



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Departamento de Física Aplicada / Applied Physics Department

**DETAILED CLINICAL MODELS AND THEIR RELATION
WITH ELECTRONIC HEALTH RECORDS**

Tesis Doctoral / Doctoral Thesis

Author: Diego Boscá Tomás

Supervisor: Montserrat Robles Viejo

Co-supervisor: José Alberto Maldonado Segura

Valencia, November 2015

Contents

I.	List of Figures	7
II.	List of tables	9
III.	Abstract	11
IV.	Resumen.....	13
V.	Resum.....	15
VI.	Acknowledgements.....	17
VII.	Introduction	23
VIII.	Hypothesis.....	24
IX.	Objectives.....	24
X.	Document organization.....	25
XI.	Abbreviations and acronyms.....	27
Chapter 1.	Background and State of the Art.....	29
1.1.	Electronic Health Records	29
1.2.	Electronic Health Record architectures.....	31
1.1.1.	Dual model architecture.....	32
1.1.2.	Archetype Definition Language	33
1.2.1.	Dual Model Standards and specifications	38
1.2.2.	Non-Dual Model Standards.....	45
1.3.	Terminologies.....	52
1.3.1.	Relevant Terminologies and ontologies.....	53
1.3.2.	Archetype terminology binding.....	56
1.4.	Data, Model, and Format transformation.....	57
1.4.1.	Data transformation.....	57
1.4.2.	Model-based transformations	60
1.4.3.	Format transformation.....	60
1.5.	Constraint definition languages	61
1.6.	Interoperability projects	62

1.7.	LinkEHR Normalization Platform	65
1.7.1.	LinkEHR Integration Engine	65
1.7.2.	LinkEHR Archetype Editor	65
1.7.3.	LinkEHR Extract Server	67
1.7.4.	LinkEHR Viewer	67
1.7.5.	LinkEHR Concept Manager	67
Chapter 2.	Migration of Health Information Systems – Model perspective.....	69
2.1.	Introduction	69
2.2.	Reference model archetypes	70
2.3.	Creation of RMA.....	71
2.3.1.	Creating RMA from meta-models	72
2.3.2.	AOM-based RMA creation	75
2.4.	Advanced archetype editing	78
2.4.1.	Plugin archetype editors	79
2.4.2.	Mapping to non-dual models archetypes	81
2.4.3.	Semantic patterns	82
2.4.4.	Archetype creation from instances	83
2.4.5.	Syntactic clinical model transformation between standards.....	85
2.5.	Improvements in LinkEHR Editor	88
2.5.1.	Connection to external repositories	89
2.5.2.	Template import.....	89
2.5.3.	Export JSON schema.....	90
2.5.4.	Export FHIR profiles.....	90
2.6.	Conclusions	90
Chapter 3.	Migration of Health Information Systems – Data perspective	93
3.1.	Introduction	93
3.2.	Data Model.....	94
3.3.	Source and target schemas	98

3.3.1.	XML Schemas	98
3.3.2.	Archetypes.....	99
3.4.	Mapping Language	100
3.5.	Customization of grouping semantics.....	106
3.5.1.	Modification of skolem functions	106
3.5.2.	Object Builders	107
3.6.	XQuery generation	109
3.6.1.	Mapping covers	109
3.6.2.	Representation of domain constraints.....	110
3.6.3.	Translation of mappings into XQuery	111
3.7.	LinkEHR mapping module	113
3.7.1.	Mapping management	114
3.7.2.	Mapping reuse	116
3.7.3.	Value correspondence editing	116
3.7.4.	Object builder editing.....	118
3.7.5.	Generation of XQuery and Testing.....	118
3.8.	Validation	120
3.8.1.	Technical Evaluation.....	121
3.8.2.	Evaluation in real settings	121
3.9.	Conclusions	124
Chapter 4.	Archetypes for generation, validation, and use in EHR Systems	128
4.1.	Introduction	129
4.2.	Background and related work.....	132
4.3.	Material and Methods.....	134
4.3.1.	LinkEHR platform.....	134
4.3.2.	Generation of implementation guides from archetypes	134
4.3.3.	Generation of NRL rules	135
4.3.4.	XML instances Generation	138

4.3.5.	Schematron Generation	139
4.3.6.	Generation of Additional Reference Materials	140
4.4.	Results	140
4.5.	Discussion.....	143
4.6.	Conclusion.....	145
Chapter 5.	Final Conclusions and Future Work.....	148
5.1.	Final conclusions	149
5.2.	Future work.....	150
5.3.	Journal and congress contributions	151
5.3.1.	<i>Journal contributions and book chapters</i>	152
5.3.2.	<i>Congress contributions</i>	152

I. List of Figures

Figure 1 Dual model meta-architecture	33
Figure 2 dADL syntax example	34
Figure 3 cADL syntax example.....	35
Figure 4 cADL example showing occurrence, existence, cardinality, and primitive value constraints.....	37
Figure 5 cADL example showing internal references and archetype slots	38
Figure 6 Excerpt from ISO 13606 Medication administration archetype	41
Figure 7 openEHR Information Model Overview. © openEHR	43
Figure 8 Excerpt from openEHR triage archetype.....	44
Figure 9 FHIR DSTU2 Medication resource in XML	49
Figure 10 Example intravenous medication instance based on medication resource	50
Figure 11 SNOMED CT example concept	54
Figure 12 SNOMED CT relationships example	55
Figure 13 Example of TGD	58
Figure 14 Example of full TGD.....	58
Figure 15 example of relationships and resources for problem clinical model.....	68
Figure 16 LinkEHR archetype edition	71
Figure 17 Steps for the creation of FHIR archetypes	72
Figure 18 Excerpt of a Systolic FHIR archetype	74
Figure 19 Archetype editing without reference model – creating attributes.....	76
Figure 20 Archetype editing without reference model – creating objects	76
Figure 21 Example of a MML BasicClinicModule Reference Model Archetype.....	78
Figure 22 Editing a FHIR Adverse Reaction archetype	79
Figure 23 editing an HL7 CDA archetype with specific editor	80
Figure 24 Reference Model Manager inside LinkEHR Editor	81
Figure 25 Including a semantic pattern in current archetype.....	83
Figure 26 ADL excerpt	97
Figure 27 Example of tree view of an XML Schema	99
Figure 28 Blood pressure archetype and an excerpt of the comprehensive archetype side by side	100
Figure 29 Example of value correspondence transforming the gender codes from a XML source.	101
Figure 30 An example of value correspondence.....	103

Figure 31 Grouping mismatch between source and target schemas	104
Figure 32 Nested relations examples, (a) and (c) are in PNF and (b) is not	105
Figure 33 Annotated archetype with skolem parameters	106
Figure 34 Parameter of the skolem function to control the creation of values in set types	107
Figure 35 Simple mapping scenario	108
Figure 36 Simple mapping scenario with object builders	109
Figure 37 XQuery template for building archetype instances from source data	112
Figure 38 Example of generated XQuery	113
Figure 39 Mapping and XQuery generation	114
Figure 40 Editing a mapping function table	117
Figure 41 Expression mapping editor.....	117
Figure 42 Object builder editing form.....	118
Figure 43 Export integration archetype dialog showing a set of covers for an archetype	119
Figure 44 HL7 CDA transformation program test output	120
Figure 45 Platform for the medication reconciliation project	122
Figure 46 Graphical representation of archetype mapping dependencies	123
Figure 47 – Proposed architecture for the generation of implementation guides from archetypes.....	132
Figure 48 Blood Pressure EN ISO 13606 sample archetype	136
Figure 49 Declaration of variables in NRL to allow the generation of readable rules	136
Figure 50 Sample rule for occurrences constraint using the readable variables.....	137
Figure 51 Sample rules for checking cardinality and existence of an attribute	137
Figure 52 Sample rules for checking different kinds of data value constraints	138
Figure 53 Options for XML instance generation in LinkEHR	139
Figure 54 Schematron rule for blood pressure measurement occurrences	140
Figure 55 Excerpt of an automatically generated implementation guide from an Apgar score openEHR archetype.....	142
Figure 56 Cells Karyotyped Count from the original Genetic Testing Report Implementation Guide	142
Figure 57 Executable rules from Cells Karyotyped Count automatically generated from the HL7 CDA Genetic Testing Report archetype.....	143

II. List of tables

Table 1 Archetypes from MML common format	77
Table 2 Archetypes from MML module concepts	77
Table 3 SNOMED CT grammar expression for categorization of openEHR classes	88
Table 4 Representation of archetype constraints in the proposed data model	97
Table 5 Functions supported by LinkEHR mapping	102
Table 6 Examples of how constraints are processed	110
Table 7 attribute mapping for the presence/absence of a metastatic tumor in a level 1 archetype	124
Table 8 attribute mapping for the presence/absence of a metastatic solid tumor from two different archetypes.....	124

III. Abstract

Healthcare domain produces and consumes big quantities of people's health data. Although data exchange is the norm rather than the exception, being able to access to all patient data is still far from achieved. Current developments such as personal health records will introduce even more data and complexity to the Electronic Health Records (EHR). Achieving semantic interoperability is one of the biggest challenges to overcome in order to benefit from all the information contained in the distributed EHR. This requires that the semantics of the information can be understood by all involved parties. It has been established that three layers are needed to achieve semantic interoperability: Reference models, clinical models (archetypes), and clinical terminologies.

As seen in the literature, information models (reference models and clinical models) are lacking methodologies and tools to improve EHR systems and to develop new systems that can be semantically interoperable. **The purpose of this thesis is to provide methodologies and tools for advancing the use of archetypes** in three different scenarios:

- Archetype definition over specifications with no dual model architecture native support. Any EHR architecture that directly or indirectly has the notion of detailed clinical models (such as HL7 CDA templates) can be potentially used as a reference model for archetype definition. This allows transforming single-model architectures (which contain only a reference model) into dual-model architectures (reference model with archetypes). A set of methodologies and tools has been developed to support the definition of archetypes from multiple reference models.
- Data transformation. A complete methodology and tools are proposed to deal with the transformation of legacy data into XML documents compliant with the archetype and the underlying reference model. If the reference model is a standard then the transformation is a standardization process. The methodologies and tools allow both the transformation of legacy data and the transformation of data between different EHR standards.
- Automatic generation of implementation guides and reference materials from archetypes. A methodology for the automatic generation of a set of reference materials is provided. These materials are useful for the development and use of EHR systems. These reference materials include data validators, example instances, implementation guides, human-readable formal rules, sample forms, mindmaps, etc. These reference materials can be combined and organized in different ways to adapt

to different types of users (clinical or information technology staff). This way, users can include the detailed clinical model in their organization workflow and cooperate in the model definition.

These methodologies and tools put clinical models as a key part of the system. The set of presented methodologies and tools ease the achievement of semantic interoperability by providing means for the semantic description, normalization, and validation of existing and new systems.

IV. Resumen

El sector sanitario produce y consume una gran cantidad de datos sobre la salud de las personas. La necesidad de intercambiar esta información es una norma más que una excepción, aunque este objetivo está lejos de ser alcanzado. Actualmente estamos viviendo avances como la medicina personalizada que incrementarán aún más el tamaño y complejidad de la Historia Clínica Electrónica (HCE). La consecución de altos grados de interoperabilidad semántica es uno de los principales retos para aprovechar al máximo toda la información contenida en las HCEs. Esto a su vez requiere una representación fiel de la información de tal forma que asegure la consistencia de su significado entre todos los agentes involucrados. Actualmente está reconocido que para la representación del significado clínico necesitamos tres tipos de artefactos: modelos de referencia, modelos clínicos (arquetipos) y terminologías.

En el caso concreto de los modelos de información (modelos de referencia y modelos clínicos) se observa en la literatura una falta de metodologías y herramientas que faciliten su uso tanto para la mejora de sistemas de HCE ya existentes como en el desarrollo de nuevos sistemas con altos niveles de interoperabilidad semántica. **Esta tesis tiene como propósito proporcionar metodologías y herramientas para el uso avanzado de arquetipos** en tres escenarios diferentes:

- Definición de arquetipos sobre especificaciones sin soporte nativo al modelo dual. Cualquier arquitectura de HCE que posea directa o indirectamente la noción de modelos clínicos detallados (por ejemplo, las plantillas en HL7 CDA) puede ser potencialmente usada como modelo de referencia para la definición de arquetipos. Con esto se consigue transformar arquitecturas de HCE de modelo único (solo con modelo de referencia) en arquitecturas de doble modelo (modelo de referencia + arquetipos). Se han desarrollado metodologías y herramientas que faciliten a los editores de arquetipos el soporte a múltiples modelos de referencia.
- Transformación de datos. Se propone una metodología y herramientas para la transformación de datos ya existentes a documentos XML conformes con los arquetipos y el modelo de referencia subyacente. Si el modelo de referencia es un estándar entonces la transformación será un proceso de estandarización de datos. La metodología y herramientas permiten tanto la transformación de datos no estandarizados como la transformación de datos entre diferentes estándares.
- Generación automática de guías de implementación y artefactos procesables a partir de arquetipos. Se aporta una metodología para la generación automática de un

conjunto de materiales de referencia de utilidad en el desarrollo y uso de sistemas de HCE, concretamente validadores de datos, instancias de ejemplo, guías de implementación , reglas formales legibles por humanos, formularios de ejemplo, mindmaps, etc. Estos materiales pueden ser combinados y organizados de diferentes modos para facilitar que los diferentes tipos de usuarios (clínicos, técnicos) puedan incluir los modelos clínicos detallados en el flujo de trabajo de su sistema y colaborar en su definición.

Estas metodologías y herramientas ponen los modelos clínicos como una parte clave en el sistema. El conjunto de las metodologías y herramientas presentadas facilitan la consecución de la interoperabilidad semántica al proveer medios para la descripción semántica, normalización y validación tanto de sistemas nuevos como ya existentes.

V. Resum

El sector sanitari produeix i consumeix una gran quantitat de dades sobre la salut de les persones. La necessitat d'intercanviar aquesta informació és una norma més que una excepció, encara que aquest objectiu està lluny de ser aconseguit. Actualment estem vivint avanços com la medicina personalitzada que incrementaran encara més la grandària i complexitat de la Història Clínica Electrònica (HCE). La consecució d'alts graus d'interoperabilitat semàntica és un dels principals reptes per a aprofitar al màxim tota la informació continguda en les HCEs. Açò, per la seua banda, requereix una representació fidel de la informació de tal forma que assegure la consistència del seu significat entre tots els agents involucrats. Actualment està reconegut que per a la representació del significat clínic necessitem tres tipus d'artefactes: models de referència, models clínics (arquetips) i terminologies.

En el cas concret dels models d'informació (models de referència i models clínics) s'observa en la literatura una mancança de metodologies i eines que en faciliten l'ús tant per a la millora de sistemes de HCE ja existents com per al desenvolupament de nous sistemes amb alts nivells d'interoperabilitat semàntica. Aquesta tesi té com a propòsit proporcionar metodologies i eines per a l'ús avançat d'arquetips en tres escenaris diferents:

- Definició d'arquetips sobre especificacions sense suport natiu al model dual. Qualsevol arquitectura de HCE que posseïska directa o indirectament la noció de models clínics detallats (per exemple, les plantilles en HL7 CDA) pot ser potencialment usada com a model de referència per a la definició d'arquetips. Amb açò s'aconsegueix transformar arquitectures de HCE de model únic (solament amb model de referència) en arquitectures de doble model (model de referència + arquetips). S'han desenvolupat metodologies i eines que faciliten als editors d'arquetips el suport a múltiples models de referència.
- Transformació de dades. Es proposa una metodologia i eines per a la transformació de dades ja existents a documents XML conformes amb els arquetips i el model de referència subjacent. Si el model de referència és un estàndard llavors la transformació serà un procés d'estandardització de dades. La metodologia i eines permeten tant la transformació de dades no estandarditzades com la transformació de dades entre diferents estàndards.
- Generació automàtica de guies d'implementació i artefactes processables a partir d'arquetips. S'hi inclou una metodologia per a la generació automàtica d'un conjunt de materials de referència d'utilitat en el desenvolupament i ús de sistemes de HCE,

concretament validadors de dades, instàncies d'exemple, guies d'implementació, regles formals llegibles per humans, formularis d'exemple, mapes mentals, etc. Aquests materials poden ser combinats i organitzats de diferents maneres per a facilitar que els diferents tipus d'usuaris (clínic, tècnic) puguin incloure els models clínics detallats en el flux de treball del seu sistema i col·laborar en la seua definició.

