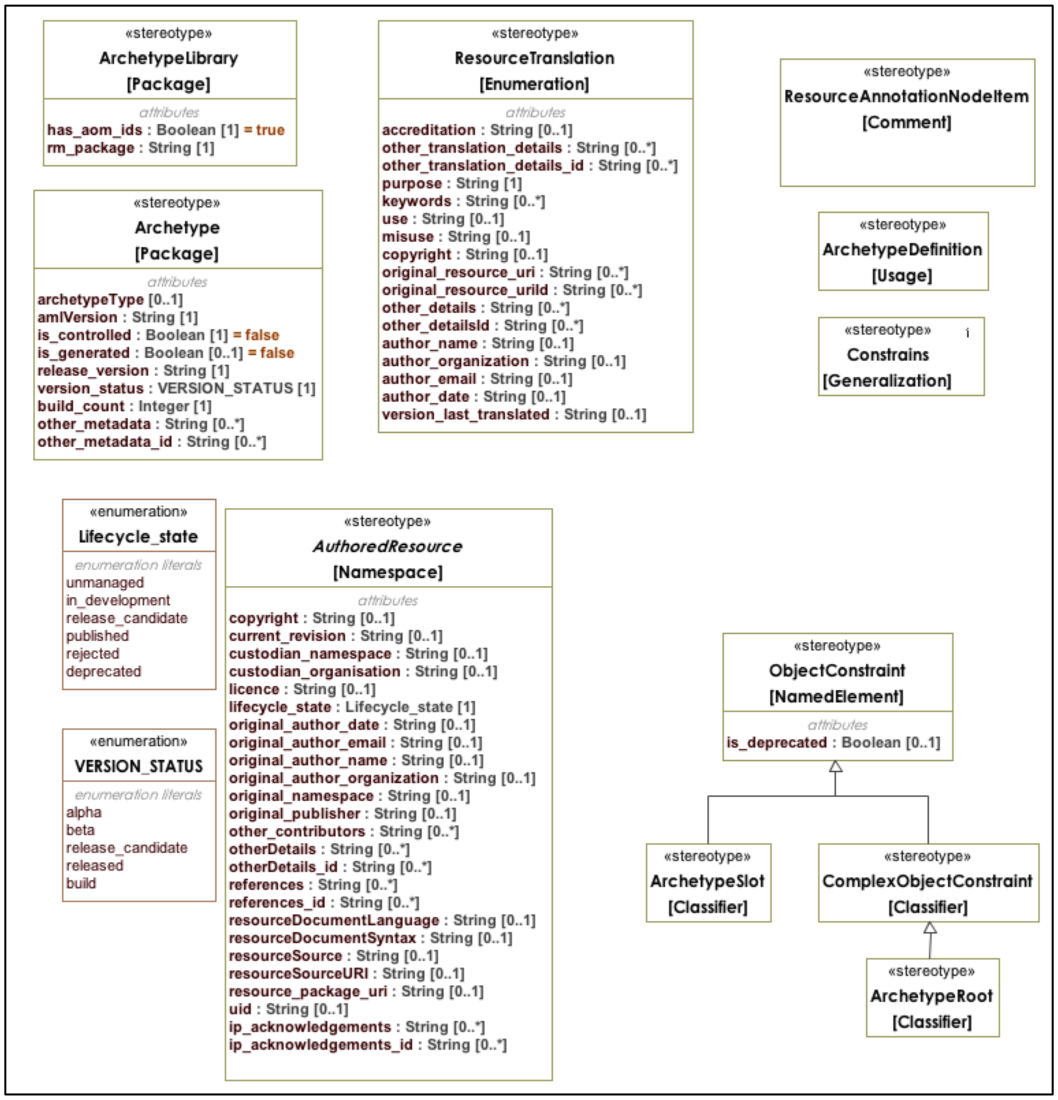

«stereotype»
ArchetypeLibrary
[Package]
attributes
has_aom_ids : Boolean [1] = true
rm_package : String [1]
«stereotype»
ResourceTranslation
[Enumeration]
attributes
accreditation : String [0..1]
other_translation_details : String [0..*]
other_translation_details_id : String [0..*]
purpose : String [1]
keywords : String [0..*]
use : String [0..1]
misuse : String [0..1]
copyright : String [0..1]
original_resource_uri : String [0..*]
original_resource_urild : String [0..*]
other_details : String [0..*]
other_detailsld : String [0..*]
author_name : String [0..1]
author_organization : String [0..1]
author_email : String [0..1]
author_date : String [0..1]
version_last_translated : String [0..1]
«stereotype»
ResourceAnnotationNodeItem
[Comment]
«stereotype»
Archetype
[Package]
attributes
archetypeType [0..1]
amlVersion : String [1]
is_controlled : Boolean [1] = false
is_generated : Boolean [0..1] = false
release_version : String [1]
version_status : VERSION_STATUS [1]
build_count : Integer [1]
other_metadata : String [0..*]
other_metadata_id : String [0..*]
«stereotype»
ArchetypeDefinition
[Usage]
«stereotype»
Constrains
[Generalization]
«enumeration»
Lifecycle_state
enumeration literals
unmanaged
in_development
release_candidate
published
rejected
deprecated
«stereotype»
AuthoredResource
[Namespace]
attributes
copyright : String [0..1]
current_revision : String [0..1]
custodian_namespace : String [0..1]
custodian_organisation : String [0..1]
licence : String [0..1]
lifecycle_state : Lifecycle_state [1]
original_author_date : String [0..1]
original_author_email : String [0..1]
original_author_name : String [0..1]
original_author_organization : String [0..1]
original_namespace : String [0..1]
original_publisher : String [0..1]
other_contributors : String [0..*]
otherDetails : String [0..*]
otherDetails_id : String [0..*]
references : String [0..*]
references_id : String [0..*]
resourceDocumentLanguage : String [0..1]
resourceDocumentSyntax : String [0..1]
resourceSource : String [0..1]
resourceSourceURI : String [0..1]
resource_package_uri : String [0..1]
uid : String [0..1]
ip_acknowledgements : String [0..*]
ip_acknowledgements_id : String [0..*]
«enumeration»
VERSION_STATUS
enumeration literals
alpha
beta
release_candidate
released
build
«stereotype»
ObjectConstraint
[NamedElement]
attributes
is_deprecated : Boolean [0..1]
«stereotype»
ArchetypeSlot
[Classifier]
«stereotype»
ComplexObjectConstraint
[Classifier]
«stereotype»
ArchetypeRoot
[Classifier]

Figure 8: The AML Constraint Model Profile

AOM Terminology Section		AML TBP
Identifiers	"id" : Identifiers	Constraint Class
	"at" : Permissible values in a value-set	*Enumeration Literals*
	"ac" : Value-set	*Enumeration*
Term Definitions		*Resource Translation*, *Entry* and *IdEntry*
Term Bindings		*Enumeration* containing *Enumeration Literals* for "ac" and "at" codes. Each *Enumeration Literal* may have a *Concept Reference* in external terminology.

Table 3: The Terminology Section Mapping

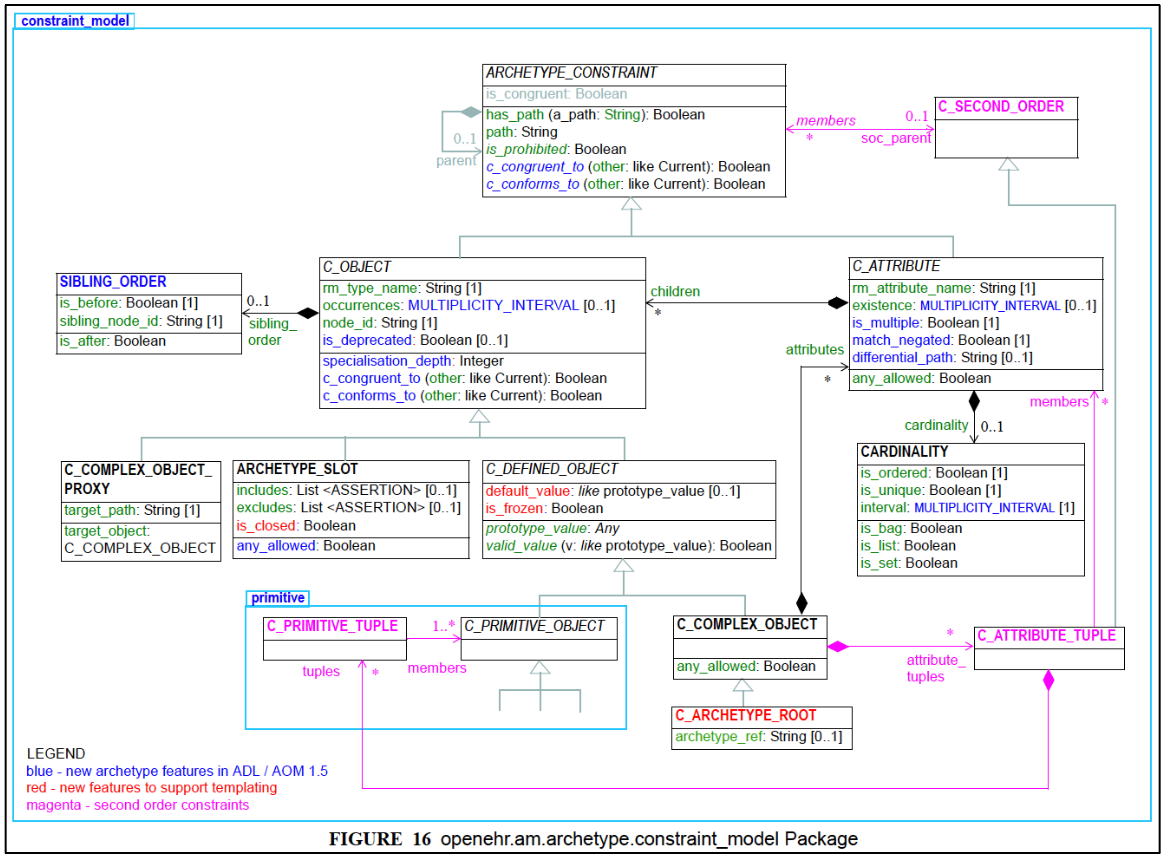

constraint_model
ARCHETYPE_CONSTRAINT
is_congruent: Boolean
has_path (a_path: String): Boolean
path: String
is_prohibited: Boolean
c_congruent_to (other: like Current): Boolean
c_conforms_to (other: like Current): Boolean
0..1
parent
members
*
0..1
soc_parent
C_SECOND_ORDER
SIBLING_ORDER
is_before: Boolean [1]
sibling_node_id: String [1]
is_after: Boolean
0..1
sibling_
order
C_OBJECT
rm_type_name: String [1]
occurrences: MULTIPLICITY_INTERVAL [0..1]
node_id: String [1]
is_deprecated: Boolean [0..1]
specialisation_depth: Integer
c_congruent_to (other: like Current): Boolean
c_conforms_to (other: like Current): Boolean
children
*
C_ATTRIBUTE
rm_attribute_name: String [1]
existence: MULTIPLICITY_INTERVAL [0..1]
is_multiple: Boolean [1]
match_negated: Boolean [1]
differential_path: String [0..1]
any_allowed: Boolean
attributes
*
members
*
cardinality
0..1
CARDINALITY
is_ordered: Boolean [1]
is_unique: Boolean [1]
interval: MULTIPLICITY_INTERVAL [1]
is_bag: Boolean
is_list: Boolean
is_set: Boolean
C_COMPLEX_OBJECT_
PROXY
target_path: String [1]
target_object:
C_COMPLEX_OBJECT
ARCHETYPE_SLOT
includes: List <ASSERTION> [0..1]
excludes: List <ASSERTION> [0..1]
is_closed: Boolean
any_allowed: Boolean
C_DEFINED_OBJECT
default_value: like prototype_value [0..1]
is_frozen: Boolean
prototype_value: Any
valid_value (v: like prototype_value): Boolean
primitive
C_PRIMITIVE_TUPLE
1..*
C_PRIMITIVE_OBJECT
tuples
*
members
C_COMPLEX_OBJECT
any_allowed: Boolean
*
attribute_
tuples
C_ATTRIBUTE_TUPLE
C_ARCHETYPE_ROOT
archetype_ref: String [0..1]
LEGEND
blue - new archetype features in ADL / AOM 1.5
red - new features to support templating
magenta - second order constraints
FIGURE 16 openehr.am.archetype.constraint_model Package

Figure 9: The AOM Constraint Model Package [© 2014 OpenEHR Foundation]

2.4 The Implementation

This section describes the third step of the approach that was taken to address the problem. Before I started writing the tool, the ADL, AOM and AML specifications were studied by me and the mappings between ADL and AML were identified after many discussions with CIMI MTF members. I described these mappings in previous sections and have implemented them in a programming library named ‘ADL2AMLConverter’. As planned, *ADL2AMLConverter* is a transform; a member of a growing set of tools organized under the parent project ‘AML Tooling’ is available at my GitHub repository [16].

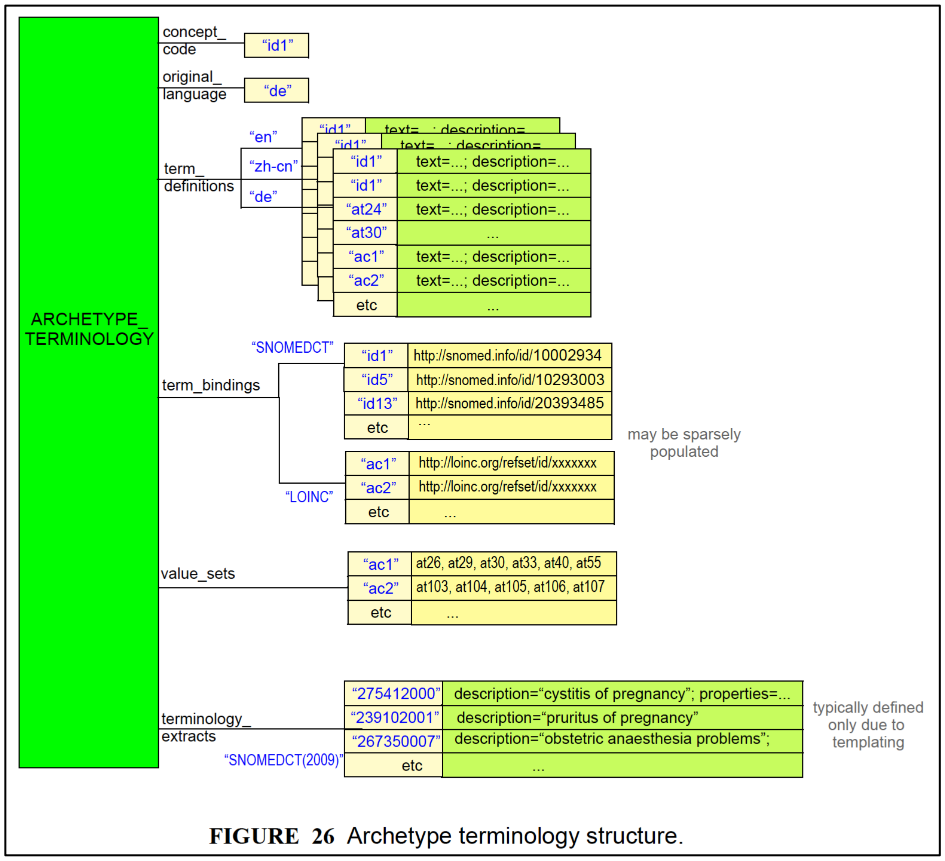

ARCHETYPE_TERMINOLOGY
concept_code
"id1"
original_language
"de"
term_definitions
"en"
"zh-cn"
"de"
"id1" text=...; description=...
"id1" text=...; description=...
"at24" text=...; description=...
"at30" ...
"ac1" text=...; description=...
"ac2" text=...; description=...
etc ...
term_bindings
"SNOMEDCT"
"id1" http://snomed.info/id/10002934
"id5" http://snomed.info/id/10293003
"id13" http://snomed.info/id/20393485
etc ...
"LOINC"
"ac1" http://loinc.org/refset/id/xxxxxxx
"ac2" http://loinc.org/refset/id/xxxxxxx
etc ...
may be sparsely populated
value_sets
"ac1" at26, at29, at30, at33, at40, at55
"ac2" at103, at104, at105, at106, at107
etc ...
terminology_extracts
"SNOMEDCT(2009)"
"275412000" description="cystitis of pregnancy"; properties=...
"239102001" description="pruritus of pregnancy"
"267350007" description="obstetric anaesthesia problems";
etc ...
typically defined only due to templating
FIGURE 26 Archetype terminology structure.

Figure 10: AOM Terminology Binding Example [© 2014 OpenEHR Foundation]

The AML Tooling Project can be downloaded, configured and used as either an embedded library for transforming the ADL archetypes into AML or creating new AML archetypes in UML. The AML Tooling currently has three sub-projects:

- *ADL2AMLConverter*: Converts ADL1.5 archetypes to AML archetypes. It utilizes the project *AML-MDLibrary* to create the AML objects in memory and eventually serialize archetypes in UML. A whole collection of the ADL archetypes can be fed to this converter and transformed into corresponding UML project creating an archetype library of them.
- *AML-MDLibrary*: A layer of convenience methods written over the MagicDraw Open API library for the easier creation of AML Objects in UML. This layer verifies the requirements to get a valid AML object created.

- *AMLMDPlugin*: This is a User Interface plugin to extend the UI capabilities of the MagicDraw UML Editor for the AML Object creation using the UI. This is an experimental project written to test the extensibility of the MagicDraw UML Editor. A complete implementation of this plugin is expected in future.