Aquestes metodologies i eines posen els models clínics com una part clau del sistema. El conjunt de les metodologies i eines presentades faciliten la consecució de la interoperabilitat semàntica en proveir mitjans per a la seua descripció semàntica, normalització i validació tant de sistemes nous com ja existents.

VI. Acknowledgements

Some people find this part of the thesis the hardest to write (and probably it is). Maybe the hardest part is not leaving anyone out. So in case I do, do not worry, you know how grateful I am of having you in my life.

Thanks to my director, Montserrat Robles, for her support, frankness, sound advice, and caring during all these years. Thank you for your support for all these years even in the bad times.

Thanks to my co-director, Jose Alberto Maldonado, for his suggestions, corrections, dedication, and help in the formalization of this thesis work. With his enthusiasm and inspiration, he helped in making the thesis subject interesting. You have provided me with lots of good ideas and helped me a lot during the writing of this thesis.

Thanks to David Moner, for our discussions in the lab until late evenings. These discussions have been key for the elaboration of this thesis. Thank you for always listening even when I do not make a lot of sense.

To all the people, current or former, from the IBIME group I have met over these 10 years. Thanks to Vicente, Juanmi, José Vicente, Carlos, Elías, Salva, Adrian, José Enrique, Miguel, Javier, Alejandro, Estibaliz, and Alfonso, and all the people that stayed temporary with us. Thanks for creating a healthy work environment where ideas and discussions can flourish. You are my second family. Thanks to everyone in the ITACA institute for their support.

I would like to specifically thank those that even from the distance have not forgotten about us after all these years, such as Luis Marco. He brings interesting topics and insights every time we speak and has helped me to moderate some of my strong opinions.

Thanks to all the hard working people in our spin-off, VeraTech. Thanks for keeping alive the spirit of innovation and letting the world know about the great ideas coming from this part of the world. Thanks to my fellow sufferer Santi as is always willing to lend a helping hand. Thanks to my buddy Crispin for being a friend and always being there.

Thanks to all the marvelous people from the Kyoto University. Thanks to Hiroyuki Yoshihara, Tomohiro Kuroda, Shinji Kobayashi, Naoto Kume, Tatsuya Tokunaga, and all the staff in the laboratory for their warm welcome and all the help they provided me. They gave me the opportunity of experiencing one of the best experiences in my live and I will always be grateful. As I said the day I left Japan:

みなさんと一緒に仕事をさせていただき、ありがとうございました。

また、このような素晴らしいみなさんとすごした日々はわすれません

(Thank you all for giving me the opportunity to work with you. I will never forget the great people I met and the marvelous days I spent there)

Thanks to the 'openEHRers', the people from the EN13606 association, the 'HL7ers', and all the great people I have met during all these years in the different research projects, scientific meetings and conferences. The conversations and discussions with you have led to some of the ideas written in this thesis.

I would like to thank specifically to the other research groups in the field located in Spain. Thanks to the people from Murcia, Castellon, Madrid, Seville, the Spanish Ministry of Health, and everyone in this research field. You all are part of the vanguard of this field and encourage us to improve and move forward.

To all my friends: the ones I grew with, the ones I studied with, the ones I played basketball and rugby with, the ones I went to the university with, the ones that I went to Japan with, the ones I share discussions with in the mailing lists, and the ones that have emigrated to other countries. To all of you, thank you. You made me grow as a person and probably I would not be writing this today without you.

I wish to thank my entire family. My brother, my sisters in-law, and my parents-in-law were specially supportive.

I cannot forget about my parents, M^aCarmen and Rafael. They raised me, supported me, taught me, and loved me. I am the kind of person I am thanks to them.

Finally, Thanks to my partner, M^aJesus, who I recently convinced somehow to be my wife. Thank you for your understanding, patience, and unconditional love, even during the time I was studying abroad or writing this thesis.

Last but not least, thanks to you, reader. I am pleased that you find this thesis worth your time. It is hard to overstate my satisfaction.

To all of you, I dedicate this thesis.

This thesis work was possible thanks to the predoctoral researcher formation grant from the Universitat Politècnica de València (FPI UPV 2007-24), the Ministry of Education and Science

project TSI2007-66575-C02-01 and Ministry of Economy and Competitiveness project TIN2010-21388-C02-01.

*A great idea is something that does not solve just one single problem,
but rather can solve multiple problems at once.*

Shigeru Miyamoto

VII. Introduction

In a world each time more interconnected, healthcare domain still lags behind, mostly due to the complexity, variability, and the always evolving knowledge of the clinical domain. Accessing the full Electronic Health Records (EHR) of the patients is still difficult, as the information tends to be distributed among different systems. This leads to a situation where the existence of information islands prevents the efficient use of the data stored in these systems. Sharing EHR in a meaningful and secure way will improve significantly patient care, patient safety, and clinical research (1).

Semantic interoperability (2) has always been the holy grail of medical informatics. From their first usages in the early 1960s until today, traditional systems development has not been able to achieve semantic interoperability. After 2010, interoperability projects started to emphasize the need of three different layers: generic reference models, clinical models, and clinical vocabularies. Despite of this, governments and providers still try to achieve high levels of semantic interoperability by removing diversity (e.g. by explicitly defining the messages that must be used). They tend to avoid variability, which at the end only allows interoperability on a limited level, as assumes that everyone will have exactly the same information needs and everything outside it is just ignored. Variability is almost intrinsic of clinic domain. New tests and data needs are continuously being introduced and changed due to improvements in the clinical practice. Systems should support this evolution and not discourage it.

One of the prerequisites to achieve semantic interoperability is the standardization of both the data and concepts present in information systems. This is even more important in healthcare domain where data needs to be exchanged in a way that its precise meaning is preserved. To achieve this goal, the syntax, structure, and semantics of health data need to be standardized. The new generation of EHR architectures (EHRA) use Detailed Clinical Models (DCM) as the discrete set of precise documentable clinical knowledge to specify the structure and constraints to follow. A set of these EHRA are based on what is called dual model methodology (3). It tries to overcome the problems caused by the complexity and continuous evolution of health domain. Dual model methodology distinguishes two models, the Reference Model that contains the basic and stable entities for representing any entry in an EHR, and the DCM (expressed as archetypes), which formally define the domain and application-specific domain models such as blood pressure, discharge report, or lab result. Despite using dual model systems provides advantages, the migration or integration of current systems is not always

easy. Existing clinical data must be transformed to meet the data structures defined by reference model and archetypes. We face a problem known in literature as the data exchange (translation or transformation) problem (4). Data exchange at schema level requires an explicit representation of how the source schema (legacy data schema) and target schema (archetypes) are related to each other. These explicit representations are called mappings (5,6). The application of data exchange methods to map existing legacy EHR systems and archetypes allow the enrichment of legacy data and their meaningful communication. Apart from mappings, the application of dual modeling to legacy EHR systems allows their improvement by means of all the available archetype-based methodologies and tools. Finally, the use of archetypes as basis for the automatic generation of reference materials such as implementation guides or Schematron rules provides further benefits, for instance, in the development of new EHR systems or the validation of legacy EHR systems.

VIII. Hypothesis

Detailed clinical models can be applied to describe the structure, content, and meaning of existing EHR systems as well as to facilitate the development and deployment of new EHR systems that require semantic interoperability.

IX. Objectives

The overall objective of this thesis is to provide a set of methodologies and tools based on archetypes for the achievement of higher levels of EHR semantic interoperability.

The concrete objectives of the research are:

1. To provide means of applying dual model methodology to non-dual model standards. This includes providing support for the definition of archetypes based on any EHR information model, either a standard or local model.
2. To provide tools for the transformation of existing data into data instances compliant with archetypes and underlying reference models. Due to the potential complexity of reference models and the evolving nature of archetypes, the transformation shall be guided by high-level non-procedural mappings that describe the relationship between archetypes and legacy clinical data. These high-level mapping must be then automatically compiled into executable scripts.
3. To enable the automatic generation of sets of reference materials from clinical archetypes, regardless of the reference model on which archetypes are based. This

includes the combination of the reference materials into views suitable for each type of final user.

X. Document organization

This thesis is divided into five chapters.

First chapter contains the background and state of the art, which offers a vision on EHR and EHR architectures, terminologies, data transformation, model-based transformations, format transformations, constraint definition languages, interoperability projects, and LinkEHR normalization platform.

Second chapter describes the methodologies and tools for the representation of non-dual model based reference models with archetypes.

Third chapter presents the created methodologies and tools for the transformation of both legacy EHR data and standard-based data into archetype-based data.

Fourth chapter describes and discusses the use of archetypes for the generation of reference materials, in the upgrade of existing clinical information systems as well as the development of new ones.

Fifth chapter presents the final conclusions, future work, related publications, and congress papers.

XI. Abbreviations and acronyms

ACE	Attempto Controlled English
ADL	Archetype Definition Language
AML	Archetype Modelling Language
ANSI	American National Standards Institute
AOM	Archetype Object Model
AQL	Archetype Query Language
AM	Archetype Model
API	Application Programming Interface
C-CDA	Consolidated Clinical Document Architecture
CCR	Continuity of Care Record
CDA	Clinical Document Architecture
CDISC	Clinical Data Interchange Standards Consortium
CEN	European Committee for Standardization
CEM	Clinical Element Model
CEML	Clinical Element Modelling Language
CM	Clinical Model. Detailed, reusable and domain-specific definition of a clinical concept.
CIM	Computation Independent Model
CKM	Clinical Knowledge Manager
CSS	Cascading Style Sheets
D-MIM	Domain Message Information Model
DCM	Detailed Clinical Models
DICOM	Digital Imaging and Communication in Medicine
DTD	Document Type Definition. A description of the structure of an XML document.
DSTU	Draft Standard for Trial Use
EHR	Electronic Health Record
EHRA	Electronic Health Record Architectures
EMF	Eclipse Modeling Framework
EMR	Electronic Medical Record
epSOS	euROpean patient Smart Open Services
FHIR	Fast Healthcare Interoperability Resources
FLWOR	XQuery expression acronym of FOR-LET-WHERE-ORDER BY-RETURN, analogous to the SQL SELECT-FROM-WHERE
GEHR	Good European Health Record
GTR	Genetic Testing Report
GUI	Graphic User Interface
HCDSNS	Historia Clínica Digital del Sistema Nacional de Salud
HIT	Health Information Technology
HL7	Health Level Seven
HTML	HyperText Markup Language
ICD	International Classification of Diseases
IHTSDO	International Health Terminology Standards Development Organization
IM	Information Model, a conceptual model of the information needed to support a business function or process
ISO	International Organization for Standardization
LOINC	Logical Observation Identifiers Names and Codes
MDA	Model-Driven Architecture

MDD	Model Driven Development
MDE	Model Driven Engineering
MDHT	Model-Driven Health Tools
MML	Medical Markup Language
NEHTA	National E-Health Transition Authority
NHS	National Health Service
NPO	Non-Profit Organization
NRL	Natural Rule Language
OCL	Object Constraint Language
ODM	Operational Data Model
OET	Operational Template. An internal format from Ocean Informatics Template Designer
OHT	Open Health Tools
OMG	Object Management Group
ONC	US Office of the National Coordinator
OPT	Operational Template. An artefact used in ADL to represent local archetype specializations.
OWL	Web Ontology Language
PIM	Platform Independent Model
PDF	Portable Document Format
POJO	Plain Old Java Objects. Simple Java classes that need no framework to be used
REST	Representational State Transfer.
R-MIM	Refined Message Information Model
RM	Reference Model
SNOMED-CT	Systematized Nomenclature of Medicine – Clinical Terminology
SWRL	Semantic Web Rule Language
UCUM	Unified Code for Units of Measure
UI	User Interface
UML	Unified Modeling Language
UMLS	Unified Medical Language System
URI	Uniform Resource Identifier
VHA	Veterans Health Administration
VMR	Virtual Medical Record
WSDL	Web Services Description Language
W3C	World Wide Consortium
XMI	XML Metadata Interchange
XML	Extensible Markup Language
WSDL	Web Services Description Language
XSD	XML Schema Definition

Chapter 1.

Background and State of the Art

1.1. Electronic Health Records

Health care is increasingly producing and consuming large amounts of information. Most of this information is the health record of individuals, in digital form, which is referred to as the Electronic Health Record (EHR). Although being a cornerstone concept in medical informatics there is not a common definition of EHR. An ISO report (7) states that “Previous attempts to develop a definition for the Electronic Health Record have foundered due to the difficulty of encapsulating all of the many and varied facet of the EHR in a single comprehensive definition”. Nevertheless, this report provides a definition that attempts to consolidate the various definitions:

“A repository of information regarding the health of a subject of care in computer processable form, stored and transmitted securely, and accessible by multiple authorized users. It has a standardized information model, which is independent of EHR systems. Its primary purpose is the support of continuing efficient and quality-integrated healthcare and it contains information, which is retrospective, concurrent, and prospective”

From the previous definition it is clear that EHR is not owned by any single information system and contains complete records of encounters of a patient throughout the visited healthcare organizations. The distribution of EHR content makes that sharing information is the norm rather than the exception, although this desideratum is far from being sufficiently addressed. Furthermore, the predictable shift towards personalized medicine will cause drastic increase in size and complexity of EHR systems, which will again affect clinical data integration. Achieving

a high level of **semantic interoperability** is one of the most important challenges for meaningful use of EHR. Semantic interoperability is the ability, facilitated by ICT applications and systems, to exchange, understand and act on Health-related information and knowledge among linguistically and culturally disparate health professionals, patients and organizations (2). It is vital to assure global consistency in meaning, a basic requirement to enable better health care as well as secondary use of EHR data for research (8–10). From the health care delivery perspective, interoperable EHR systems ideally enable the automation of processes across different healthcare organizations, save time and resources while increasing patient safety. From a research perspective, interoperable EHRs provide a computable collection of fine-grained longitudinal phenotypic profiles, facilitating cohort-wide investigations and knowledge discovery on an unprecedented scale (11).

The intrinsic complexity and variability of health data makes standardization crucial to achieve a high level of semantic interoperability. Currently there is a mature body of EHR standards covering the three layers of artefacts to represent the meaning of health data (2):

- **Generic Reference Models** for EHR communication. They contain a basic and stable representational framework for describing all EHR entries, the way how they are aggregated, and the context information required to meet ethical, legal and provenance requirements. The last generation of such models is the result of international research over the past decades, e.g. ISO/EN13606 (12), HL7 CDA Release 2 (13), the openEHR Reference Model (14) or HL7 FHIR (15).
- **Clinical models** are detailed, reusable and domain-specific definition of a clinical concept (such as Apgar score, discharge report, and primary care EHR). Examples of such models are openEHR/ISO13606 archetypes (16), CDA templates, detailed clinical models (17) and Clinical Element Model (CEM) (18). Currently, the International Clinical Information Modelling Initiative (CIMI) (19) is working on providing a common format for the definition of health information content based on the openEHR/ISO13606 archetype model. Concretely, archetypes are standardized and reusable models for capturing and representing clinical content. They are formed by the constrained combination of the reference model entities. Archetypes may logically include other archetypes, and can be specialized. They provide a powerful way of managing the description, creation and validation of the EHRs.
- **Clinical vocabularies** such as terminologies, ontologies and classification systems. Increasingly clinical vocabularies, particularly the clinical terminology SNOMED CT (20)

and the upcoming ICD-11 (21) classification, are based on ontologies. This means that they do not only aggregate the common meaning of domain terms as concepts, but also provide precise description about the things these terms denote in the clinical domain. One important challenge to be met is to find an effective way to use them within the EHR (22). The semantic description of information models (expressed as archetypes) is achieved by linking data structures and content to terminologies and ontologies. The crucial difference is that information models describe information in the EHR whereas ontologies describe (classes of) objects and processes in the world (23).

The deployment of interoperable information systems in the healthcare sector is presently much less than in other public service sectors or industries. This is mainly due to the complexity and variability of health data. There already exist a plethora of proprietary and standardized data and metadata definitions, organized vocabularies in the form of classifications and terminologies, communication standards and profiles. But their rapid change and increasing complexity makes them barely affordable to be implemented even for big companies. It must be noted that eHealth interoperability standards are usually defined as documented specifications that must be brought to life by system designers and implementers. This initial implementation effort requires deep expertise in the profiles and standards, often not accessible for many organizations (companies, hospitals, health authorities, etc.). But this challenge is even more acute given the significant efforts needed to test and meet conformance criteria. When organizations fail to perform these conformance and interoperability tests, they are faced with a large variability among the implementations, and have to resolve incompatibilities project by project in an ad-hoc and reactive manner. Those induced costs not only discourage, but they also slowdown the adoption of profiles and standards, thus generating uncertainty about standards.

1.2. Electronic Health Record architectures

It is widely acknowledged that standardization of data and concepts is a prerequisite to achieve semantic interoperability in any domain. This is even more important in the healthcare sector where the need to exchange data is not an exception but the rule. The faithful communication of EHRs crucially depends on the standardization of its syntax, structure and semantics, i.e. on the standardization of the EHR architecture (EHRA) and vocabulary used to communicate data. Currently there are several international organizations working on the

standardization of EHRA. Health Level 7 (24) is an international standardization organization that has defined standards for communicating data between different systems through messaging (HL7v2.x and HL7 v3 messages) and also a model that defines the structure and semantics of medical documents (the Clinical Document Architecture, CDA). The Technical Committee (TC) 251 of the European Committee for Standardization (CEN) has also developed a European Standard, now also accepted as an ISO standard, for the communication of the electronic health record called ISO/EN13606 (12). TC 251 has also developed the Health Informatics Service Architecture (HISA) (25), a 3 part standard for specifying unified and open service architectures based upon a middleware of information services. The openEHR foundation (14) has also developed the specifications of a complete architecture designed to support the constructions of distributed, patient-centered, life-long, shared care health records. ISO13606 and openEHR share the same dual model philosophy. Finally, Clinical Data Interchange Standards Consortium (CDISC) is an open organization that develops data standards that enable information system interoperability to improve medical research and related areas of healthcare (26).

1.1.1. Dual model architecture

Due to the complexity and the continuous evolution of the health domain the development of EHRA is not an easy task. A new approach for their development has been proposed known as the dual model methodology. The most remarkable feature of the dual model approach is separation of information models representing the generic and stable properties of EHR (called the reference model) from domain models such as blood pressure measurement, discharge report, prescription or microbiology result which are represented by archetypes. Only the stable reference model is hard-coded in database schemas or software, while the possible numerous and volatile domain concepts (archetypes) are modeled separately by domain specialists. Since the software is only bound to the reference model it has no direct dependency on domain concepts. Therefore, systems do not need to be changed when domain concepts are created or altered. Examples of Dual Model architectures are CEN/ISO 13606 and openEHR. Although HL7 v3 cannot be considered a true Dual Model standard; it uses a quite similar approach. These architectures will be discussed in the following sections.

In EHR environments, a reference model represents the generic and stable properties of health record information. It specifies the set of classes that form the generic building blocks of the EHR, how these building blocks should be aggregated to create more complex data structures and the context information that must accompany every piece of data in order to meet ethical,

legal and provenance requirements (27). The generality of reference models is complemented by the particularity of archetypes. Archetypes are formal definitions of a distinct domain-level concept in the form of constrained hierarchies of the building blocks defined in the reference model. Archetypes define or constrain the names and other relevant attribute values, optionality and multiplicity at any point in the hierarchy, the data types and value range that atomic attributes may take. Their formal description is achieved by linking the data structures and content to knowledge resources such as terminologies and ontologies.

Figure 1 (28) shows the implications of a dual model approach. The information created by the user is compliant both with the defined archetypes/templates and the underlying Reference Model. Archetypes constraint the Reference Model and are created by domain experts by using the Archetype Object Model (AOM). AOM can be serialized into Archetype Definition Language (ADL). Terminologies are used in archetypes to express the archetype semantics and to constrain the codes that will be used when the user creates the information.

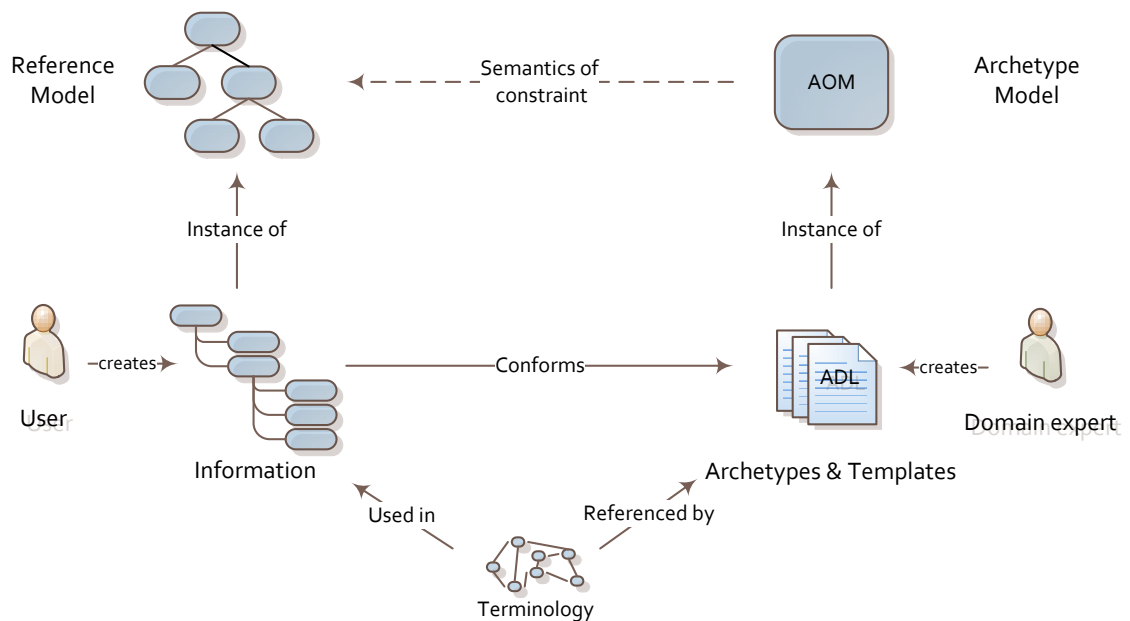


FIGURE 1 DUAL MODEL META-ARCHITECTURE

1.1.2. Archetype Definition Language

Archetype Definition Language (ADL) (29) is a model independent structured syntax for the specification of clinical information models. Both ISO13606 and openEHR use ADL for archetype specification. Current stable version of ADL is 1.4, which is the one used in this thesis. ADL defines three syntaxes: cADL for constraint definition, dADL for data definition and

a version of first-order predicate logic (FOPL) similar to OCL (30) to describe additional invariants. The cADL syntax is used to express the archetype 'definition' section while dADL syntax is used to express the 'language', 'description', and 'terminology' (formerly 'ontology'), and 'revision_history' sections. The FOPL is used in the 'rules' (formerly 'assertions') section. Archetype structure includes the following sections:

- An archetype header containing the archetype identifier
- An optional parent archetype identifier that the current archetype specializes
- A concept code that represents the real world concept this archetype represents, such as "body mass", "patient", or "blood pressure"
- The original language of the archetype
- An optional description section containing the archetype metadata
- A definition section containing the formal constraint definitions of the archetype
- An optional assertions section containing the invariants
- A mandatory section containing the definition of the terms in different languages and the terminology definitions and bindings
- An optional revision history section containing the history of changes and audit

dADL

dADL syntax (also called Object Data Instance Notation, ODIN) provides formal means of expressing instance data based on an underlying information model. dADL is intended to be readable by humans and machines. An example of dADL code can be seen in Figure 2.

```

film = (List<FILM>) <
  [0001] = <
    id = <                -- film identifier
      film_name = <"Star Wars">
      studio = <"20th Century Fox">
    >
    film_specs = <        -- film specifications
      length = <121>
      genre = <"scifi">
      year = <1977>
    >
  >
  [0002] = <
    -- Another film
  >
>

```

FIGURE 2 dADL SYNTAX EXAMPLE

dADL is intended to represent data making as few assumptions as possible about the underlying information model. Only a simple set of types are included in the syntax, namely Integer, Real, Boolean, String, and ranges of Dates and Times. Every other complex type is

derived from these. dADL is similar to XML, but one of dADL principles is being human-readable and is designed to better represent object-oriented semantics.

There are two types of identifiers in dADL: type names and attribute names. Type names are identifiers with the initial letter in uppercase followed by a combination of letters, digits, and underscores. An attribute name is an identifier with an initial lowercase letter followed by a combination of letters, digits, and underscores. As a convention, all type names are written all in uppercase, e.g. COMPOSITION. This excludes the built-in types, that are written in mixed case (String, Real, Integer, etc.). Also as a convention, attribute names are written all in lowercase. However, some reference models (e.g. HL7 CDA) do not follow this pattern, and require additional steps and transformations before being able to represent their data in dADL syntax.

dADL primitive types allow the definition of values in leaf nodes. These values include instances of primitive types (e.g. "a string", 123.4, True, 2015-10-26), lists, or intervals of primitive types. In addition to that, everything following '--' is considered a comment and is ignored during parsing.

cADL

cADL is a syntax that enables the definition of constraints over data defined by object-oriented information models expressed as archetypes or other knowledge representation formalisms. cADL constraints can be used at runtime to allow the systems to validate data following a given information model. An example of cADL code can be seen in Figure 3.

```
FILM[at0000] matches {                                -- FILM definition
  name matches {
    TEXT matches {/.+}/                               -- non-empty string
  }
  actors cardinality matches {1..*} matches {
    PERSON[at0001] matches {
      -- etc --
    }
  }
}
```

FIGURE 3 CADL SYNTAX EXAMPLE

Unlike dADL, cADL includes a set of reserved words, such as matches, occurrences, existence, cardinality, includes, use_node, etc. Of these reserved words, 'matches' is probably the most important one, as is the one used for the definition of constraints in both objects or parts of an object (attributes). When it occurs between a name and a braces-delimited block refers to the

set of valid values for that name. In data, constraints are recursively matched for either objects and attributes until leaf level constraints are matched.

Identifiers in cADL follow the same rules that in dADL: A type name is an identifier with an initial uppercase letter followed by a combination of letters, digits, or underscores. An attribute name is an identifier with an initial lowercase letter followed by a combination of letters, digits, or underscores. In the same way, everything following '--' is considered a comment and is ignored during parsing. As in dADL, some reference models do not adhere to these rules and thus must be processed for their representation in cADL.

In cADL, an entity in brackets e.g. [xxxx] is used to identify "object nodes", i.e. nodes expressing constraints over types. These identifiers allow the definition of different sets of constraints over the same type (e.g. PERSON[at0001] could define male actors and PERSON[at0002] define actresses, each one with their own set of constraints). These object nodes correspond to entities in an object-oriented information model.

cADL constraints always constraint parts from the underlying information model. This implies, on the one hand, that constraints cannot be stronger than the ones in the information model. On the other hand, cADL includes only the constraints for the parts of the information model which are useful or meaningful to constraint. E.g. it is useful to constrain the valid values for a measure, but probably is not useful to constrain the name of the patient.

Different kinds of constraints can be defined depending on the kind of node being constrained (element, attribute, or primitive type). For attribute constraints, every attribute can redefine their existence. These constraints say if an attribute must exist. There are three states on the existence (0..1 optional, 1..1 mandatory, and 0..0 not allowed; default is 1..1). Figure 4 shows an example of an existence constraint for the mandatory attribute 'identifier' or for the optional 'actors'. In addition to existence constraints, container attributes (also known as multiple attributes as opposed to the single attributes) allow to define constraints over lists and sets. These attributes use cardinality keyword to indicate both the allowed membership of the container (e.g. 0..* for 0 to many, 2..4 for from 2 to 4, etc.) and the semantics of the set (ordered, unique). Figure 4 shows an example of 'actors' cardinality and semantics. In this case, at least one actor or actress must be included and is a logical set (as it is unordered and unique).

It is also possible to define how many times in runtime an instance of an object can occur. This is stated by defining the object occurrences. Objects inside a single attribute are only allowed

to have occurrences of 0..1 or 1..1. In Figure 4 both the actor (PERSON[at0001]) and actress (PERSON[at0002]) have occurrences 0..*, i.e. any number of actors or actresses can be included under 'actors' attribute.

More than one object constraint can be defined inside an attribute, and the meaning depends on whether the attribute is a single attribute or multiple attribute. If more than one object constraint is included inside a single attribute then they are considered as alternatives, and only one of the constraints needs to be matched on the data. As shown in Figure 4, a valid instance needs to include either a TEXT object or an INTEGER object inside an 'identifier' attribute.

```

FILM[at0000] matches {
    -- FILM definition
    identifier existence {1..1} matches {
        TEXT matches
            value matches {/.+}/ -- non-empty string
        INTEGER matches
            value matches {|0..1000|} -- an integer
    }
    actors existence {0..1} cardinality matches {1..*; unordered; unique} matches {
        PERSON[at0001] occurrences matches {0..*} matches { -- actor definition
            name existence matches {1..1} matches {*}
            honorific existence matches {0..1} matches {"Mr"}
        }
        PERSON[at0002] occurrences matches {0..*} matches { -- actress definition
            name existence matches {1..1} matches {*}
            honorific existence matches {0..1} matches {"Mrs", "Ms", "Miss"}
        }
    }
}

```

FIGURE 4 CADL EXAMPLE SHOWING OCCURRENCE, EXISTENCE, CARDINALITY, AND PRIMITIVE VALUE CONSTRAINTS

cADL also allows to leave a constraint open. This is called "any", and is shown by an asterisk in braces. This shows explicitly that the property can have any value (always having the underlying information model as a basis). In Figure 4, 'name' attribute can contain any value that is allowed by the underlying model.

cADL also allows the reuse of complex structures from inside and outside of the archetype. The former is called archetype internal reference, which allows including a constraint already defined in a different block. Archetype internal references are denoted by the keyword 'use_node'. The latter is called archetype slot and allows the use of other external archetypes rather than define the constraints inline. Archetype slots are defined by a set of assertions (includes, excludes) which define the set of allowed or excluded archetypes in the slot. Archetype slots are denoted by the keyword 'allow_archetype'. Figure 5 shows both an

example of an archetype internal reference (in 'director') and an archetype slot (for the definition of PERSON[at0003], 'extra').

```

FILM[at0000] matches {
    identifier existence {1..1} matches {
        -- etc ---
    }
    actors existence {0..1} cardinality matches {1..*; unordered; unique} matches {
        PERSON[at0001] occurrences matches {0..*} matches {
            name existence matches {0..1} matches {*}
        }
        PERSON[at0002] occurrences matches {0..*} matches {
            -- actress definition
        }
        allow_archetype PERSON[at0003] occurrences {0..*} matches {
            -- include archetype for extra
            include
                archetype_id matches {/.*\.PERSON\.extra\.*/}
        }
    }
    director existence {1..1} matches {
        use_node PERSON /actors[at0001] -- reuse actor constraint
    }
}

```

FIGURE 5 CADL EXAMPLE SHOWING INTERNAL REFERENCES AND ARCHETYPE SLOTS

Finally, cADL also allows the definition of constraints over primitive types. For the definition of these constraints, type name is omitted and the constraint is put directly on the braces. Each different kind of primitive types allows a different set of valid constraints. For String constraints, they can be constrained in two ways: a list of fixed strings, or by using a regular expression. For Integer and Real constraints, either a list of values or a range can be defined. For Date, Time, DateTime, and Duration constraints specific lists of values, intervals, or patterns can be defined. Figure 4 shows several examples of primitive constraints, namely the pattern “/./” for defining non-empty strings inside a TEXT object, the range from 0 to 1000 for the INTEGER object, or the list of valid honorific for actor and actress.