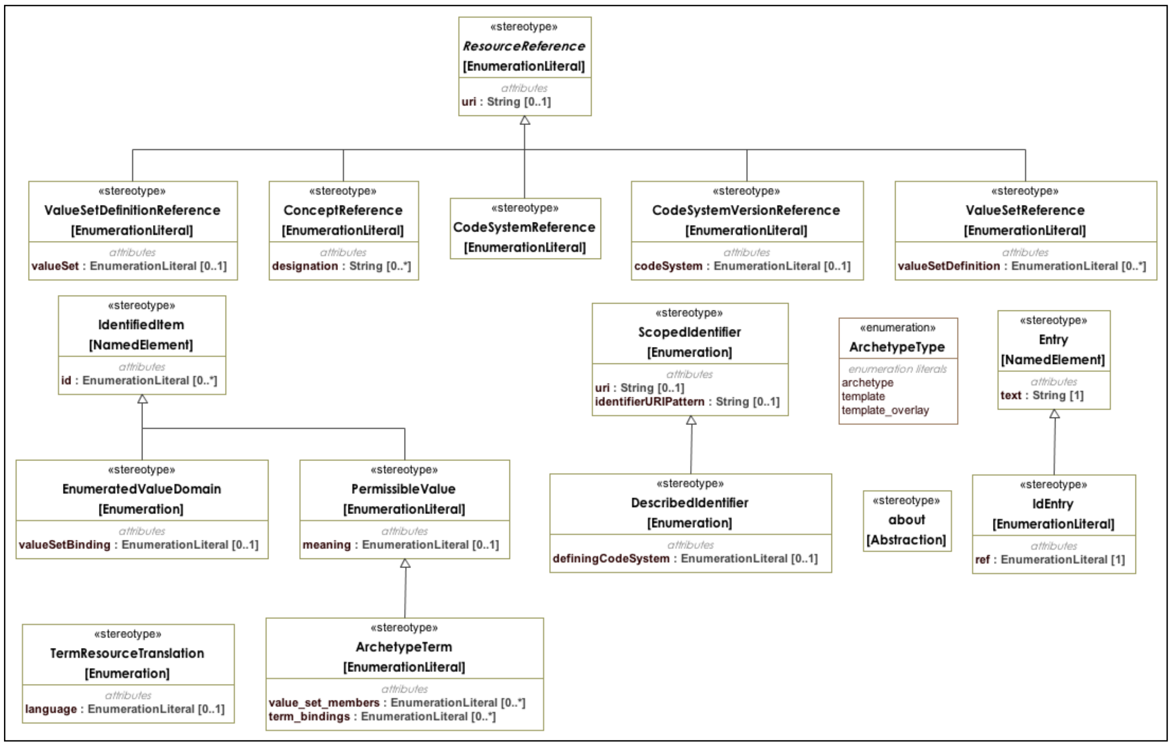

«stereotype» ResourceReference [EnumerationLiteral]
uri : String [0..1]
«stereotype» ValueSetDefinitionReference [EnumerationLiteral]
valueSet : EnumerationLiteral [0..1]
«stereotype» ConceptReference [EnumerationLiteral]
designation : String [0..*]
«stereotype» CodeSystemReference [EnumerationLiteral]
«stereotype» CodeSystemVersionReference [EnumerationLiteral]
codeSystem : EnumerationLiteral [0..1]
«stereotype» ValueSetReference [EnumerationLiteral]
valueSetDefinition : EnumerationLiteral [0..*]
«stereotype» IdentifiedItem [NamedElement]
id : EnumerationLiteral [0..*]
«stereotype» ScopedIdentifier [Enumeration]
uri : String [0..1]
identifierURIPattern : String [0..1]
«enumeration» ArchetypeType
archetype
template
template_overlay
«stereotype» Entry [NamedElement]
text : String [1]
«stereotype» EnumeratedValueDomain [Enumeration]
valueSetBinding : EnumerationLiteral [0..1]
«stereotype» PermissibleValue [EnumerationLiteral]
meaning : EnumerationLiteral [0..1]
«stereotype» DescribedIdentifier [Enumeration]
definingCodeSystem : EnumerationLiteral [0..1]
«stereotype» about [Abstraction]
«stereotype» IdEntry [EnumerationLiteral]
ref : EnumerationLiteral [1]
«stereotype» TermResourceTranslation [Enumeration]
language : EnumerationLiteral [0..1]
«stereotype» ArchetypeTerm [EnumerationLiteral]
value_set_members : EnumerationLiteral [0..*]
term_bindings : EnumerationLiteral [0..*]

Figure 11: The AML Terminology Binding Profile

For the development of the AML specifications, CIMI uses the AML GitHub Project [1] for managing the AML profiles, AML reference documents (normative and non-normative) for the mappings, the examples and the transformation scripts for the specifications themselves. These documents are great resources to find out more about the AML specifications and definitions of the terms used in this document.

2.5 The Development Environment

The open-source OpenEHR ADL Parser [13], used to parse the ADL1.5 and ADL2.0 archetypes, is provided by the OpenEHR foundation. The ADL parser libraries were downloaded and also hosted at Mayo Clinic's Apache Maven Repository [14] with OpenEHR's development team's permissions.

The MagicDraw Open APIs [15] library provides the reference implementation of the UML Essential Core (ECORE) Model and is used to create the AML objects in UML. The 'Traceability Matrix', which includes the mappings between the AOM and the AML Object Model (Tables 1, 2 and 3), is also part of the AML documentation [1]. Figure 12 shows the interaction among the components during the transformation of the ADL archetypes using the ADL2AML Converter.

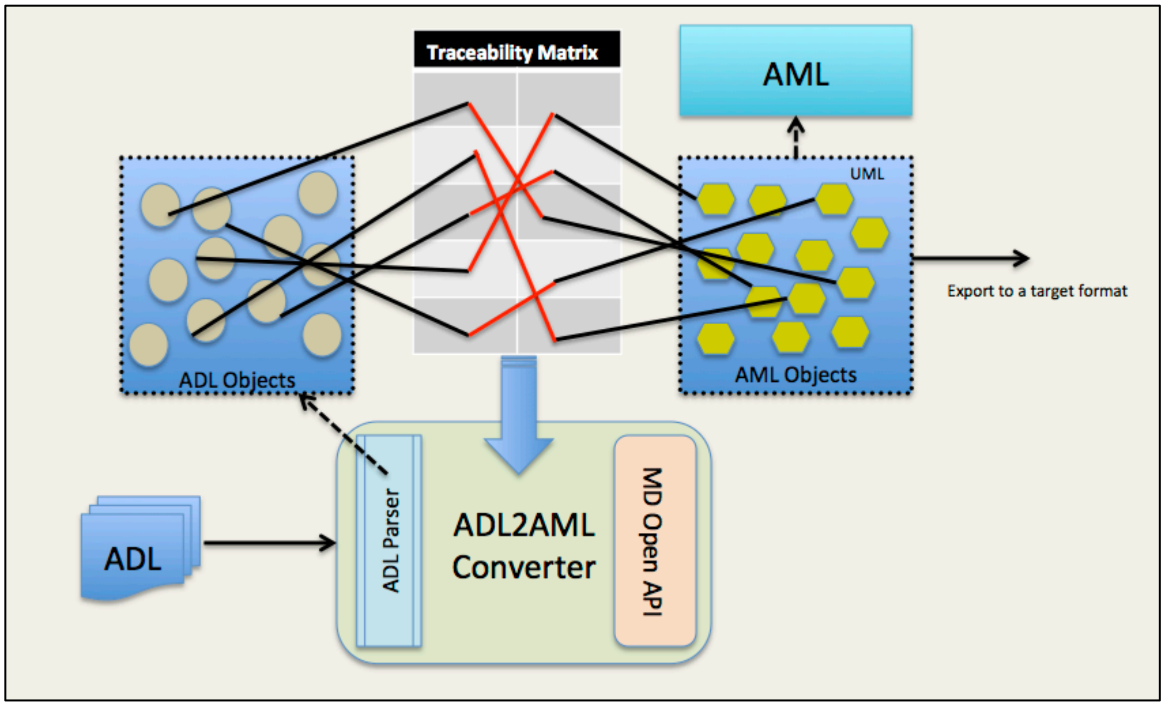


Figure 12: The ADL to AML Transform workflow

3 Conclusion

The support of robust set of tools is required for almost every standard to be used to its full potential in the real world. AML is a new OMG standard with great promise but lacks any effective tooling. At this time, manually creating archetypes from scratch in a UML editor, by just following AML specifications, is a challenging, time-consuming, tedious and error-prone process, even for an experienced UML modeler. While ADL workspace has accumulated many rich sets of clinical archetypes, it still lacks the benefits of AML archetypes, including smooth integration with other UML models and an easy gateway to model-driven architecture. I developed the mappings between ADL Object Model (AOM) and AML Object Model that created a way to bridge this gap between ADL and AML.

The AML Tooling's *ADL2AMLConverter*, developed by me, provides a way to transition between ADL and AML formats by implementing the mappings I catalogued. The proprietary format of ADL and its archetype-modeling environment do not provide such a transition. With the help of these mappings, archetype modelers can get instant access to rich sets of existing, well-established ADL archetypes and jump start their own set of AML archetypes, which they can create by either re-using the imported ones and/or extending them based on their requirements.

A modeler can opt to continue working with ADL, till AML authoring environment is created, and transform them to AML using this tool. The ADL modelers, who would like to have archetypes in AML, can use this tool as a 'short cut' to minimize their AML archetype modeling efforts. To create a new AML archetype, a new ADL archetype can be created first using ADL modeling tools and then converted into AML easily with this transform.

I tested the *ADL2AMLConverter* with OpenEHR ADL archetypes and was able to verify implementation of mappings for the archetype identification and constraints. Both types of ADL archetypes, *model-level* and *user-level,* were transformed using CIMI RM and successfully tested for the mappings and for the archetype *specialization*. As I manually validated a number of converted ADL archetypes, it helped me to investigate and question some of the choices that were made by CIMI MTF members for AML specifications.

The tooling, which is still evolving, already proved to be an effective tool to reduce the manual process to recreate and test new and existing ADL archetypes in AML workspace. This has been an excellent tool to quickly compare an ADL archetype with its AML re-incarnation to identify the translational gaps, that either AML specifications or the converter itself might have missed. The members of CIMI Modeling Task Force are using this tooling and plan to incorporate it in upcoming AML tooling related projects.

This is also a first step in the direction of creating more user-friendly interfaces for AML archetype authoring and maintenance. The collaborating organizations at CIMI already have started working towards this goal.