1.2.1. Dual Model Standards and specifications

1.2.1.1 ISO/EN13606, International standard

ISO/CEN EN13606 (12) is a five-part standard for the communication of Electronic Health Records (EHR). It was first approved as a European norm by the TC/251 in 1999. The norm was proposed and accepted as an ISO norm in 2008 (parts 1 and 2), 2009 (parts 3 and 4) and 2010 (part 5). The five parts are:

- ISO13606 part 1: Reference model (12).defines a generic information model that defines generic data structures and their relationships to express any kind of information that can be included in the EHR.

ISO13606 part 1 most relevant classes are:

EHR_EXTRACT

The EHR_EXTRACT is the top-level container of the complete or partial EHR of a single patient.

FOLDER

Folders are higher level hierarchical organizers of the EHR. Folders logically join several compositions by different criteria. Examples of folders include pediatrics, all patient EHR, episodes from last year, GP folder, etc.

COMPOSITION

A composition is all the information committed by a single agent as a result of a single clinical encounter. Examples of this include discharge summary, referral letter, radiology report, health summary, etc.

SECTION

A section groups data within a composition in order to reflect the flow of information gathering during a clinical encounter, or to structure it in order to ease human readership. Examples of this include family history, subjective symptoms, treatment, reason for encounter, past history, etc.

ENTRY

An entry includes all the information recorded in the EHR as a result of a clinical observation, evaluation (clinical interpretation), instruction (intention), or action. Examples of entries include blood pressure measurement, diagnosis, a symptom, an observation, etc.

CLUSTER

A cluster is the way of representing columns or tables, and other nested data structures such as time series. Examples of clusters include Electro-encephalogram interpretation, heart rate response to exercise diagrams, etc.

ELEMENT

The element is the leaf node of the EHR. It contains a single value. Examples of elements include body weight, body height, allergy code, allergy name, medication dose, etc.

- ISO13606 part 2: Archetype interchange specification (31)

Part 2 includes a generic information model to define archetypes, which are re-usable models of a domain concept. It also includes the formal language for the specification of these archetypes called Archetype Definition Language (ADL). This part is based on the openEHR ADL.

- ISO13606 part 3: Reference archetypes and term lists (32)

Contains both the local terminology used by part 1 and a set of reference archetypes that represent how other standards can be expressed with this norm.

- ISO13606 part 4: Security (33)

This part includes a basic security framework for the specification and communication of security and access policies for the EHR.

- ISO13606 part 5: Exchange models (34)

Part 5 includes a minimal interface and messaging specification for the EHR communication.

ISO13606 archetypes are expressed in ADL. Figure 6 shows an excerpt from an ISO13606 medication administration archetype. The first section includes a header with the archetype identifier (CEN-EN13606-CLUSTER.medication_admin.v1). The next section to appear is the concept code (pointing to “Medication Administration”). Then the original language of the archetype (English) is stated. Next, it comes a description section containing information about the original author, the archetype lifecycle or language dependent metadata such as purpose,

use, misuse, etc. The archetype definition section contains the medication administration (CLUSTER[at0000]) with the different parts of its structure, namely the route (ELEMENT[at0001]), site (ELEMENT[at0002]), delivery method (ELEMENT[at0003]), dose duration (ELEMENT[at0004]), and infusion details (CLUSTER[at0005]) which is defined in its own archetype CEN-EN13606-CLUSTER.infusion_details. Finally the ontology section contains the definition for the terms used in the archetype.

```

archetype (adl_version=1.4)
  CEN-EN13606-CLUSTER.medication_admin.v1
concept
  [at0000]
language
  original_language = <[ISO_639-1::en]>
...
description
  original_author = <
    ["organisation"] = <"openEHR Foundation">
    ["name"] = <"Sam Heard">
  >
  lifecycle_state = <"CommitteeDraft">
  details = <
    ["en"] = <
      language = <[ISO_639-1::en]>
      purpose = <"To describe how a medication should be administered or was actually administered.">
    >
  >
...
definition
  CLUSTER[at0000] matches { -- Medication administration
    parts existence matches {1..1} cardinality matches {1..*; unordered} matches {
      ELEMENT[at0001] occurrences matches {0..1} matches { -- Route
        value existence matches {0..1} matches {
          SIMPLE_TEXT occurrences matches {0..1} matches {*} --
        }
      }
      ELEMENT[at0002] occurrences matches {0..1} matches { -- Site
        value existence matches {0..1} matches {
          SIMPLE_TEXT occurrences matches {0..1} matches {*} --
        }
      }
      ELEMENT[at0003] occurrences matches {0..1} matches { -- Delivery method
        value existence matches {0..1} matches {
          SIMPLE_TEXT occurrences matches {0..1} matches {*} --
        }
      }
      ELEMENT[at0004] occurrences matches {0..1} matches { -- Dose duration
        value existence matches {0..1} matches {
          DURATION occurrences matches {0..1} matches { --
            value existence matches {1..1} matches {>=PT0S}
          }
        }
      }
    }
  allow_archetype CLUSTER[at0005] occurrences matches {0..*} matches { -- Infusion details
    include
      archetype_id/value matches {/CEN-EN13606-CLUSTER\.infusion_details(-[a-zA-Z0-9_]+)*\.v1/}
  }
}
...
ontology
  term_definitions = <
    ["en"] = <
      items = <
        ["at0000"] = <
          text = <"Medication administration">
          description = <"Information about the future or actual administration of medication.">
        >
      >
    >
  >

```

FIGURE 6 EXCERPT FROM ISO 13606 MEDICATION ADMINISTRATION ARCHETYPE

ISO13606 has been adopted by several national and regional projects, such as Spain (35), Sweden (36), or Minas Gerais (Brazil) (37,38).

1.2.1.2 openEHR specifications

openEHR (14) is an international non-profit foundation created by University College London, UK and Ocean Informatics in 2000. They have developed a technology-independent

architecture that includes a reference model, archetypes, and templates. OpenEHR is based around the openEHR reference model, which is closely related to ISO 13606-1, and openEHR archetype model, which has been adopted by ISO 13606-2. The most relevant classes of OpenEHR reference model are:

COMPOSITION Class

In openEHR, COMPOSITION is the top level data container. It is the committal unit where all information within the EHR will be contained. Compositions can contain SECTION or ENTRY classes to contain the clinical content.

Compositions correspond to commonly used clinical documents, such as Discharge summary, Referral document, Health Summary, Pharmacy dispense, etc.

openEHR Composition class is equivalent to ISO13606 Composition Class

SECTION Class

Sections provide both a logical structure and a navigational structure for readers of the record. openEHR Section class is equivalent to ISO13606 Section Class.

ENTRY Class

The abstract entry class can be subclassed in order to record clinical statements. openEHR entries can be one of the following: Observation, Evaluation, Instruction, Action, or Admin_entry. openEHR Entry class is equivalent to ISO13606 Entry Class.

OBSERVATION

Observation is the kind of Entry whose purpose is to document all the information about observed phenomena, including any kind of measure or responses in an interview.

EVALUATION

Evaluation is the kind of Entry whose purpose is to document all the information about assessments, diagnoses, or plans.

INSTRUCTION

Instruction is the kind of Entry whose purpose is to document all the information about actionable statements such as care plans, medication orders, etc. They are statements about what should happen in the future.

ACTION

Action is the kind of Entry whose purpose is to document all the information recorded as a result of performing instructions. These are statements about things that were actually performed.

ADMIN_ENTRY

Admin entry is the kind of Entry whose purpose is to document all the administrative information.

openEHR also includes CLUSTER and ELEMENT classes similar to the ones described for ISO13606.

Figure 7 shows an overview of the openEHR information model.

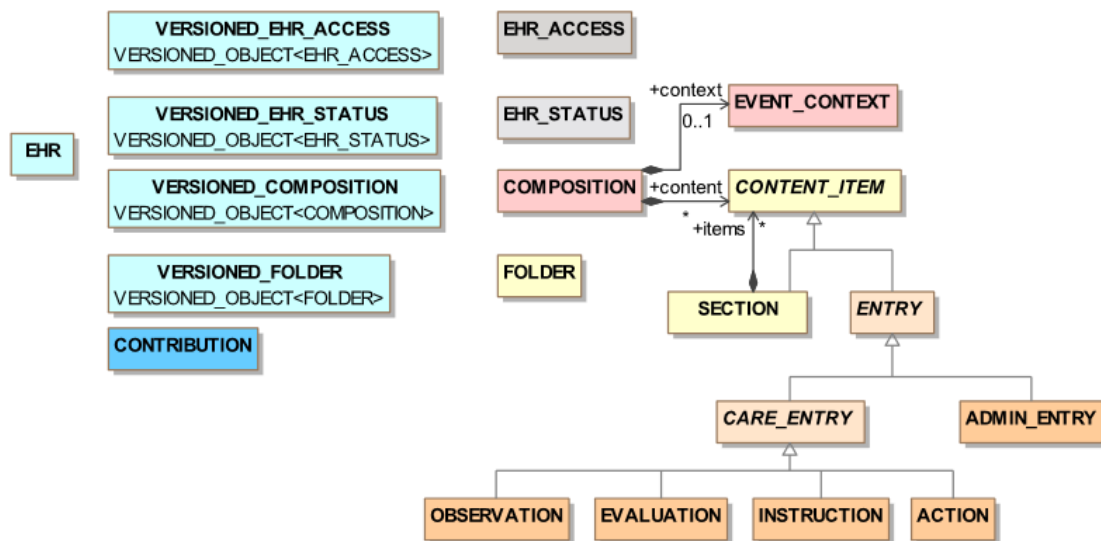


FIGURE 7 OPENEHR INFORMATION MODEL OVERVIEW. © OPENEHR

OpenEHR also contains a top level structure, the EHR, which provides access control settings (EHR_ACCESS), the current status of the EHR (EHR_STATUS), the versioned data containers (VERSIONED_COMPOSITION), a set of the changes to the EHR (CONTRIBUTION) and optionally a higher level hierarchical organizer (FOLDER).

OpenEHR archetypes are also expressed in ADL format. Figure 8 shows an excerpt from an openEHR triage archetype. The header contains the archetype identifier (openEHR-EHR-EVALUATION.triage.v1). The concept code (pointing to “Triage evaluation”) appears next, followed by the original language of the archetype (English). The description section contains

information about the original author, the archetype lifecycle or other metadata such as keywords, purpose, etc. The archetype definition section contains the Triage evaluation (EVALUATION[at0000]) containing both the data in the triage evaluation (ELEMENT[at0002]), and rationale (ELEMENT[at0008]). Triage evaluation contains an ordinal to code each one of the values to a code. Finally the ontology section contains the definition for the terms and labels used in the archetype.

OpenEHR also provides specifications for Archetype Query Language (AQL) (39), a query language based on archetypes and pattern matching, and a Guideline Definition Language (GDL) (40), a formal language for the expressing decision support logic.

```

archetype (adl_version=1.4)
  openEHR-EHR-EVALUATION.triage.v1
concept
  [at0000] -- Triage evaluation
language
  original_language = <[ISO_639-1::en]>
...
description
  original_author = <
    ["name"] = <"Heather Leslie">
    ["organisation"] = <"Ocean Informatics">
  >
  details = <
    ["en"] = <
      language = <[ISO_639-1::en]>
      purpose = <"An evaluative statement for the conclusions or summary by a clinician regarding need for priority of treatment">
      keywords = <"triage", "evaluation", "assessment">
    >
  >
...
definition
  EVALUATION[at0000] matches { -- Triage evaluation
    data matches {
      ITEM_TREE[at0001] matches { -- Tree
        items cardinality matches {0..*}; unordered matches {
          ELEMENT[at0002] occurrences matches {0..1} matches { -- Triage evaluation
            value matches {
              1|[local::at0014], -- Immediate
              2|[local::at0015], -- Very urgent
              3|[local::at0016], -- Urgent
              4|[local::at0017], -- Standard
              5|[local::at0018] -- Non-urgent
            }
          }
          ELEMENT[at0008] occurrences matches {0..1} matches { -- Rationale
            value matches {
              DV_TEXT matches (*)
            }
          }
        }
      }
    }
  }
}ontology
term_definitions = <
  ["en"] = <
    items = <
      ["at0000"] = <
        text = <"Triage evaluation">
        description = <"The evaluation of need for immediacy of treatment">
      >
      ["at0001"] = <
        text = <"Tree">
        description = <"@ internal @">
      >
    >
  >

```

FIGURE 8 EXCERPT FROM OPENEHR TRIAGE ARCHETYPE

OpenEHR has been used for several national and regional projects, such as Australia, Brazil, Norway, and Slovenia.

1.2.1.3 Dual model EHR experiences

The number of archetype-based developments is continuously growing. Every year more governments provide definitions of their clinical models as archetypes. Countries such as

Norway (41), Brazil(38,42), UK(43), Spain(35), Australia(44), or Slovenia (45) already publish their national archetypes and more countries and regions are planning to do it in the foreseeable future. The involvement of official bodies and governments has been translated into an exponential growth of available archetypes. As an example, currently there are almost 600 archetypes available in the openEHR archetype repository (16), almost a fourth of them were created in the last year and more than 250 are currently on active development. However, as there is no commonly used methodology for the definition of archetypes (46), the resulting archetypes can have great variability (47) and inconsistent terminology bindings. This is directly related with the quality of archetypes and clinical information models (48–50).

Creating good terminology bindings and improving current bindings is in fact one of key archetype research areas. The generation of semi-automatic bindings to clinical terminologies is important to reduce medical errors and to achieve interoperability between health information systems (51–54). These well-defined terminology bindings allow the correct validation (55) and management (56,57) of archetypes.

Archetypes themselves can be considered as semantic constructs and several projects have been devoted to their representation by semantic web technologies. For instance, they have been transformed into OWL (58,59), for advanced use cases such as clinical model transformation (60) or the calculation of health care quality indicators (61).

Semantic web technologies also allow the reasoning over clinical data. Clinical research and Clinical Decision Support Systems (CDSS) can benefit the most from reasoning over quality agreed definitions of the clinical models included in the systems. Advanced alert systems (62), identification of patient cohorts (63), or creation of research data warehouses from legacy data (64) are only a few examples of the joint use of archetypes and semantic web technologies.

1.2.2. Non-Dual Model Standards

1.2.2.1 HL7 Standards

HL7 v2.x Messaging

HL7 v2 (65) is a messaging standard used for the electronic exchange of clinical, care, economic, and logistic information to support workflow between applications, between organizations, or inside of an organization. HL7 v2 messages use a non-XML syntax based on segments and delimiters.

There have been several releases of HL7 v2.x (e.g. v2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.7, etc.). Every new version of v2 is designed to be backwards compatible (i.e. messages developed for a given version will be understood on applications that support newer versions).

HL7 v2.x is one of the most used healthcare standards. More than 35 countries have HL7 v2.x implementations. In fact, 95% of United States healthcare organizations use HL7 v2.x (65).

HL7 CDA

HL7 Clinical Document Architecture (CDA) (13,66) is a XML-based document markup standard for the exchange of clinical documents. It specifies the encoding, structure, and semantics of the data elements to describe the actors, actions, and events in healthcare. HL7 is based on the HL7 Reference Information Model (RIM) and uses HL7 v3 data types. CDA defines clinical documents which have the following characteristics: persistence, stewardship, potential for authentication, context, wholeness, and human readability. HL7 CDA release 2.0 was published by HL7 in 2005 and has been adopted as an ISO standard 27932 in 2009.

HL7 CDA can contain any type of clinical content, from discharge summaries to genetic testing information. CDA contains a mandatory textual part to ensure the human interpretation of the contents and an optional structured part used for automatic processing. LOINC is recommended to specify the document types, but any terminology (such as SNOMED, ICD or LOINC itself) can be used inside of the structured part to represent the contents.

The entry point for HL7 CDA is the ClinicalDocument class. A CDA document is logically divided into the CDA Header and CDA body. The purpose of the CDA header is to enable clinical document exchange between institutions, facilitate clinical document management, and ease the process of obtaining a patient lifelong electronic health record. Header contains information about the participants (e.g. patient, clinician), the custodian of the information, information about the clinical encounter and related documents. On the other hand, the CDA body can be either an unstructured blob or a structured body. A structured body contains one or more sections that can contain zero or more clinical statements (CDA entries). HL7 CDA provides the following set of entry classes:

Act class

Act class is a type of entry that should be used when no other entry class is found suitable.

Encounter class

Encounter class is used to represent related clinical encounters between healthcare provider and a patient, such as referenced past visits or follow-up visits

Observation class

Observation class is the entry used to document information about observed phenomena. Observation can be seen as similar to a non-altering procedure.

ObservationMedia class

ObservationMedia class is a derived observation that represents multimedia that is logically part of the document, such as an imaging result.

Organizer class

Organizer is a class designed to support grouping of information that shares a common context. Organizer can contain other organizers and/or other CDA entries.

Procedure class

Procedure class is the class to document information about a procedure that results in a physical alteration of the subject.

RegionOfInterest class

RegionOfInterest is the class that represents a region of interest on an image, using an overlay shape.

SubstanceAdministration class

Substance administration is the class intended to represent the administration, past or planned, of a particular substance such as medication. This also includes information about the exposure of the patient to a substance they have to be treated for.

Supply class

Supply class is used to document the information about the provision of materials between to entities, such as the medications given to a patient for a later use.

Both CDA section and entry classes can redefine the information provided in the ClinicalDocument header (e.g. author, informant, and subject). In addition to that, entries can be semantically linked to other entries contained in the same document by traversing the entryRelationship class.

HL7 CDA is the basis of several national and international projects, such as the United States Meaningful Use (67) and Europe epSOS (68) projects.

HL7 FHIR

Fast Health Interoperability Resources (FHIR) (15) is a HL7 Draft Standard for Trial Use (DSTU) open specification for the electronic exchange of healthcare information. FHIR takes advantage of the lessons learnt with HL7 v2 and HL7 v3 to provide a specification for the interoperability of healthcare information with focus on implementation ease. FHIR is based on a set of basic modular components called Resources, which describe the contents of the health records (clinical or administrative) that can be exchanged. Resources are reusable profiles defined in a common way based on other Resources and a set of data types. They also include a human readable part to ease their understanding by clinicians. Resources can be used directly, extended, or combined to satisfy most common use cases.

FHIR Resources are based on the 80-20 principle where an element will be included only if 80% of the systems implement it.

FHIR resources can be expressed in either XML or JSON. Figure 9 shows an example of a FHIR DSTU2 Medication resource definition. Every FHIR resource includes implicitly all the attributes from their parent resources, in this example 'Resource' and 'DomainResource'. Resources attributes can point either to primitive types (e.g. 'name' has data type 'string' and 'isBrand' has data type 'boolean'), complex data types (e.g. code contains a 'CodeableConcept'), or to other resources (usually in the form of references to a given resource such as manufacturer that references an 'Organization'. Figure 10 shows an XML example of an intravenous medication based on FHIR Medication resource.


```

<Medication xmlns="http://hl7.org/fhir">
  <!-- from Resource: id, meta, implicitRules, and language -->
  <!-- from DomainResource: text, contained, extension, and modifierExtension -->
  <name value="[string]"/><!-- 0..1 Common / Commercial name -->
  <code><!-- 0..1 CodeableConcept Codes that identify this medication --></code>
  <isBrand value="[boolean]"/><!-- 0..1 True if a brand -->
  <manufacturer><!-- 0..1 Reference(Organization) Manufacturer of the item --></manufacturer>
  <kind value="[code]"/><!-- 0..1 product | package -->
  <product> <!-- 0..1 Administrable medication details -->
    <form><!-- 0..1 CodeableConcept powder | tablets | carton + --></form>
    <ingredient> <!-- 0..* Active or inactive ingredient -->
      <item><!-- 1..1 Reference(Substance|Medication) The product contained --></item>
      <amount><!-- 0..1 Ratio How much ingredient in product --></amount>
    </ingredient>
    <batch> <!-- 0..* -->
      <lotNumber value="[string]"/><!-- 0..1 -->
      <expirationDate value="[dateTime]"/><!-- 0..1 -->
    </batch>
  </product>
  <package> <!-- 0..1 Details about packaged medications -->
    <container><!-- 0..1 CodeableConcept E.g. box, vial, blister-pack --></container>
    <content> <!-- 0..* What is in the package? -->
      <item><!-- 1..1 Reference(Medication) A product in the package --></item>
      <amount><!-- 0..1 Quantity How many are in the package? --></amount>
    </content>
  </package>
</Medication>

```

FIGURE 9 FHIR DSTU2 MEDICATION RESOURCE IN XML

```

<Medication xmlns="http://hl7.org/fhir">
  <id value="medexample017"/>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml"><!-- Snipped for brevity --></div>
  </text>
  <name value="Gentamycin "/>
  <code>
    <coding>
      <system value="http://snomed.info/sct"/>
      <code value="389245004"/>
      <display value="Gentamicin sulfate 80mg/100ml infusion (product)"/>
    </coding>
  </code>
  <isBrand value="false"/>
  <kind value="product"/>
  <product>
    <form>
      <coding>
        <system value="http://snomed.info/sct"/>
        <code value="440132002"/>
        <display value="Parenteral dosage form product"/>
      </coding>
    </form>
  </product>
</Medication>

```

FIGURE 10 EXAMPLE INTRAVENOUS MEDICATION INSTANCE BASED ON MEDICATION RESOURCE

Despite being still a draft, new HL7 FHIR servers and applications using FHIR API are constantly emerging (69)

1.2.2.2 *MedXML MML*

Medical Markup Language (MML) is a standard for the exchange of medical data developed in Japan in 1995. MML is a standard for the exchange of medical data from different health institutions. Since version 2.2.1, XML is used as a meta-language. From version 3.0 and onward MML conforms to HL7 CDA.

MML specification is divided into two big parts, MML common formats and MML content modules. The common formats module contains these definitions:

- Address expression: A common format for indicating addresses. A choice between a full address and an address divided in four elements
- Telephone number format: A common format for indicating telephone numbers. A choice between a separated phone number and a full telephone number
- ID format: Common format for expressing identifiers

- External reference format: A common format for expressing references to external contents
- Name expression format: Common format for expressing names. A choice between a separated name expressed as three elements and a full name expression
- Facility information format: Common format for expressing locations and facilities
- Medical department information format: A common format for expressing medical departments
- Personal information format: Common format for expressing all the available information from a person, such as names, department, addresses, phones, identifiers, etc.
- Creator information format: Common format for expressing the author. Includes the personal information and the classification of the creator (doctor, nurse, lab, etc.)

On the other hand, MML content modules are defined as follows:

- Patient information: A module to store all the demographic information known from a single patient
- Health insurance: A module to store both public and private patient insurance information. This module has some Japan specific elements
- Diagnosis record: A module to store one or more diagnosis from a patient. Supports full disease name including modifiers or name divided into a main disease part and modifiers
- Lifestyle: A module to describe and store a set of daily behaviors of a given patient
- Basic clinical: A module to describe information about allergies, blood information and infection
- Initial-consultation-specific information: A module used to describe the information of a child regarding his birth, vaccination, family history, and past history
- Progress course: A module to describe a progress course in free text form or as a SOAP (Subjective, Objective, Analysis, Plan) structure

- Surgery record: A module to describe a set of surgeries performed to a patient, with all the available context information
- Clinical summary: A module to group various items of information (such as patient information, diagnosis, surgery, etc.)
- Test history: A module to provide laboratory test result information
- Report: A module to provide reports on radiological, physiological, and pathological tests
- Referral: A module used to specify the data needed when a patient is referred from a hospital and a local clinic

MedXML MML is currently being used in several Japanese (70) and Chinese (71) regions.

1.3. Terminologies

Terminologies, and more precisely clinical terminologies are structured vocabularies (or lists of terms) used in clinical practice to describe accurately and unambiguously the care and treatment of patients. These vocabularies or terms cover concepts such as diseases, operations, drugs, or treatments.

As the way of describing different kinds of terminologies usually is confusing as different authors have used the same words differently (e.g. 'ontology' or 'knowledge'), we will use the glossary of terms proposed by SemanticHealth European Project (2) for the clear definition of terms in this context.

- Controlled Vocabulary: A list of specified items to be used for some purpose, usually in an information system to reduce ambiguity, misspellings, etc.
- System of identifiers ("codes"): Controlled vocabularies and many lexicons, ontologies and thesauri are usually accompanied by systems of identifiers for their units, e.g. typically, identifiers act as the primary unambiguous means of referring to the entities in the system for computational purposes with the text form being used for communication with users. Examples include the Concept Unique Identifiers(CUIs) from UMLS, the SNOMED identifiers, etc. In many contexts, identifiers are known as "codes."
- Lexicon: A list of linguistic units that may be attached to a controlled vocabulary or ontology, in a specific language or sublanguage, often including linguistic information

such as synonyms, preferred terms, parts of speech, inflections and other grammatical material. Example: Term terms and lexical material in UMLS identified by Lexical Unique Identifiers LUIs).

- **Ontology:** A symbolic logical model of some part of the meanings of the notions used in a field, i.e. those things that are universally true or true by definition. The key relationship in an ontology is "subsumption" or "kind-of". Every instance of a subkind must be an instance of the kind, without exception. Typically, ontologies are implemented in logic languages such as Ontylog or OWL or frame systems such as Protégé-Frames. Examples: The GALEN Core Model, the stated form of SNOMED.
- **Classification:** An organisation of entities into classes for a specific purpose such as international reporting or remuneration. Examples ICD and Diagnosis Related Groups.
- **Thesaurus:** A system of terms organised for navigation with the primary relationship being "broader than"/"narrower than". The "broader than"/"Narrower than" relation is explicitly not limited to subsumption/kind of relation. It is a general form of linguistic hyper/hyponymy aimed at assisting human navigation. However, it is explicitly not intended that it be used as the basis for logical interferences, e.g. in decision support. Examples MeSH, WordNet.
- **Knowledge Representation System / Background knowledge base:** The common knowledge to be assumed by the system, including both the ontology – what is universally true – and generalisations about what is typically true.
- **Terminology:** Any or all of the above in various combinations. Most health terminologies consist, at a minimum, of a controlled vocabulary and a system of identifiers. They may include extended lexicons, ontologies, thesauri or background knowledge base. This definition is deliberately broader and less specific than that in most of the standard references and intended to approximate common usage.
- **Coding system:** A terminology with attached identifiers or "codes".

1.3.1. Relevant Terminologies and ontologies

SNOMED CT

SNOMED CT (20)(Systematic Nomenclature of Medicine – Clinical Terms) is clinical terminology result of the fusion of SNOMED RT (Reference Terminology) developed from the College of American Pathologist (CAP) and Clinical Terms Version 3 (CTV3) from the UK National Health Service (NHS). Currently IHTSDO (International Health Terminology Standards Development Organization) has SNOMED CT development and distribution rights. IHTSDO is a Non-for-profit

(NPO) international organization established in Denmark. This organization was founded in 2007 by 9 countries (Australia, Canada, Denmark, Lithuania, the Netherlands, New Zealand, Sweden, United Kingdom, and United States of America) with the objective of maintaining and developing international clinical terminologies. This is the reason because IHTSDO purchased the intellectual property of SNOMED CT. Currently IHTSDO has over 25 members, and Spain is a member of IHTSDO since 2009. New versions of SNOMED CT are released twice a year, with a Spanish version released at the same time.

SNOMED is the most comprehensive, multilingual clinical vocabulary available in English or any other language. SNOMED CT contains more than 300.000 active concepts, their descriptions and relationships. Each SNOMED CT concept represents a clinical thought. Concepts have a numeric identifier and are included in a hierarchy, organized from the general to the more detailed.

Each concept can be further described by various clinical terms or phrases called Descriptions, which are divided into Fully Specified Name (unique across all SNOMED CT), Preferred Terms (selected by a group of clinicians as the most common way of expressing the meaning of a concept) and synonyms (which are additional ways to refer to this concept). Each concept has exactly one unambiguous Fully Specified Name, exactly one preferred term, and zero to many synonyms. An example of an SNOMED CT structure can be seen in Figure 11.

```
ConceptID: 22298006
• Fully Specified Name: Myocardial infarction (disorder)
• DescriptionID: 751689013
• Preferred Name: Myocardial infarction
• DescriptionID:374436014
• Synonym: Infraction of heart
• DescriptionID: 37441018
• Synonym: Heart attack
• DescriptionID: 37443015
• Synonym: Myocardial infract
• DescriptionID: 178483012
```

FIGURE 11 SNOMED CT EXAMPLE CONCEPT

SNOMED CT concepts can also be linked to other concepts whose meaning is related in some way by relationships. These relationships provide formal definitions and properties for the concept. One of the most common relationships in SNOMED CT is the 'is a' relationship, which define hierarchies in the terminology (e.g. Myocardial infarction 'is a' Myocardial disease).