3.1 Limitations & Mitigations

- The tooling includes the mappings only for constraint definitions and archetype identification stereotypes. These two items have been the most important and critical part of getting the resulting archetype right. The implementation of mappings for terminology binding, archetype metadata and resource translations are still in development stage. The tooling is being developed to use the CTS2 terminology services [3] [4] for terminology binding. The completed implementation will allow any CTS2 terminology services to get integrated smoothly, if needed.
- The validation of the transformed AML archetypes was performed manually. I plan to devise a strategy for validating the resulting AML archetype for their complete correctness. One way to do this is to extend the implementation in both directions - ADL →AML and AML → ADL. This will allow the conversion of the transformed AML archetype back into its ADL form and it could be compared against the original ADL archetype.
- The converter depends on the MagicDraw OpenAPI libraries, though they are free and open-source, the installation of the MagicDraw Editor or its interface libraries is a prerequisite to make them work. I plan to extend the implementation to use the Eclipse Modeling Framework (EMF) [17] ECORE Reference Implementation and the Model Development Tools [18].
- The converted archetype library is in MagicDraw Project format, which can only be opened and viewed in the MagicDraw UML Editor. The solution for this problem is not part of the implementation yet, but can easily included by utilizing the UML

model export mechanism of MagicDraw to store the model in a standard format like XML Metadata Interchange (XMI) format.

3.2 Next Steps

3.2.1 Using ADL modeling tools upgrades

The modeling tools available at the OpenEHR (including ADL Parser and ADL Workbench) are in process of getting upgraded for the technical platforms of Java and XML and make them available to the developers in near future. These upgrades still do not bridge the gaps discussed in this thesis between ADL and AML, but this will enable the bidirectional archetype transformation between ADL and AML get implemented with greater ease. This will also enhance the capabilities of archetype validation and transforming them into various other formats with fewer efforts. I plan to extend the mappings for the bidirectional transformation.

3.2.2 Exploiting Model Driven Architecture

The creation of AML Objects is guided by the AML Specifications, which are interpreted manually and implemented in the AML tooling described in this thesis. I plan to exploit the AML specifications themselves, which are the UML model artifacts themselves, by feeding into a Model Driven Architecture (MDA) [19] based workflow. The AML Profiles and the Metamodel itself be utilized to validate the completeness of the AML archetypes (the would be instances of the meta model) will be greatly helpful and hugely simplify the subsequent development of the user libraries and GUI elements. This will also minimize the efforts of developing and maintaining the AML archetypes and templates.

3.2.3 Shape Expressions (ShEx)

The archetypes constraint validation of archetype instances is a challenging task ahead. An AML UML model can have constraints specified in the OMG OCL in addition to the instances of the AML constraint objects. The AML specification itself is a collection of numerous OCL constraints specified to create the details of the specifications. Unfortunately the lack of free, open-source and, robust implementation of OCL makes it challenging for consistent constraints application. There are proprietary implementations available, but adopting a particular one would force the users to acquire it and shrink the pool of potential users.

The ‘Shape Expression’ Language (ShEx) [20] is an upcoming solution by Eric Prud'hommeaux from World Wide Web Consortium (W3C). The ShEx is a language for expressing constraints on RDF Graphs. The Shape Expression Language provides a SPARQL [12] like constructs that are simpler and consistent way to writing constraints. There are already few ShEx implementations [21] available to validate RDF. The efforts are already underway to transform UML into ShEx and once one can successfully transform a UML resource (e.g. an AML archetype) into ShEx, the validation process is consistent and easier thereafter. Since an RDF feed is a popular and standard form of disseminating messages (here these would be instances of clinical models), one could apply the archetypes (in their ShEx form) to validate the instance data on the fly. I am a W3C RDF Working Group participant for the development of the Shape Expressions and related solutions. I am already working on transforming archetypes (ADL and AML) into the ShEx constructs (called *shapes*), which provide consistent representation of the constraints and a flexible way of associating the *semantic actions* along.

4 Bibliography

[1] AML Specifications artifacts at Github, https://github.com/opencimi/AML (Accessed August 18, 2015).

[2] ISO/IEC 11179-3:2013(E) Information technology, Metadata registries (MDR) - Part 3: Registry meta-model and basic attributes. International Organization for Standardization (ISO), February 2013, http://metadata-standards.org/11179/#A3 (Accessed August 18, 2015).

[3] CTS2 Terminology Services for SNOMED CT, written in Python, http://informatics.mayo.edu/py4cts2/ (Accessed August 18, 2015).

[4] TLAMP - CTS2 Terminology Services mainly for LOINC, http://tlamp.org (Accessed August 18, 2015).

[5] Enterprise Vocabulary Server (EVS) – LexEVS implementation at National Cancer Institute (NCI), https://wiki.nci.nih.gov/display/LexEVS/LexEVS (Accessed August 18, 2015).

[6] ADL Workbench release 2.0.5, OpenEHR, http://www.openehr.org/downloads/ADLworkbench/home (Accessed August 18, 2015).

[7] OpenEHR – Home, http://www.openehr.org/ (Accessed August 18, 2015).

[8] Ocean Informatics – Home, http://oceaninformatics.com (Accessed August 18, 2015).

[9] Clinical Knowledge Manager (CKM), OpenEHR, http://www.openehr.org/ckm (Accessed August 18, 2015).

[10] Clinical Element Model (CEM) Browser, Intermountain HealthCare (IHC), http://www.opencem.org/#/ (Accessed August 18, 2015).

[11] Detailed Clinical Models (DCM), Health Level 7 (HL7), http://wiki.hl7.org/index.php?title=Detailed_Clinical_Models (Accessed August 18, 2015).

[12] SPARQL Query Language for RDF, World Wide Web Consortium (W3C), http://www.w3.org/TR/rdf-sparql-query/ (Accessed August 18, 2015).

[13] OpenEHR Archetype Definition Language (ADL) parser, https://github.com/openEHR/adl2-core (Accessed August 18, 2015).

[14] OpenEHR ADL Parser at the Mayo Clinic's Maven public repository, http://informatics.mayo.edu/maven (Accessed August 18, 2015).

[15] MagicDraw Open APIs, NoMagic. Inc., http://www.nomagic.com/files/manuals/MagicDraw_OpenAPI_UserGuide_1701.pdf (Accessed August 18, 2015).

[16] The AML Tooling Project at GitHub, https://github.com/semantix/AMLTooling (Accessed August 18, 2015).

[17] The Eclipse Modeling Framework (EMF), https://eclipse.org/modeling/emf/ (Accessed August 18, 2015).

[18] The Eclipse Modeling Development Tools, https://eclipse.org/modeling/mdt/?project=uml2 (Accessed August 18, 2015).

[19] The Model Driven Architecture (MDA), http://www.omg.org/mda/ (Accessed August 18, 2015).

[20] The Shape Expression, W3C, http://www.w3.org/2001/sw/wiki/ShEx (Accessed August 18, 2015).

[21] The RDF Validator using Shape Expressions, https://github.com/labra/rdfshape (Accessed August 18, 2015).

5 Appendices

A. CIMI Reference Model

CIMI RM is relatively smaller and generic model for the creating archetypes. CIMI RM is described with the help of Table 4 and the snapshots of the UML model for each package to help explain the archetype example in the following sub-sections.

Primitive Types	The basic types that are assumed in external type systems. This package is a guide for integrating models into the type systems of implementation technologies.
Data Value Types	Includes a set of clearly defined data types, which underlies all other models. These data types provide both general and clinically specific types required for all kinds of health information.
Core Model	Includes the main classes in the CIMI reference model upon which clinical models will be defined.
Party Model	Defines the generic concepts of PARTY, ROLE and related details, which allow archetypes for any type of person, organization, role and role relationship to be described. This approach provides a flexible way of including the arbitrary demographic attributes that may be required.