Figure 12 shows an example of the relationships in SNOMED CT for Myocardial infarction concept.

```

ConceptID: 22298006
• Fully Specified Name: Myocardial infarction (disorder)
• DescriptionID: 57809008
• Is a: Myocardial disease (disorder)
• DescriptionID:374436014
• Is a: Necrosis of anatomical site (disorder)
• DescriptionID: 609410002
• Associated morphology: Infarct (morphologic abnormality)
• DescriptionID: 55641003
• Finding site: Myocardium structure (body structure)
• DescriptionID: 74281007

```

FIGURE 12 SNOMED CT RELATIONSHIPS EXAMPLE

One of SNOMED CT characteristics is that concepts are usually the result of linking already existing concepts. This is called pre-coordination. In the same way, new concepts that do not appear in the terminology can be expressed as a result of existing concepts. This is called post-coordination.

IHTSDO has also provided a compositional grammar (72) for the representation of SNOMED CT expressions.

ICD

The International Classification of Diseases (ICD) (73) is the standard diagnostic tool for epidemiology, health management, and clinical purposes. It is especially useful for the analysis of general health situation in countries and populations. Most countries part of World Health Organization (WHO) use ICD to report mortality data, which is a primary indicator of health status.

Currently, most systems use ICD 9, which is being replaced by ICD 10. There is also an ICD 11 in draft form that is expected to be approved by 2018.

LOINC

Logical Observation Identifiers Names and Codes (LOINC) (74) is a terminology standard for identifying laboratory and clinical observations. LOINC provides a catalog of laboratory tests, clinical, and anthropomorphic measures. LOINC initiated at 1994 in the Regenstrief Institute. It is endorsed by the American Clinical Laboratory Association and the College of American

Pathologists. LOINC uses six fields (or axis) for the unique specification of each test, observation, or measurement:

- Component: What is measured, evaluated, or observed.
- Kind of property: Which are the characteristics of what is measured (e.g. mass, substance, catalytic activity).
- Time aspect: When was measured, evaluated, or observed (typically a time or interval of time).
- System type: What context or specimen type within the measurement was made (e.g. blood, urine)
- Scale: Which kind of scale was used in the measure. It can be quantitative, ordinal, nominal, or narrative.
- Method: What procedure was used to perform the measurement. This axis is optional and is used to distinguish between different measurements when every other axis does not provide enough information.

1.3.2. Archetype terminology binding

One of the basic parts of an archetype definition are the terminology bindings. These bindings give the archetype an exact meaning. Archetypes allow the definition of terminology bindings in the form of label bindings and value bindings.

Label bindings describe the equivalences between archetype local terms and external terminology terms. They provide clear meaning to the archetype labels and can be used for their semantic description.

Value bindings provide the means to bind value constraints in the archetype as sets of terms from an external terminology. These sets of terms, called usually subsets, can be defined in an extensional manner (by enumerating all the terms in the subset) or in an intensional manner (by defining the necessary and sufficient conditions for belonging to the set). This mechanism is used for defining which terminology subset is valid in a given coded text constraint. The definition of subsets is a key issue, as each implementation usually requires a particular set of concepts, descriptions, and relationships. IHTSDO provides an expression syntax for the definition of these SNOMED CT subsets (72).

These different types of bindings are stored into the archetypes section called 'ontology' (which will be renamed to 'terminology' in ADL2) where node identifiers, constraints on texts

or terms, and bindings to terminologies are defined using dADL. This section is divided into four subsections: term definitions, constraint definitions, term bindings, and constraint bindings. Term definitions refer to label bindings and Constraint definitions refer to value bindings.

1.4. Data, Model, and Format transformation

1.4.1. Data transformation

Existing clinical data must be transformed to meet the data structures and constraints defined by reference models and archetypes. We face a problem known in the literature as the data exchange (translation or transformation) problem (4). Data exchange is the problem of generating an instance of a target schema from a source schema and a given set of relationships between both. More precisely, data exchange is a quadruple expressed as $(S, T, \Sigma_{st}, \Sigma_t)$, where S is the source schema, T is the target schema, Σ_{st} is the mapping expressing the relationships between S and T , and Σ_t is a set of constraints on T . The data exchange problem can be defined as: given an instance I over source schema S , find an instance J over target schema T such that both I and J satisfy the relationship Σ_{st} and J satisfies Σ_t . That J instance is called a solution for I in the data exchange setting. Different solutions can exist for a given I instance, and the challenge is to find the best one (75).

Typically, the explicit relationships Σ_{st} are called schema mappings (5,6). Research in this area has focused on the formal specification of schema mappings and their semantics (4,6), most of the formalisms use a subset of first-order logic to specify them. There are three basic approaches for specifying the mapping in the literature: global-as-view (GAV), local-as-view (LAV) and global-local-as-View (GLAV) (5). In GAV systems the intermediate schema is defined in terms of the source schemas. Alternatively, in LAV systems each element of the source schemas is defined in terms of the intermediate schema. LAV mappings are suitable, in particular, when the intermediate schema is based on an enterprise model, standard or an ontology (76). Conversely, GAV approach is suitable when the data sources are stable since the addition or modification of a source schema would require the redefinition of various elements of the intermediate schema. GLAV approach (77) generalizes LAV and GAV and allows flexible schema definitions independent of the particular details of the sources. Intuitively, GLAV mappings relate a query over the source schema S to a query over the target schema T .

Schema mappings are used in two different, but highly related problems: data integration (also known as data federation) (5) and data transformation (4). In data translation users pose

queries on the target schema that are answered using the local data sources. The key point here is that the translated source data are not materialized in the target, i.e. no target instance is generated. In contrast, in data transformation the data structured under the source schema is transformed into data structured under the target schema. Any target query is then answered using the materialized target instance without reference to the original source instance.

In data transformation, source-to-target dependencies are the prevailing option to specify how and what source data should appear in the target. Source-to-target dependencies are assertion between a source query and a target query and therefore are a kind of GLAV mappings. Formally, a source-to-target dependency has the form:

$$\forall x(\phi_s(x) \rightarrow \chi_T(x))$$

Where $\phi_s(x)$ is a formula in some logical formalism over the source schema S and $\chi_T(x)$ is a formula in some (perhaps different) logical formalism over T [6].

Among the different types of source-to-target dependencies source-to-target tuple-generating-dependencies (TGDs) is the most commonly used specification for schema mappings. TGSs have the form:

$$\forall x(\phi_s(x) \rightarrow \exists y\psi_T(x, y))$$

where $\phi_s(x)$ is a conjunction of atomic formulas in some logical formalism over the source schema S and $\psi_T(x, y)$ is a conjunction of atomic formulas in some (perhaps different) logical formalism over T (75). A full TGD is a TGD with no existential quantifiers in the right-hand side. Figure 13 shows an example of TGD and Figure 14 an example of full TGD:

$$\begin{aligned} &\forall \text{diag} \forall \text{date}(\text{diagnostic}(\text{diag}) \wedge \text{ends}(\text{diag}, \text{date})) \\ &\rightarrow \exists \text{initDate} \text{closedProblem}(\text{diag}, \text{initDate}, \text{endDate})) \end{aligned}$$

FIGURE 13 EXAMPLE OF TGD

$$\forall \text{diag} \forall \text{date}(\text{diagnostic}(\text{diag}) \wedge \text{ends}(\text{diag}, \text{date}) \rightarrow \text{closedProblem}(\text{diag}, \text{endDate}))$$

FIGURE 14 EXAMPLE OF FULL TGD

TGDs offer a balance between high expressive power and good algorithmic properties (6). The unrestricted use of first-order logic as a schema-mapping specification gives rise to undecidability of basic algorithmic problems about schema mappings.

The use of high-level schema mappings, such as TGDs, makes it possible to separate the specification (design) from the implementation. This is a crucial issue since the effort required to create and manage data transformations is considerable, as it may involve writing and managing complex data transformations programs. This is even more complex when one deals with EHR standards due to their complexity. High-level mappings are easier to specify and manage than executable scripts or programs such as XSLT, XQuery or SQL. Therefore, schema mapping generation has received lots of attention. It studies how to compile into a processable language a mapping specification between two schemas (4,78–80). Systems supporting data exchange and schema mapping generation are more necessary than ever, as new formats, technologies, and approaches are continuously emerging, for instance archetypes. Schema generation tools usually provide GUIs that place the source schema on one side and the target schema on the other side. Users can specify the high-level assertions by drawing lines connecting source and target elements. Examples of modern data exchange systems include Clio (81), HePToX (82), EIRENE (83), CLIP (78), and MapMerge (84). Clio and CLIP are particularly interesting in our scenario since they are capable of dealing with hierarchical data structures.

In the health care domain very few generic EHR data transformation efforts exist. Although several research prototypes and commercial mapping tools are capable of processing XML schemas they cannot handle archetypes. The reason is the lack of expressivity of XML Schemas due to the unique particle attribution constraint rule. This rule is violated by most archetypes. Therefore, such tools cannot be used for archetype-based data transformation. Furthermore, archetypes are used to model arbitrary complex domain concepts without any consideration regarding the potential internal architecture or database design of EHR systems. As a consequence, complex and expressive mapping specifications are necessary due to the low similarity between archetypes and EHR systems. Nevertheless, some previous experiences exist, probably the most interesting is the work by Duftschmid et al. (85,86). They proposed an approach for transforming data from an Entity-Attribute-Value based EHR into XML documents compliant with ISO 13606. It was based on mapping the structure of the local EHR systems (described by a generic XML schema) to archetypes, which are, in turn, also expressed as XML Schemas. In order to overcome the unique particle attribution constraint they rename some

schema elements. The main disadvantage of this approach is that a specific XML schema has to be created for each archetype making it complex and not completely archetype-based.

1.4.2. Model-based transformations

Model-Driven Development (MDD) tries to improve correctness and productivity in software creation by producing software from modeling diagrams created by humans. Model-Driven Architecture (MDA) (87)(88) is the OMG (89) proposal to support Model-driven engineering. In MDA the applications and business processes are specified using Platform-Independent Models (PIM), which define the functionality of the systems. PIMs are then transformed into Platform-Specific Models (PSM) and final implementation languages with standard mapping techniques. MDA also defines a layer to bridge the gap between domain experts and IT people called Computation Independent Model (CIM). Requirements expressed in CIM should be traceable to both PIM and PSM implementations.

MDD is similar to dual model approach, as both put models as key parts of the methodologies. In fact, archetypes can be considered as MDA CIMs (90), and have been used as such in several projects. E.g. for the automatic generation of Graphical User Interfaces in an endoscopy reporting application (91), the creation of an agile EHR web framework from archetypes (92), or combined with semantic web technologies for the translation between reference models (namely, ISO13606 and openEHR)(93).

1.4.3. Format transformation

Promoting interoperability in healthcare infrastructure through shared artifacts generated from formal model definitions is the main goal of initiatives such as Model-Driven Health Tools (MDHT) Project (94) from Open Health Tools (OHT). MDHT is an open source effort for the promotion of shared artifacts between related standards and the creation of modeling tools for their seamless integration. The project is supported by the US Veteran's Health Administration (VHA), IBM, and the US Office of the National Coordinator (ONC). Their original focus was to develop HL7v3 specifications via UML, but they later moved to work in the specification of HL7 CDA Implementation Guides. They have provided models and reference implementations for several HL7 C-CDA Implementation Guides. They are planning to support other standards besides HL7 CDA, for instance by using UML for the specification of archetypes. A UML profile (Archetype Modeling Language, AML) has been proposed to OMG to deal with the specific requirements of the archetype modeling. MDHT is also working in the generation of Schematron for XML instance validation.

1.5. Constraint definition languages

There exists a wide range of formal rule languages for the definition of constraints on data. One of the most known is the Object Constraint Language (OCL) (30), an OMG (89) standard for the definition of rules over UML models (95). There are also languages for defining Horn-like rules for the Ontology Web Language (OWL) (96), such as Semantic Web Rule Language (SWRL) (97) or RuleML (98). The widespread use of rules, formal or not, has caused the creation of proposals, like the W3C Rule Interchange Format (RIF) (99), for the exchange of rules between different rules languages. The main disadvantage with most rule languages is that rules are not easily understood by non-technical staff. To solve this problem, some rule languages with natural language-like syntax have been proposed. Two main examples are Natural Rule Language (100) and Attempto Controlled English (101). Each one of them addresses the problem of natural language rules representation from a different perspective.

Human-readable validation languages

Natural Rule Language (NRL) is a formal language for specifying constraints and rules in a human readable way. The main feature of this language is the capacity of defining constraints in a way that facilitates their understanding by non-technical people. Moreover, NRL also defines an extension to deal with actions, such as the creation or deletion of objects, or setting values when certain conditions are met. Although we will not use this extension, it could be used to complete the rules with actions, for instance to calculate derived values. There is only one prior use of NRL in the clinical domain, concretely for the representation of clinical practice guidelines and its evaluation in a real world case (102). Rules drawn from a hypertension guideline were translated into NRL in order to be validated by clinicians and subsequently they were transformed into OCL and finally used in the system. The NRL rules were generated by hand which can be a time-consuming task.

Attempto Controlled English (ACE) (101) is a controlled natural language, which means that it is a subset of Standard English with a restricted syntax. ACE can be translated into other languages, such as RuleML, OWL, or SWRL. The meaning of words in ACE is not predefined and must be defined in an existing ontology or in additional ACE sentences. Although ACE has been in use for more than ten years, it only has been used once applied to the clinical domain (103), specifically for clinical guidelines readability. In this work, rules from a pediatric clinical guideline were expressed in ACE, although they were not applied to real data.

XML validation languages

XML documents contain specific characteristics that also need to be validated. These specific constraints can be validated with technologies such as Schematron, DTD, or XML Schema. Schematron (104) is a rule-based validation language for making assertions about patterns in XML trees that is an ISO norm since 2006. Since it is a path based validation language, Schematron can express constraints that neither XML Schema nor DTD can express. Each rule can be associated with a descriptive text of the type of error or warning encountered. Schematron plays a key role on current CDA implementations as Schematron rules are typically attached to implementation guides alongside sample XML instances. It has been proved that Schematron rules can be directly generated from NRL rules (105) as well as from archetypes (106). Advanced features of archetype methodology, such as reuse of internal or external types can be also reproduced with Schematron.

Drools

Drools (107) is an open source Business Rule Management System (BRMS) written in java for the centralization and management of business logic. In addition to provide syntax based on First Order Logic to describe the rules, Drools also provides a business rule engine for their execution. Drools uses an optimized version of Rete algorithm for object oriented systems. Drools also supports the definition of Domain-specific languages (DSL) to write rules in natural language. Drools uses a knowledge base for the collection of compiled definitions (such as rules and processes). Knowledge base can be updated from inside the rules in order to insert, update, or delete objects to it.

Both openEHR Guideline Definition Language (GDL) (40) and open source project openCDS (108) use Drools for the implementation of their execution engine.

1.6. Interoperability projects

SemanticHEALTH

SemanticHealth (2) was a European Project from the 6th Framework Programme. The project objective was to identify the steps needed to achieve semantic interoperability. For this, SemanticHEALTH developed a roadmap for semantic interoperability of EHR, focusing on patient care, clinical research and public health.

SemantiHEALTH identifies four levels of interoperability:

- Level 0: No interoperability at all. E.g. the patient must repeat tests in order to know what is happening.
- Level 1: Technical and syntactical interoperability (no semantic interoperability). E.g. Doctors are able to retrieve electronic documents on the original language.
- Level 2: Partial semantic interoperability
 - Level 2a: unidirectional semantic interoperability. E.g. the electronic documents can be accessed remotely and few parts (demographics, diagnosis, etc.) can be understood by the receiving system.
 - Level 2b: bidirectional semantic interoperability. E.g. same as above, but both ends of the system can do it.
- Level 3: Full semantic interoperability, sharable context, seamless co-operation. E.g. the foreign Hospital Information System (HIS) can access, interpret, and present all necessary information about the patient.