Table 4: The CIMI RM Packages

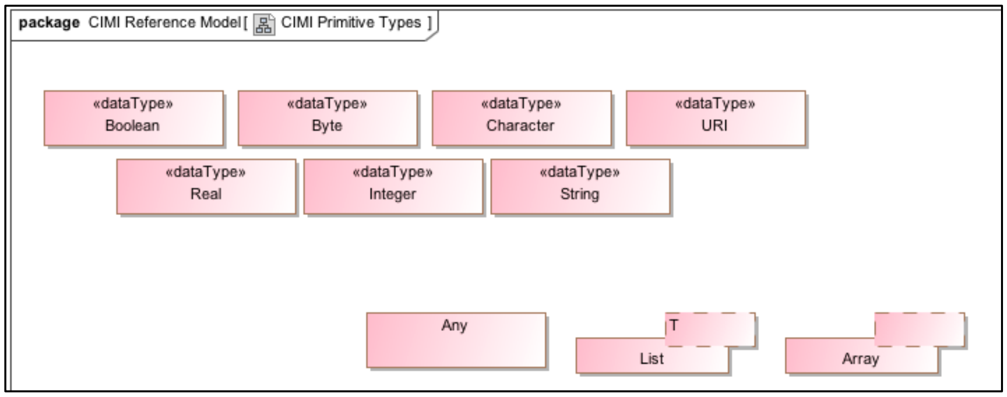


Figure 13: CIMI RM Primitive Types

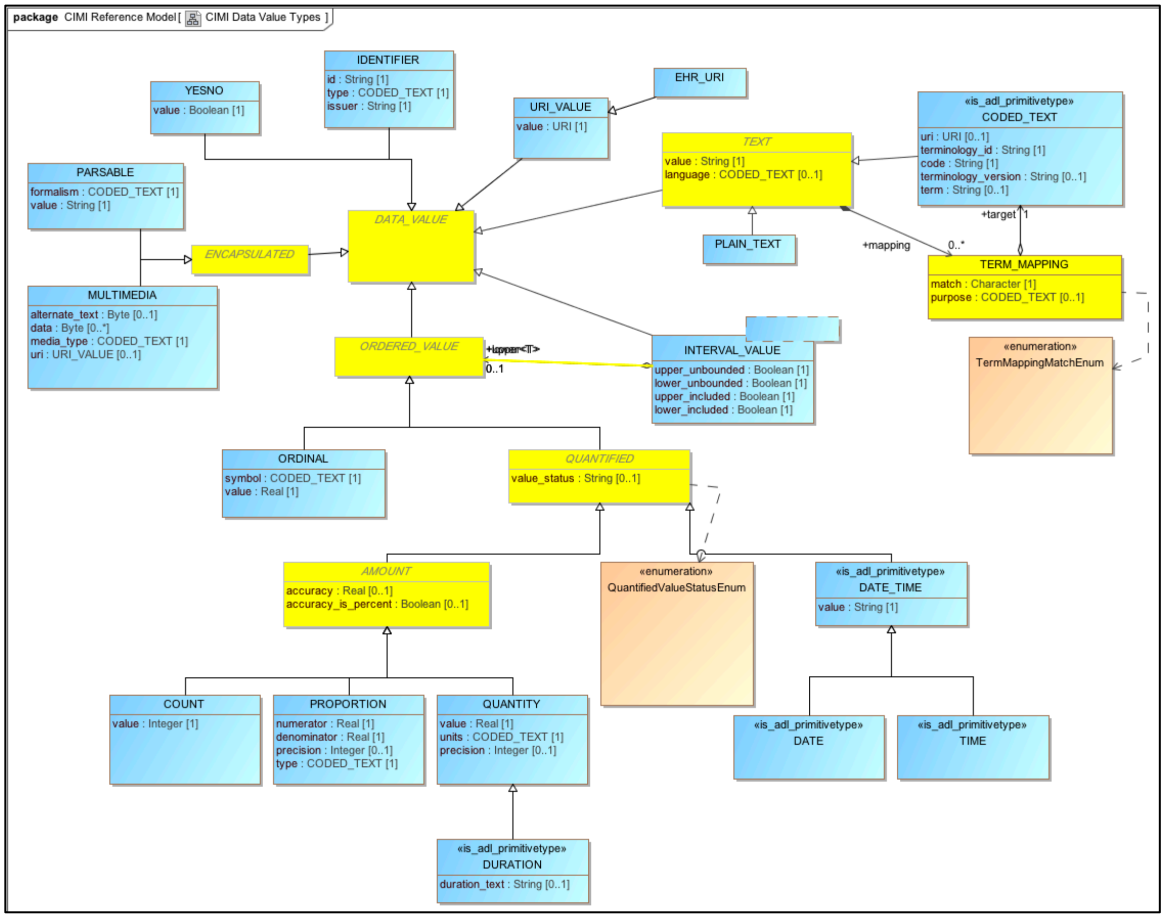


Figure 14: CIMI RM Data Value Types

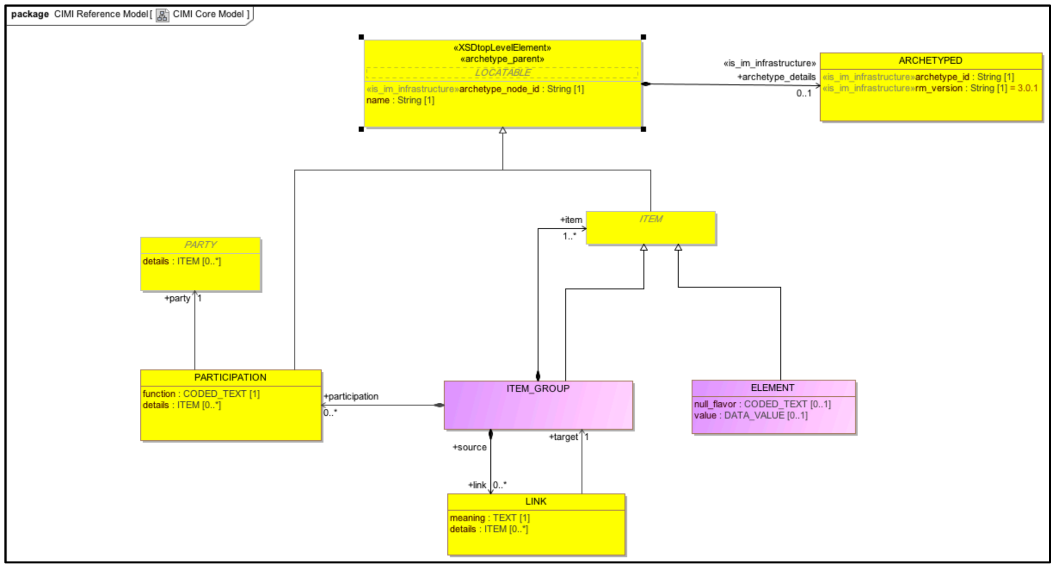


Figure 15: CIMI RM Core Model

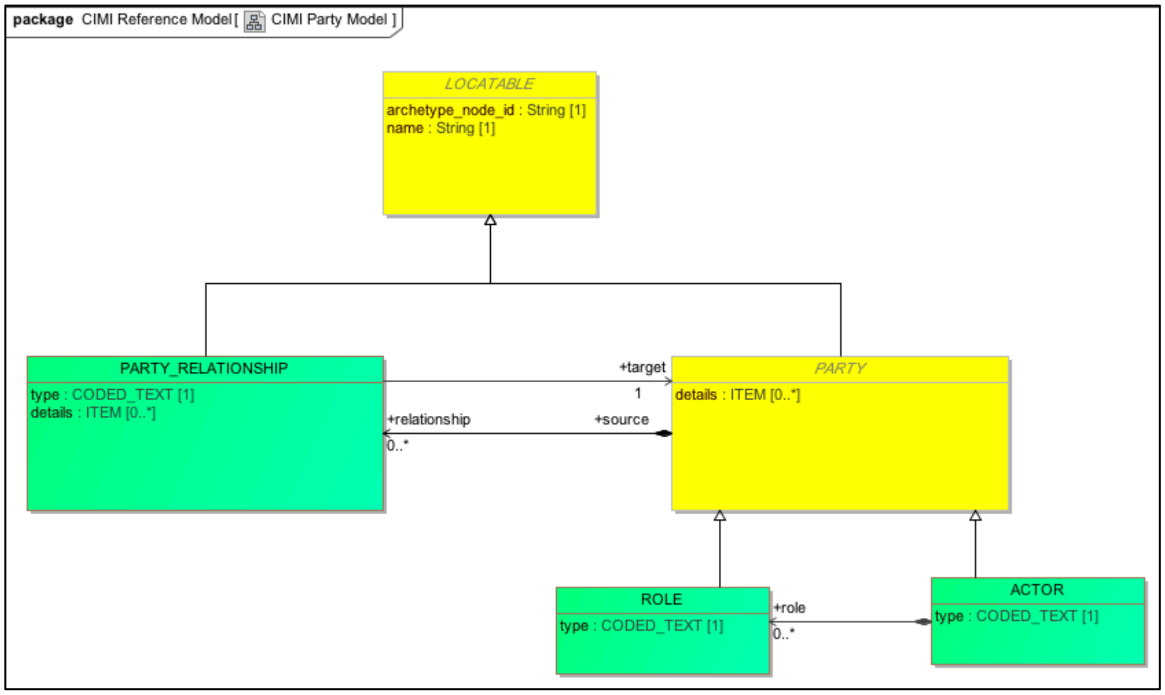


Figure 16: CIMI RM Party Model

B. Identifying an ADL archetype

An ADL archetype's identification is combination of multiple attribute values, to indicate its location in a hierarchical organization of archetypes, as follows (Figure 17, which is reproduced from the AOM specification's figure 9):

[rm_publisher]-[rm_closure]-[rm_class].[concept_id].v[release_version]-[version_status].[build_count]

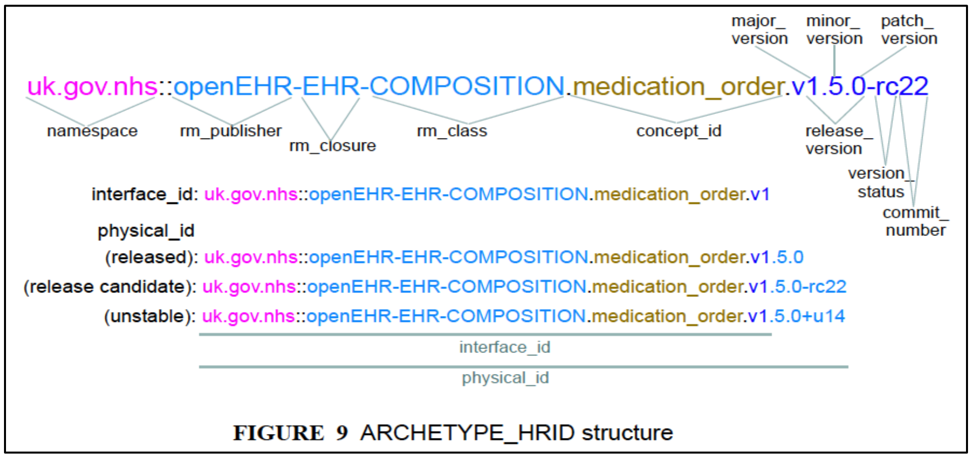

major_ version
minor_ version
patch_ version
uk.gov.nhs::openEHR-EHR-COMPOSITION.medication_order.v1.5.0-rc22
namespace
rm_publisher
rm_closure
rm_class
concept_id
release_ version
version_ status
commit_ number
interface_id: uk.gov.nhs::openEHR-EHR-COMPOSITION.medication_order.v1
physical_id
(released): uk.gov.nhs::openEHR-EHR-COMPOSITION.medication_order.v1.5.0
(release candidate): uk.gov.nhs::openEHR-EHR-COMPOSITION.medication_order.v1.5.0-rc22
(unstable): uk.gov.nhs::openEHR-EHR-COMPOSITION.medication_order.v1.5.0+u14
interface_id
physical_id
FIGURE 9 ARCHETYPE_HRID structure

Figure 17: The archetype identification [© 2014 OpenEHR Foundation]

The name of an archetype is specified in the attribute *physical_id* of AOM class *ARCHETYPE_HRID*. Figure 7 (reproduced from the AOM specification's figure 8) shows the attributes of the *ARCHETYPE* & *ARCHETYPE_HRID* Classes to form the archetype identification.

C. Metadata Registries (MDR) - ISO/IEC 11179

The international standard, *ISO/IEC – 11179, Information Technology – Metadata Registries (MDR) Part 3: Registry Metamodel and basic attributes* [2], provides the common understanding of the meaning and the representation of the data to both its owner and users. To facilitate this common understanding, the metadata has specification for metadata items for the purpose of identification, designation, definition and value-meaning linkage.

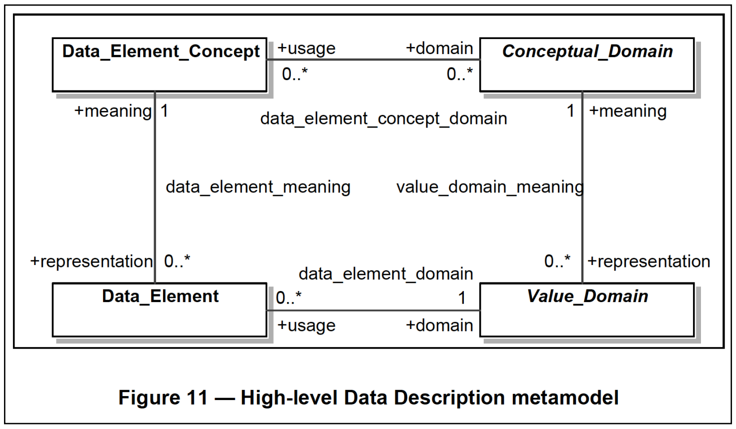


Figure 18: Meaning and representation of data [© ISO/IEC 11179 -3]

The MDR Model helps understand, present and share data in a form that combines 'what the data is' (meaning) and 'how it is presented' (representation). Figure 18, reproduces Figure 11 of MDR specifications, shows the metadata elements for *the meaning* (upper part of the figure) and *the representation* (lower part of the figure).

D. Common Terminology Services 2 (CTS2)

The Interoperability is directly related with the semantics attached to the data being exchanged – association with identical term results into same understanding. A terminology serves as the common language between the systems, and is one of the keys to solving the problem of interoperability. The terminologies have coded entries, which are referenced in clinical models and medical records. Some of the examples of terminologies are the SNOMED CT, ICD-10, and LOINC among others.

The Common Terminology Services 2 (CTS2) is a collection of the OMG standards for representing, accessing, and disseminating the terminological content. CTS2 is built on the Representational State Transfer (REST) architecture and its specifications lists interfaces for creating and accessing terminology resources like the Code Systems, Code System Versions, Entity Description, Association, and Value-Sets. There are various CTS2 implementations available, e.g. the CTS2 Services for SNOMED CT [3] and TLAMP [4], which are being used for creating the terminology bindings in archetypes. The CTS2 terminological content can be accessed in three main formats – XML, JSON, and RDF. The detailed information on the CTS2 and its implementations can be found at the Mayo Clinic's CTS2 site - `http://informatics.mayo.edu/cts2`.

E. An Example of Transformed archetype

This section discusses an archetype with few sample constraints using the reference model classes of the CIMI RM. The CIMI RM comes packaged with the AML specifications [1] and it serves as an appropriate one to use in this example.

This hypothetical example shows an archetype named as *adult_patient*, which is defined in terms of the CIMI RM model classes, to model a clinical patient with few details. This example is only for the purpose of demonstrating how to record constraints in the ADL and AML archetypes. Lets say the archetype *adult_patient* has following details:

- An identifier defined for the patient - *subjectID*

- A value for patient age - *age*
    - If specified has to be greater than *17* years
    - If not specified then age value is coded as either *Missing* (encoded with string value *9999*) or *Not Assessed* (encoded with string value *1000*).
- A value for patient gender – *patientGender*

  The gender value comes from a set of enumerated values:

  {*Female*, *Male*, *Not Specified*, *Unknown*}
- The SNOMED-CT is used to bind the terms. (It is to be noted that the terms identifiers used in this example are not real resource identifiers and may or may not map to the relevant concept in SNOMED-CT).

<u>*The ADL Model*</u>: The ADL Workbench [6] is used to create this sample archetype. The CIMI RM version 2.0.2 was used. This archetype takes CIMI RM class *ITEM_GROUP* and narrow down the instances of the *ITEM_*GROUP by adding constraints for the members. The restrictions on the patient (for identifier, age and gender) are not of grouping nature (like *ITEM_GROUP*), so the RM Model class *ELEMENT* class fits to specify these constraints. A possible serialized version of the ADL text for this example might look like the following:

```
archetype (adl_version=2.0.5; rm_release=2.0.2)
	CIMI-Core-ITEM_GROUP.adult_patient.v0.0.1
language
	original_language = <[ISO_639-1::en]>
description
	lifecycle_state = <"unmanaged">
	original_author = <
		["name"] = <"Deepak Sharma">
		["organisation"] = <"Mayo Clinic">
		["email"] = <"sharma.deepak2@mayo.edu">
		["date"] = <"2015-07-12">>
	custodian_namespace = <"edu.mayo">
	custodian_organisation = <"Mayo Clinic <http://www.mayo.edu>">
	copyright = <"Copyright © 2015 Mayo Clinic <http://www.mayo.edu>">
	licence = <"Creative Commons<https://creativecommons.org/licenses/by-sa/3.0/>">
	details = <
		["en"] = <
			language = <[ISO_639-1::en]>
			purpose = <"Demonstrates a simple ADL Archetype for a ">
			keywords = <"ADL", "adult", "patient">
		>
	 >
definition
	ITEM_GROUP[id1] matches {
		item matches {
			ELEMENT[id2] occurrences matches {1}
			ELEMENT[id3] occurrences matches {0..1} matches {
				value matches {
					COUNT[id4] occurrences matches {0..1} matches {
						value matches {|>17|}
					}
					CODED_TEXT[id5]
					CODED_TEXT[id6]
				}
			}
			ELEMENT[id7] occurrences matches {0..1} matches {
				value  matches {[ac1]}
			}
		}
	}


terminology
	term_definitions = <
		["en"] = <
			["id1"] = <
				text = <"adultPatent">
				description = <"Adult Patient">
			>
			["id2"] = <
				text = <"subjectID">
				description = <"Patient Identifier">
			>
			["id3"] = <
				text = <"age">
				description = <"Patient Age">
			>
			["id4"] = <
				text = <"ageInterval">
				description = <"Patient Age Interval">
			>
			["id5"] = <
				text = <"9999">
				description = <"Missing">
			>
			["id6"] = <
				text = <"1000">
				description = <"Not Assesssed">
```

```
                      >
                      ["id7"] = <
                              text = <"patientGender">
                              description = <"Patient Gender">
                      >
                      ["at11"] = <
                              text = <"F">
                              description = <"Female">
                      >
                      ["at12"] = <
                              text = <"M">
                              description = <"Male">
                      >
                      ["at13"] = <
                              text = <"NS">
                              description = <"Not Specified">
                      >
                      ["at14"] = <
                              text = <"UNK">
                              description = <"Unknown">
                      >
                      ["ac1"] = <
                              text = <"sexes">
                              description = <"xxx">
                      >
              >
        >
term_bindings = <
              ["snomed-ct"] = <
                      ["id2"] = <http://snomed.info/id/118522005>
                      ["id3"] = <http://snomed.info/id/424144002>
                      ["id4"] = <http://snomed.info/id/2667000>
                      ["id5"] = <http://snomed.info/id/255582007>
                      ["id6"] = <http://snomed.info/id/424144012>
                      ["id7"] = <http://snomed.info/id/703117000>
                      ["at11"] = <http://snomed.info/id/385432009>
                      ["at12"] = <http://snomed.info/id/394743007>
                      ["at13"] = <http://snomed.info/id/385432009>
                      ["at14"] = <http://snomed.info/id/394743007>
              >
        >

value_sets = <
              ["ac1"] = <
                      id = <"ac1">
                      members = <"at11", "at12", "at13", "at14">
              >
        >
```

Table 5: The example archetype serialized in the ADL format

The equivalent sets of constraints were transformed from the ADL format to the corresponding constructs for constraints using the ADL2AML Converter. For the purpose of clarity the resulting UML diagram of the AML archetype of *adult_patient* was edited manually.

Since an AML archetype is rendered in the UML form, the archetype is defined using the UML entities like the Package annotated with stereotypes to organize an archetype library and the archetype, the stereotyped UML Classes representing the constraints and the stereotyped UML Enumerations for the coded values. The following figures show the transformed archetype and its components.