SemanticHEALTH recommends the use of generic reference models (ISO13606, openEHR, and HL7 CDA R2), archetypes or templates, and clinical terminologies such as SNOMED CT and LOINC (74) as needed to achieve level 3 interoperability.

SemanticHealthNet

SemanticHealthNet (SHN) (109) was a European Union Project from 7th Framework Programme. SHN developed a scalable and sustainable pan-European organizational and governance process for the semantic interoperability of clinical knowledge. SHN follows the recommendations from SemanticHEALTH project for the integration of clinical information models, ontologies, and terminologies to achieve semantic interoperability. The project used heart failure use case to capture the needs from patient and public health perspectives. Different standards were used for the modeling of the use case, which were included in an ontology framework to achieve semantic interoperability.

epSOS

European Patients – Smart Open Services (epSOS) project (68) was a 7th Framework Programme. Project team consisted of 22 European Union (EU) and 3 non-EU member states. epSOS project was aimed to design, build, and evaluate a service infrastructure that can provide cross-border interoperability between EHR systems in Europe. A cross-border patient summary and electronic prescription pilots were demonstrated.

Meaningful use

Meaningful Use (MU) (67,110) is a Medicare (111) and Medicaid (112) United States (US) government program that awards incentives for using certified EHR to improve the exchange of clinical data between healthcare providers, between providers and insurers, and between providers and patients. The program defines three stages for a gradual EHR adoption.

- Stage 1 is focused in basic EHR adoption.
- Stage 2 is focused in advanced clinical processes such as clinical decision support.
- Stage 3 is focused in health information exchange and improvements of healthcare incomes

MU policy outcome priorities is to improve quality, safety, efficiency, care coordination, population and public health, reduce health disparities, to engage patients in their health, and to ensure privacy and security of personal health information.

Trillium Bridge

The Trillium Bridge support action (113) extends epSOS and Meaningful use to establish an interoperability bridge for the exchange of patient summaries and EHR among EU and US. Trillium Bridge objective is to make epSOS and Meaningful use outcomes compatible by identifying the misalignments and provide solutions to them.

Expand

Expanding Health Data Interoperability Services (EXPAND) (114) is thematic network created for the integration and deployment of the results of the relevant eHealth European projects pilots and deploy them as large-scale cross border services. Expand started in January 2014 and is expected to finish by the end of 2015. 17 countries are currently represented in the thematic network

Clinical Information Modeling Initiative (CIMI)

Clinical Information Modeling Initiative (CIMI) (19) is an international collaboration whose objective is to provide a common format for representing shared implementable clinical information models. CIMI is formed by both individuals and organizations including standards bodies (CEN, HL7, IHTSDO, CDISC), national agencies (NEHTA, NHS), and software developers (SMART, Tolven). CIMI uses ADL 1.5 as the modeling formalism, and SNOMED CT as the primary reference terminology. CIMI will also make use of AML profile (Archetype Modeling

Language UML profile) for the definition of the clinical models. CIMI provides a Reference Model and a set of models and patterns in an open repository.

1.7. LinkEHR Normalization Platform

The work of this thesis is based on LinkEHR normalization platform. LinkEHR is a modular platform whose objective is to facilitate the achievement of semantic interoperability of biomedical information. LinkEHR Normalization Platform allows the creation of a normalized virtual federated view of the EHR of a patient with data distributed among heterogeneous systems, as long as the original data is accessible. LinkEHR modules can be used standalone or combined together as a complete EHR integration and normalization system. All the developments of this thesis have been incorporated into different modules of the platform, mainly in the archetype editor.

1.7.1. LinkEHR Integration Engine

LinkEHR integration engine (115) is a lightweight, fully scalable integration engine to access multiple heterogeneous and distributed data sources, launch queries over them, and integrate all the results into a single XML document.

To know what part of the EHR can be shared, LinkEHR-IE provides the Integration Message Definitions (IMD). IMD definitions include the specification on the data sources and the tables and fields that will be used for each source. It also includes query parameters to filter the results, and the nested labeled structure constituting the output XML document structure. IMD is the minimum communication unit, i.e. petitions to LinkEHR-IE are based on IMD with parameters. When a request is received by the system, the corresponding IMD is executed, which queries the original legacy sources into an integrated XML view. The output of this process is an XML with a known format, which allows us to apply further transformations for data visualization or the normalization of the unified view

1.7.2. LinkEHR Archetype Editor

LinkEHR Editor is a framework for editing archetypes based on different reference models. LinkEHR Editor is not the only framework that supports the review of archetypes from different reference models (3). However, usually the reference models are hardcoded in these tools, which makes adding new model or evolving current ones a difficult task. LinkEHR Editor allows the inclusion of new reference models based on the analysis of the reference model schemas.

For all classes available in the source schema, only a subset of them is suitable to be used to define archetypes. These are called the reference model business concepts. These business classes are different for each reference model. E.g. ISO13606 has six business classes: Folder, Composition, Section, Entry, Cluster and Element, openEHR on the other hand has more: Composition, Section, Observation, Evaluation, Instruction, Action, Admin_entry, Item_tree, Item_list, Item_table, Item_single, Cluster, and Element.

As long as the XML Schema is available, any reference model can be imported into LinkEHR Editor. In the import process, users need to specify the business classes from all the classes available in the model. The available classes are sorted by complexity in order to easily find the business classes over all class set. With the business classes selected, a module analyses the schemas and generates a set of archetypes that represent the reference model. These archetypes are called business archetypes or reference model archetypes. This method has been tested with several reference models, such as ISO13606, openEHR, HL7 CDA (and CCD), CDISC ODM, and ASTM CCR.

Having the reference model represented as archetypes allows reusing the same process for archetype creation that we use for archetype specialization, i.e. new archetypes are archetypes that specialize the reference model archetype. By using the business archetype, the editor guides the edition process to guarantee that the archetype will follow the reference model, as only allows to constraint the types or attributes allowed at any point in the archetype.

LinkEHR Editor also supports the edition of ADL syntax by hand with an included ADL editor. It is possible to go from the tree-based editor to the ADL editor at any time. However, this functionality provides a major challenge for the archetype creation: Syntactically valid ADL are not necessary semantically valid (e.g. classes have been correctly used in the correct place in the hierarchy). There is a need of having formal methods for the validating the design and contents of the archetypes (116). An archetype is valid if the constraints are compatible with the ones in the reference model and the parent archetype (if there is one).

To validate this, we assume that archetypes are labeled trees. Every constraint on an archetype is expressed by either a regular expression, which describe a set of valid labels, or a label predicate. We can formalize the inheritance relationship of archetypes by a subsumption relation (117) based on the containment of regular expressions and label predicates. We say that an archetype specializes another if the other archetype subsumes it. An exhaustive

explanation of the subsumption applied to archetypes we refer the reader to (27,118). We use this approach in LinkEHR Editor for the validation of the archetypes being edited.

1.7.3. LinkEHR Extract Server

LinkEHR extract server provides a simple web service interface for accessing the integration and normalization modules. LinkEHR extract server partially implements ISO13606 part 5 (34) for an extract server. LinkEHR extract server supports the querying of data by several parameters, such as archetype identifier, patient identifier, or time period. LinkEHR extract server can be deployed in a distributed environment, i.e. an instance of the server can be deployed in each organization and all of them can be queried from a central instance of the server.

1.7.4. LinkEHR Viewer

LinkEHR viewer is a web-based generic EHR viewer for existing clinical information. It provides a read-only view of patient EHR. It provides an user interface for user authentication and patient search. As LinkEHR viewer is not a complete EHR system infrastructure, it can be deployed on top of existing HIS or be integrated into them in order to provide access to all the available information.

1.7.5. LinkEHR Concept Manager

LinkEHR concept manager (119) is a web application for the publication, management, and governance of clinical information models and other reference materials including archetypes, templates, or schematron rules.

As all the aforementioned LinkEHR platform modules, LinkEHR concept manager supports multiple reference models and formats. The manager is focused on the management of generic concepts which have attached definitions in several standards and formats, e.g. ISO13606 archetypes, openEHR archetypes, HL7 CDA templates, or XML Schemas.

The application handles the versioning, specialization, validity period, and lifecycle management of clinical information models. It provides the possibility of defining relationships between clinical information models (such as specializations, inclusions, exclusions, etc.) providing a graphical representation of these relationships. Figure 15 shows an example of how models relationships are stored in LinkEHR concept manager.

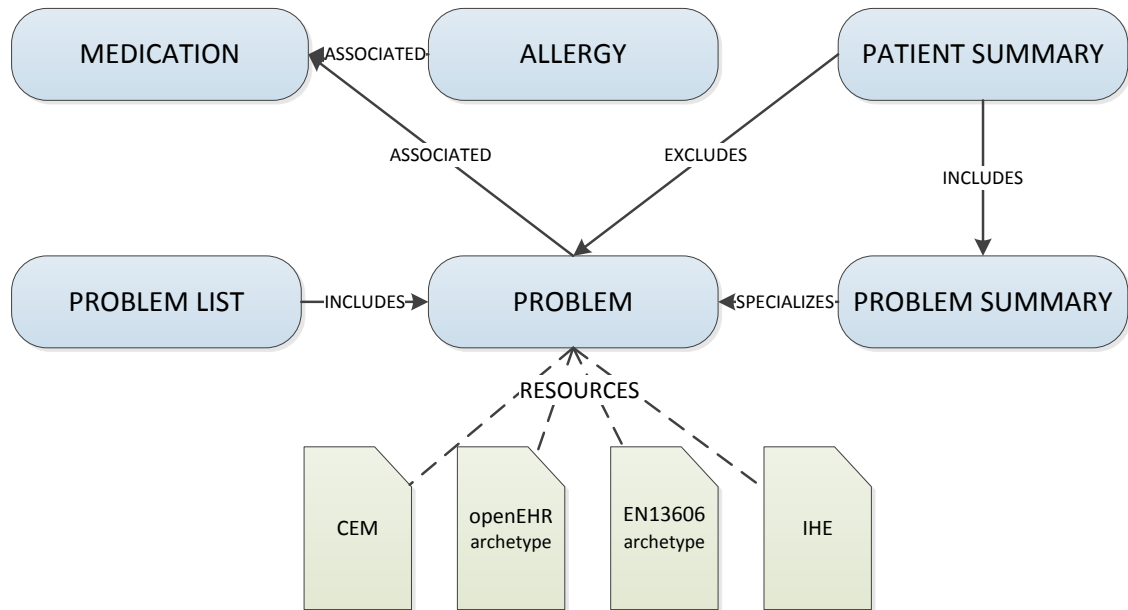


FIGURE 15 EXAMPLE OF RELATIONSHIPS AND RESOURCES FOR PROBLEM CLINICAL MODEL

LinkEHR concept manager follows a social network approach to encourage clinicians' engagement. Clinicians can subscribe to a given clinical model or archetype to be informed of changes to that model. LinkEHR concept manager allows the definition of roles that have a set of rights assigned to them in order to manage what each user is allowed to do in the platform.

Chapter 2.

Migration of Health Information Systems – Model perspective

2.1. Introduction

Dual model methodology allows the formal description of the clinical models of a given EHR information model. This formal definition provides a set of advantages, such as knowledge reuse, terminology bindings, or multilinguality. Usually traditional EHR architectures lack such a formal mechanism. However, as long as the architecture supports directly or indirectly the notion of detailed clinical models, they can be used as a reference model for the archetype definition. This chapter describes a set of methodologies and developments to support the definition of archetypes for EHR architectures that do not support dual model architecture natively.

Parts of this chapter were presented at the Medical Informatics Europe (MIE 2015) conference under the name “Combining Archetypes with Fast Health Interoperability Resources in Future-proof Health Information Systems”¹, at Medinfo 2013 conference under the name “Genetic testing information standardization in HL7 CDA and ISO13606”², and published in the Indian Journal of Medical Informatics (IJMI) under the name “Reforming MML (Medical Markup Language) Standard with Archetype Technology”³. The creation of MML Reference Model and all the required developments were carried out during a Research Internship at the Graduate School of Informatics in Kyoto University.

¹ Open access. Accessible at <http://ebooks.iospress.nl/volumearticle/39320>

² Open access. Accessible at <http://ebooks.iospress.nl/volumearticle/34014>

³ Open access. Accessible at <http://ijmi.org/index.php/ijmi/article/view/284>

2.2. Reference model archetypes

Only a subset of the classes contained in reference models define logical building blocks of EHRs and can be used to define archetypes. We call these classes ‘business concepts’. For instance ISO EN13606 defines six business concepts, namely: Folder, Composition, Section, Entry, Cluster, and Element. The representation of a business class as an archetype is what we call a Reference Model Archetype (RMA).

A RMA contains all the attributes and classes of the reference model that are used to define the business concept. For instance, the business class Element of ISO 13606 represents the leaf nodes within the EHR hierarchy. Each instance of Element has a single data value (attribute value), which is one of a defined set of data types (Boolean, coded value, physical quantity, etc.). In the corresponding RMA all the possible data types are explicitly defined as an alternative for the value attribute. RMAs represent the most general archetypes that can be defined based on a reference model and hence any other archetype must be a specialization of one of them.

The main consequence here is that with RMAs the archetype editing becomes a process of subtyping by constraints (118). The rules used to control the archetype editing are those specified in the archetype model such as strengthening of domain constraints on primitive attributes or the narrowing of cardinality intervals. In other words, the same logic can be applied both to the specialization of an existing archetype and to the definition of a new archetype as shown in Figure 16. This has also interesting consequence for archetypes validation, as the validation with respect to a reference model becomes a problem of finding a subsumption function to the corresponding RMA. Those for which it is not possible to find such functions are considered invalid with respect to the reference model. With RMAs the editors can be then independent of the reference model as long as it is possible to create the set of RMAs for a particular EHR information model. This approach was first implemented in LinkEHR archetype editor(118), making it the first editor capable of handling multiple reference models. In the rest of publicly available editors the reference model was hard-coded meaning they only supported one reference model (namely openEHR reference model).

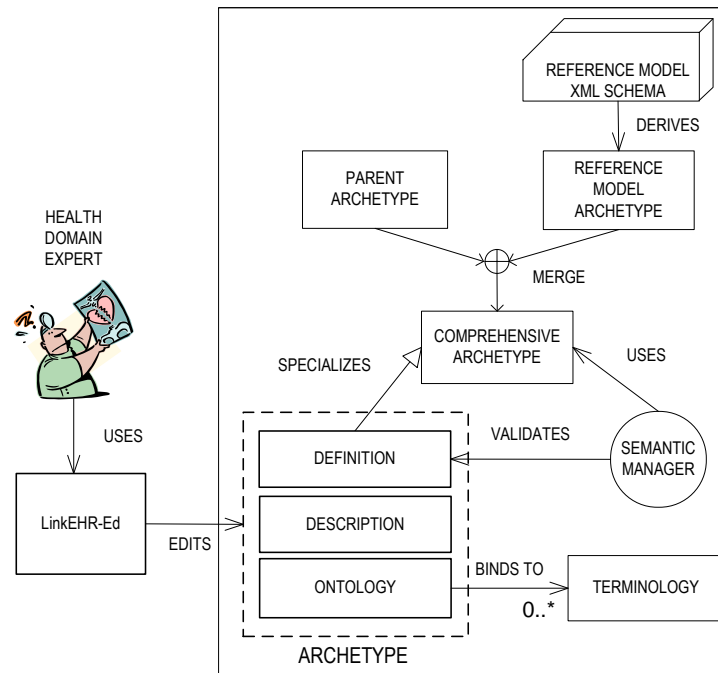


FIGURE 16 LINKEHR ARCHETYPE EDITION

2.3. Creation of RMA

As seen above, in order to allow the edition of archetypes in a given model we must first create the set of RMA for that model. Usually the standards provide some kind of information model which can be expressed in formats such as XML Schema Definition (XSD). In (118) we presented a methodology for the generation of RMA from XSD.