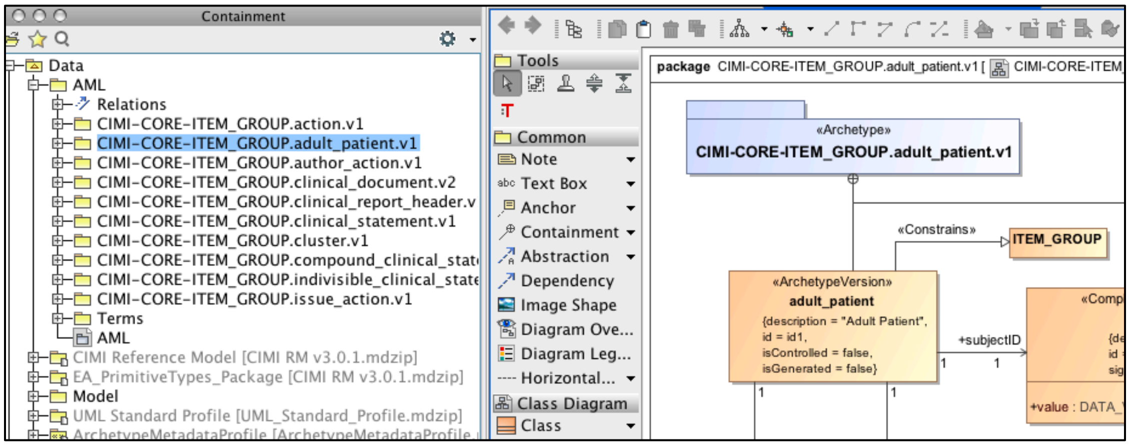

Containment
Data
AML
Relations
CIMI-CORE-ITEM_GROUP.action.v1
CIMI-CORE-ITEM_GROUP.adult_patient.v1
CIMI-CORE-ITEM_GROUP.author_action.v1
CIMI-CORE-ITEM_GROUP.clinical_document.v2
CIMI-CORE-ITEM_GROUP.clinical_report_header.v
CIMI-CORE-ITEM_GROUP.clinical_statement.v1
CIMI-CORE-ITEM_GROUP.cluster.v1
CIMI-CORE-ITEM_GROUP.compound_clinical_stat
CIMI-CORE-ITEM_GROUP.indivisible_clinical_state
CIMI-CORE-ITEM_GROUP.issue_action.v1
Terms
AML
CIMI Reference Model [CIMI RM v3.0.1.mdzip]
EA_PrimitiveTypes_Package [CIMI RM v3.0.1.mdzip]
Model
UML Standard Profile [UML_Standard_Profile.mdzip]
ArchetypeMetadataProfile [ArchetypeMetadataProfile
Tools
Common
Note
Text Box
Anchor
Containment
Abstraction
Dependency
Image Shape
Diagram Ove...
Diagram Leg...
Horizontal...
Class Diagram
Class
package CIMI-CORE-ITEM_GROUP.adult_patient.v1 [CIMI-CORE-ITEM
«Archetype»
CIMI-CORE-ITEM_GROUP.adult_patient.v1
«Constrains»
ITEM_GROUP
«ArchetypeVersion»
adult_patient
{description = "Adult Patient",
id = id1,
isControlled = false,
isGenerated = false}
+subjectID
1
1
1
1
«Comp
+value : DATA_

Figure 19: The packaged archetype 'adult_patient'

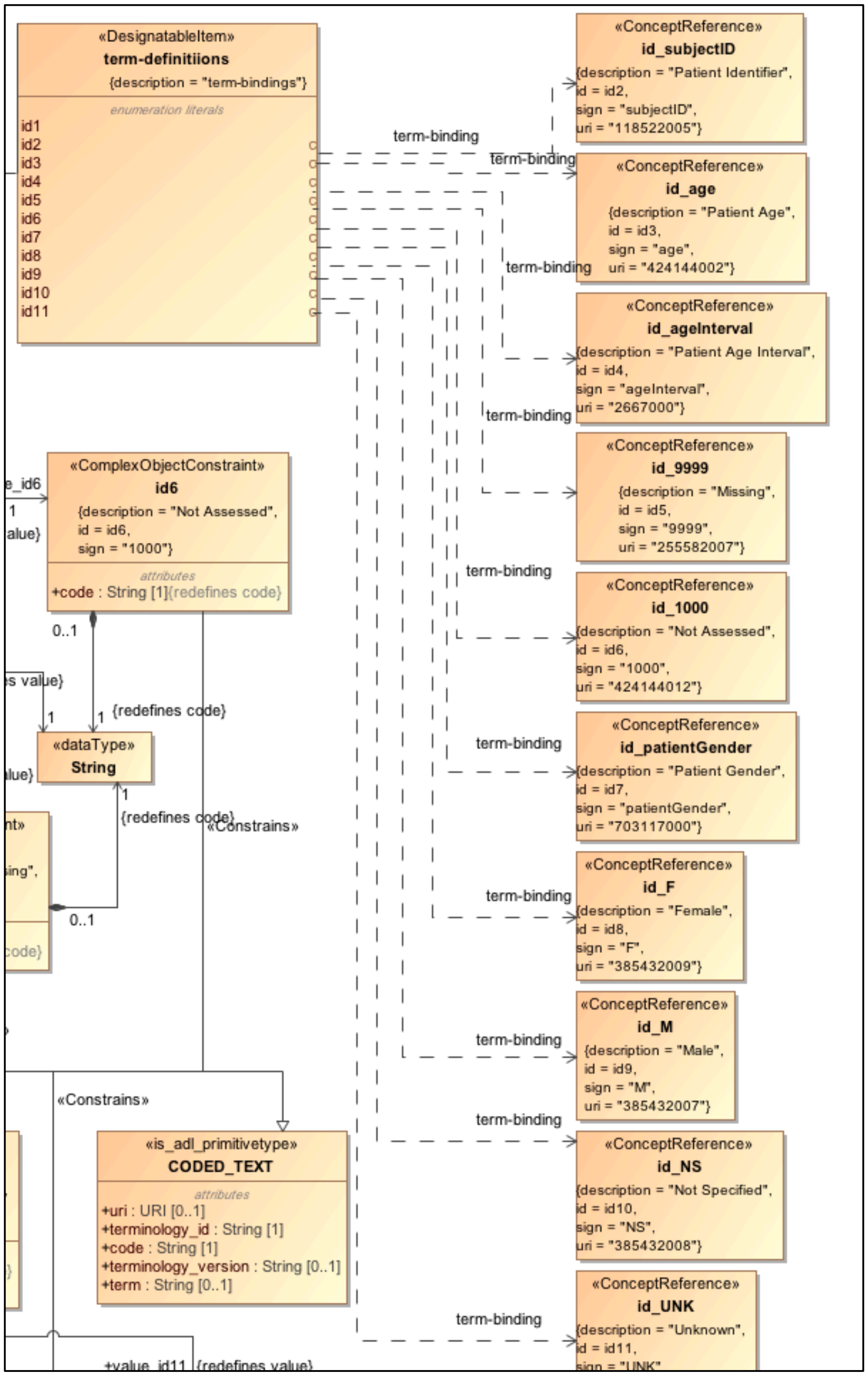


Figure 20: AML terminology binding for coded values

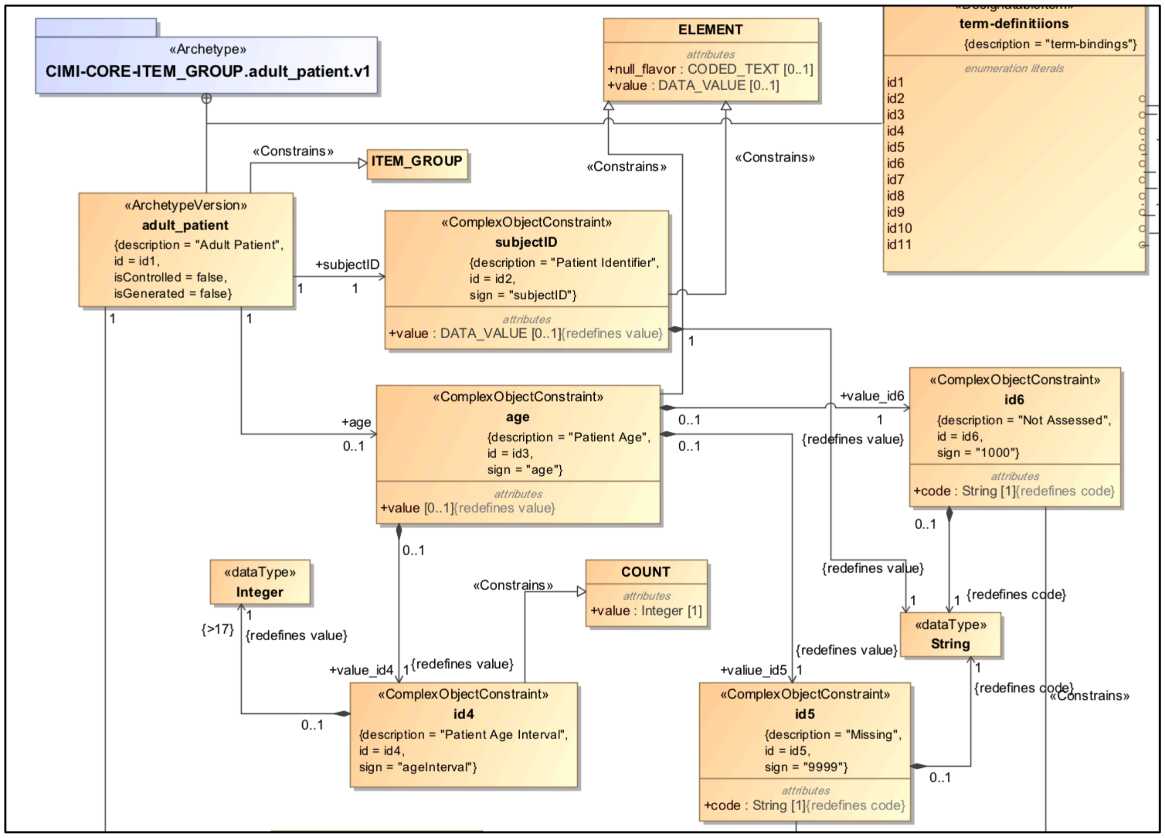

«Archetype»
CIMI-CORE-ITEM_GROUP.adult_patient.v1
ELEMENT
+null_flavor : CODED_TEXT [0..1]
+value : DATA_VALUE [0..1]
term-definitiions
{description = "term-bindings"}
id1 id2 id3 id4 id5 id6 id7 id8 id9 id10 id11
«Constrains»
ITEM_GROUP
«ArchetypeVersion»
adult_patient
{description = "Adult Patient", id = id1, isControlled = false, isGenerated = false}
+subjectID
«ComplexObjectConstraint»
subjectID
{description = "Patient Identifier", id = id2, sign = "subjectID"}
+value : DATA_VALUE [0..1]{redefines value}
«ComplexObjectConstraint»
age
{description = "Patient Age", id = id3, sign = "age"}
+value [0..1]{redefines value}
+age
«ComplexObjectConstraint»
id6
{description = "Not Assessed", id = id6, sign = "1000"}
+code : String [1]{redefines code}
+value_id6
{redefines value}
{redefines code}
«dataType»
Integer
{>17}
«dataType»
String
COUNT
+value : Integer [1]
+value_id4
«ComplexObjectConstraint»
id4
{description = "Patient Age Interval", id = id4, sign = "ageInterval"}
+value_id5
«ComplexObjectConstraint»
id5
{description = "Missing", id = id5, sign = "9999"}
+code : String [1]{redefines code}

Figure 21: The constraints of 'subjectID' and 'age' of 'adult_patient'

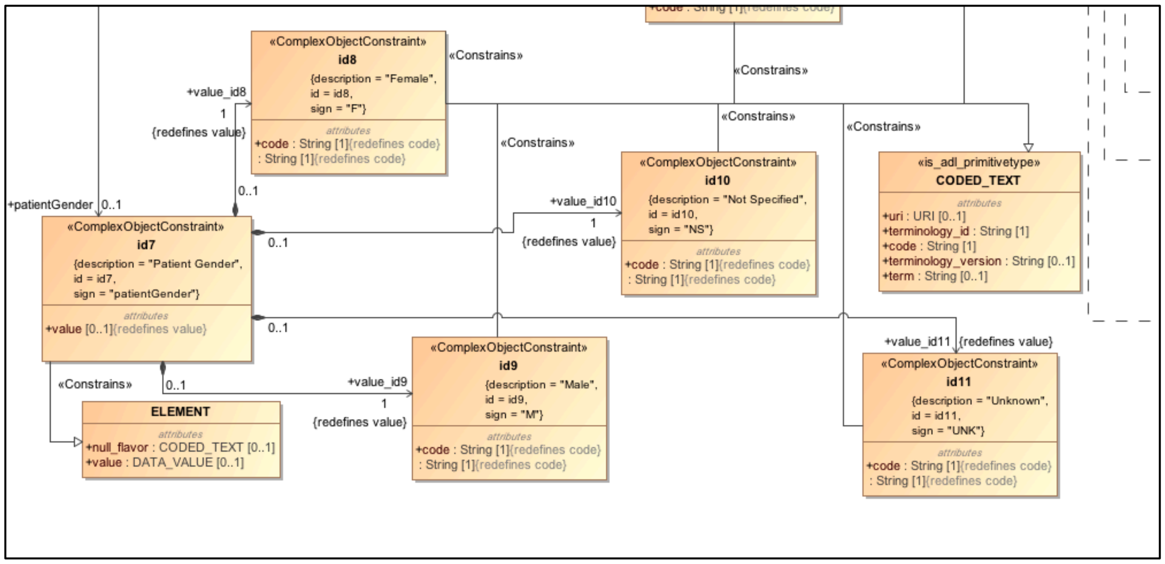

«ComplexObjectConstraint»
id8
{description = "Female", id = id8, sign = "F"}
+code : String [1]{redefines code}
+value_id8
{redefines value}
+patientGender
«ComplexObjectConstraint»
id7
{description = "Patient Gender", id = id7, sign = "patientGender"}
+value [0..1]{redefines value}
«Constrains»
«ComplexObjectConstraint»
id10
{description = "Not Specified", id = id10, sign = "NS"}
+value_id10
«is_adl_primitivetype»
CODED_TEXT
+uri : URI [0..1]
+terminology_id : String [1]
+code : String [1]
+terminology_version : String [0..1]
+term : String [0..1]
ELEMENT
+null_flavor : CODED_TEXT [0..1]
+value : DATA_VALUE [0..1]
«ComplexObjectConstraint»
id9
{description = "Male", id = id9, sign = "M"}
+value_id9
«ComplexObjectConstraint»
id11
{description = "Unknown", id = id11, sign = "UNK"}
+value_id11

Figure 22: The constraint for 'patientGender' of 'adult_patient'

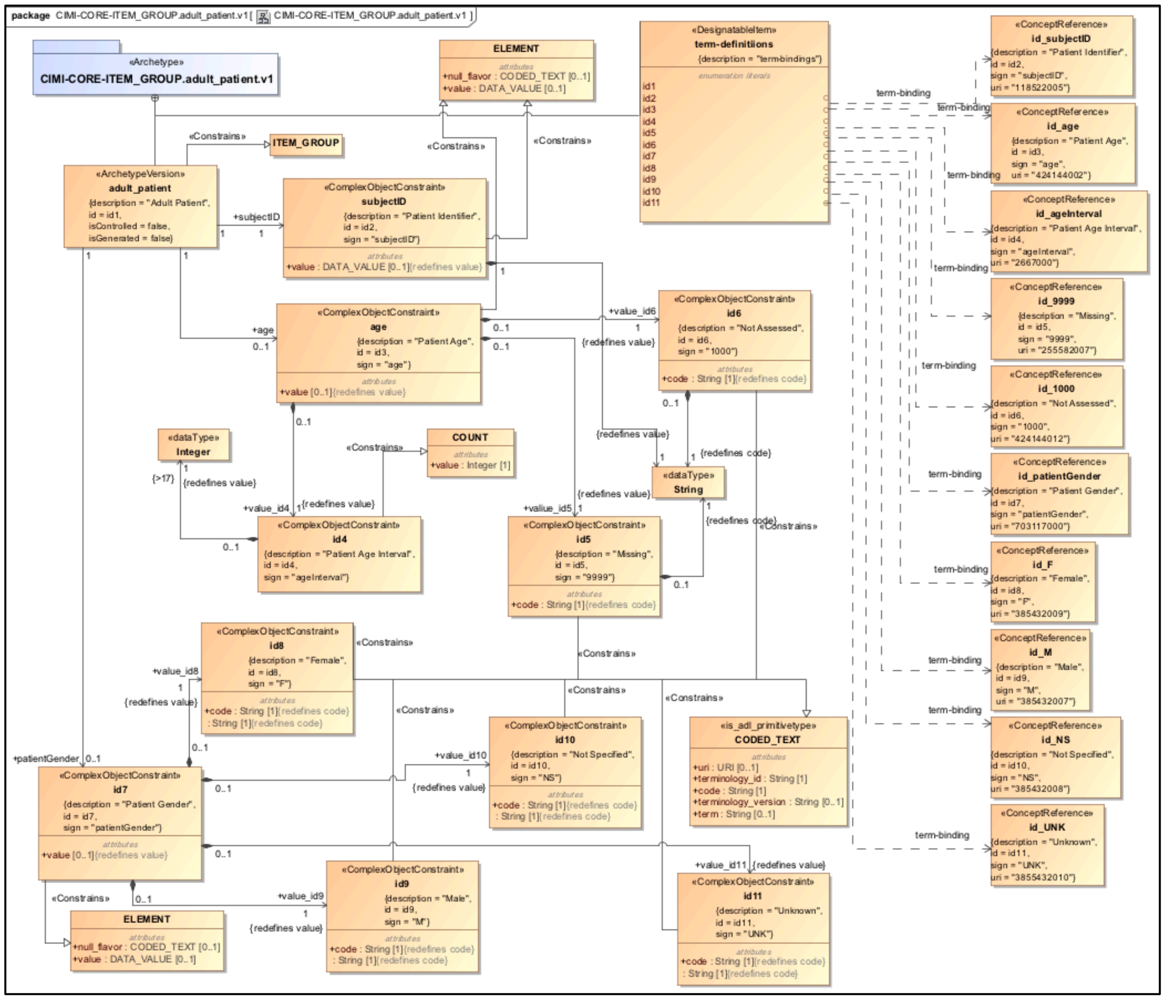


Figure 23: The AML archetype 'adult_patient' as a whole

The AML rendering of this somewhat much smaller set of constraints already results into an archetype, which is too big and complex to create manually for multiple archetypes. The promise of AML can only be realize when there are tools to make it easier for modelers to create and maintain archetypes and disseminate them for their utilization as real-world clinical models. The *AML Tooling* and its *ADL2AMLConverter* is a first step in making it easier to create the AML archetype programmatically.