However, the process of deriving the reference model is not always possible by different reasons such as the schema being not public (e.g. DICOM SR XML Schema (120)), being in a non-supported format (e.g. MedXML MML provides DTD schemas and openEHR official schemas are provided in Basic Meta Model or BMM(121)), or the complexity and size of the schema makes the process impractical (e.g. HL7 FHIR DSTU and specially DSTU2 as the Resource number increases). For these cases, alternative processes for generating the RM are needed. To support these new RM we generate the RMA, which are archetypes containing explicit and exhaustive definitions of the structure and possible contents for each business class. We have used different approaches for each one of the aforementioned standards, which exemplifies the different approaches when importing new reference models:

- Creating RMA from meta-models: This approach was used for the creation of HL7 FHIR DSTU reference model and the creation of BMM derived reference models.
- Creating RMA manually: This was used for the creation of MedXML MML reference model

2.3.1. Creating RMA from meta-models

For this process, the meta-models are analyzed and RMA are created from them. The problems are still the same as the ones found in the creation of RMA from XML Schema (118): How to determine which classes are archetypable (i.e. which classes we want to define archetypes from), and how to deduce the structure. We exemplify this approach for the generation of HL7 FHIR RMA from an ecore model and the creation of RMA from BMM files.

OMG provides an standard format for the serialization of models in XML called XML Metadata Interchange (XMI) (122). XMI is an ISO norm since 2005 (2005 ISO/IEC 19503 and 2014 ISO/IEC 19509). XMI suffers a number of issues, such as being too complex or not having a consistent serialization format among different tools (123). Eclipse Modeling Framework (EMF) (124) provides its own meta model (ecore) for describing models and runtime constraints, and it is also serializable in XMI. HL7 FHIR DSTU provides the reference model in both XML Schema and ecore definition (125). FHIR XML Schemas were too complicated to be analyzed by the XML Schema reference model import due to the number of Resources, as each FHIR Resource is potentially a RMA, and the possibility to include almost all resources into one another due to the extension mechanism. For this reason, we created an iterative process that transforms each one of the selected Resource types defined in the ecore model into RMA. The process for the creation of FHIR archetypes is summarized in Figure 17.

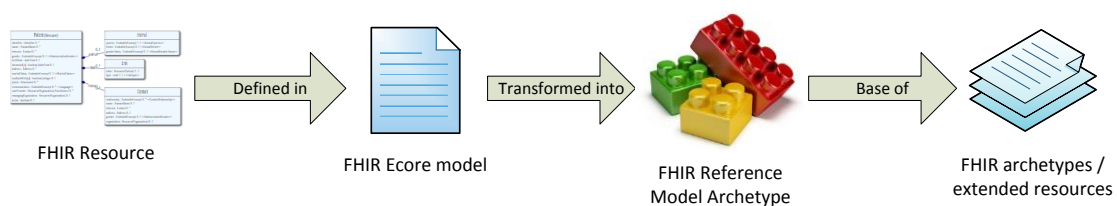


FIGURE 17 STEPS FOR THE CREATION OF FHIR ARCHETYPES

For the creation of the RMAs, first we chose the archetypable entities. We selected all clinical, all administrative, and an infrastructure (namely Composition) resources. After selecting these entities, we parsed the XMI to obtain all types, their attributes, and the types of these attributes. From this analysis, a set of all the types in the model represented as small RMA are created. These RMA contain the corresponding attributes defined in the ecore model and

reference each one of the types inside them. This reference is made by using the archetype slot mechanism to point to the RMA where the type is defined. In this step, inheritance is solved by deriving any type into all their child types. If a type can be derived into more than one type then an alternative of archetype slots is created for each one of the subtypes. In order to reduce the size of the resulting RMA, only one type is referenced explicitly (i.e. one slot exists) in each RMA, and any further use of the same type in the archetype is transformed into internal references to the first appearance of this type to avoid the repetition of archetype structures. With all this set, the iterative process can begin. In each one of the iterations, every archetype slot reference from the RMA is expanded with the corresponding archetype. Internal references are adjusted so all point to the same archetype slot, as expanded slots can include more slots to other RMA. Only one archetype slot is kept, and the other ones are changed into internal references. The iterative process ends when the RM archetypes selected as archetytable entities (i.e. the types we want to be able to generate archetypes from) do not contain any external references (i.e. archetype slots) to other RMA, as a restriction of RMA is that they must be completely defined on their own.

Once we have created a set of RMA from a given RM we can create archetypes from that RM with LinkEHR Editor. Figure 18 shows an excerpt of a FHIR Systolic archetype.

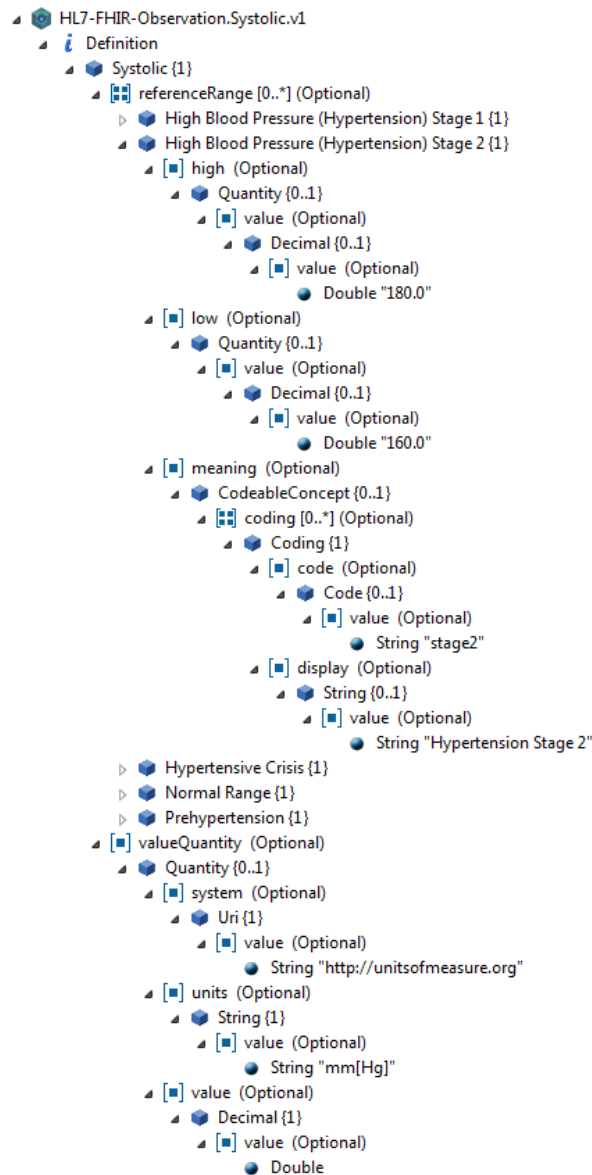


FIGURE 18 EXCERPT OF A SYSTOLIC FHIR ARCHETYPE

On the other hand, there also exist meta-model representations used only in dual model reference model definitions. This is the case of openEHR BMM syntax (123), which has available representations of openEHR (121), ISO13606 (12) and CIMI (19) reference models. Although openEHR RM is also distributed in XSD format, BMM is considered as the primary reference model definition, as XML schemas may have their own set of problems when representing an object-oriented model. BMM is based in ODIN syntax (126) (formerly known as dADL).

An ad-hoc Java parser was created in order to generate the RMAs from a BMM reference model specification. BMM format has some advantages over XMI, as the set of archetytable classes is explicitly included in the meta-model definition and the translation of the structure to archetypes is almost direct, but also needs additional documentation (such as knowing which attribute contains the archetype node identifier) in order to correctly generate systems and validators based on a BMM file. This specific parser was included in LinkEHR to generate the RMA automatically from the BMM definitions.

2.3.2. AOM-based RMA creation

An alternative to create RMA when the meta-model definitions are not available or they are available in a format not processable by our methodology is to create the RMA manually.

In the case of MedXML MML, RMA were created by hand from MML specifications (127). A mode to edit archetypes without having available an underlying RM was developed, i.e. an editor to create archetypes based on Archetype Object Model (AOM). This is based on the principle that reference models define the basic constraints that describe a given information model, which is precisely what in the end provides the AOM (i.e. the AOM provides the basic blocks to define constraints over a given model). As RMA are in fact archetypes it is feasible to build these archetypes by defining the object, attribute, and primitive type constraints they contain. The usefulness of the creation of this kind of editors over simple tree models has been proved in conversion systems like YAT (128). This process is possible due to the tool not being based in any given reference model but in the AOM itself

To edit an archetype, first an object name must be provided. This object name is used in the archetype identifier. This also creates the root object of the archetype. From there, it is possible to constraint attributes on objects, as seen in Figure 19, and objects, data types, internal references, and archetype slots in attributes, as seen in Figure 20. The only parameter needed for the creation of attributes and objects is the attribute name and object name respectively. Once the attribute, object, or data type is created it can be edited as a normal constraint of its kind in order to modify occurrences, ranges, and allowed values.

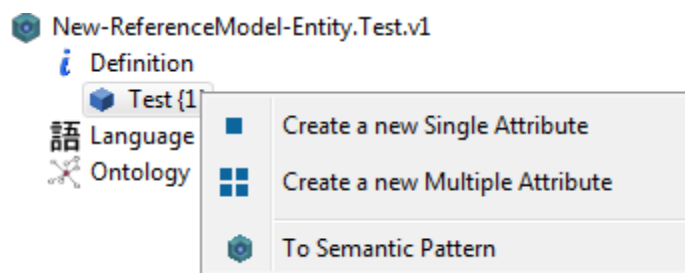


FIGURE 19 ARCHETYPE EDITING WITHOUT REFERENCE MODEL – CREATING ATTRIBUTES

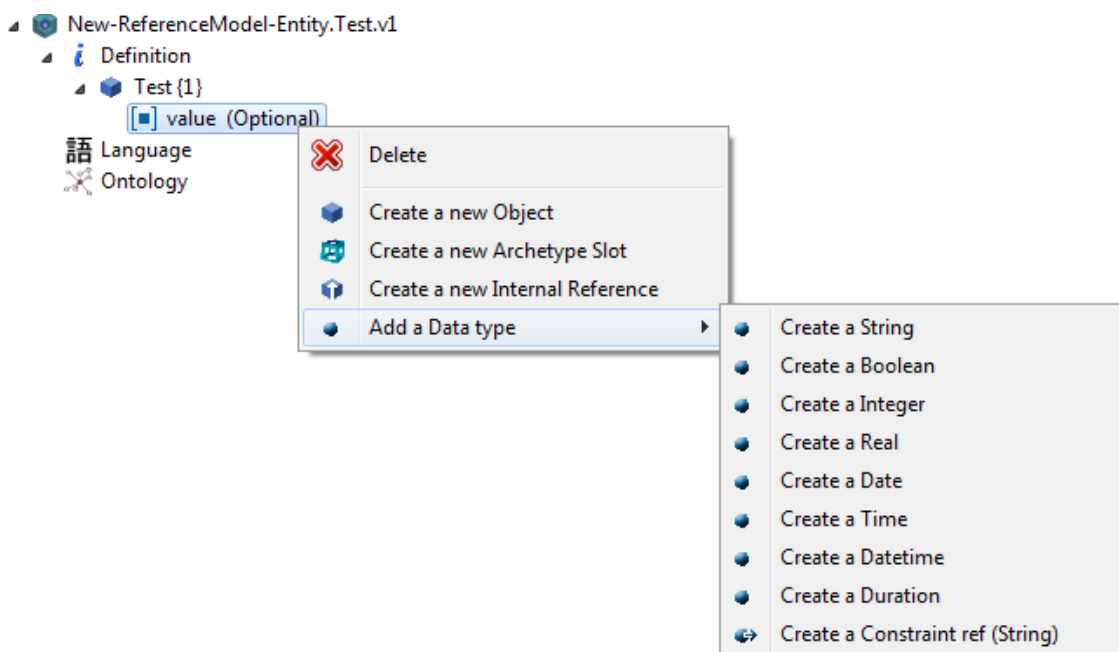


FIGURE 20 ARCHETYPE EDITING WITHOUT REFERENCE MODEL – CREATING OBJECTS

With this new editor, a RMA for each business class can be created. In the case of MedXML MML, RMAs were created for MML common formats and content modules. All the requirements and structures from the MML documentation were able to be expressed with archetypes. Archetypes were created using all the reutilization mechanisms that archetype model provides (namely archetype slots and internal references). 9 common module archetypes were created and 13 archetypes were created from the 12 clinical modules defined in MML documentation. The additional one (mmISm:Clinicalcourse) is a concept defined in both mmSm:SummaryModule and mmIRe:ReferralModule with the same structure, and thus it was extracted as a new RMA. The list of created archetypes can be seen in Table 1 and Table 2.

MML common format	Created archetypes
mmIAd:Address	MedXML-MML-Address.Address.v1
mmIph:Phone	MedXML-MML-Phone.Phone.v1
mmICm:Id	MedXML-MML-Id.Id.v1

mmlCm:Ref	MedXML-MML-Address.Address.v1
mmlNm:Name	MedXML-MML-Name.Name.v1
mmlFc:Facility	MedXML-MML-Facility.Facility.v1
mmlDp:Department	MedXML-MML-Department.Department.v1
mmlPsi:PersonalizedInfo	MedXML-MML-PersonalizedInfo.PersonalizedInfo.v1
mmlCi:CreatorInfo	MedXML-MML-CreatorInfo.CreatorInfo.v1

TABLE 1 ARCHETYPES FROM MML COMMON FORMAT

MML module concepts	Created archetypes
mmlPi:PatientModule	MedXML-MML-PatientModule.PatientModule.v1
mmlHi:HealthInsuranceModule	MedXML-MML-HealthInsuranceModule.HealthInsuranceModule.v1
mmlRd:RegisteredDiagnosisModule	MedXML-MML-RegisteredDiagnosisModule.RegisteredDiagnosisModule.v1
mmlLs:LifestyleModule	MedXML-MML-HealthInsuranceModule.HealthInsuranceModule.v1
mmlBc:BaseClinicModule	MedXML-MML-BasicClinicModule. BasicClinicModule.v1
mmlFc1:FirstClinicModule	MedXML-MML-FirstClinicModule.FirstClinicModule.v1
mmlPc:ProgressCourseModule	MedXML-MML-ProgressCourseModule.ProgressCourseModule.v1
mmlSg:SurgeryModule	MedXML-MML-SurgeryModule.SurgeryModule.v1
mmlSm:SummaryModule	MedXML-MML-SummaryModule.SummaryModule.v1
mmlLb:TestModule	MedXML-MML-TestModule.TestModule.v1
mmlRp:ReportModule	MedXML-MML-ReportModule.ReportModule.v1
mmlRe:ReferralModule	MedXML-MML-ReferralModule.ReferralModule.v1
-	MedXML-MML-ClinicalCourse.ClinicalCourse.v1

TABLE 2 ARCHETYPES FROM MML MODULE CONCEPTS

These archetypes faithfully represent the structure and contents of MML modules. Some parts of MML are influenced by HL7 CDA(129), which introduces XML-only constraints such as mixed elements (elements with value that also contain attributes) and heavy use of namespaces. Support for HL7 CDA particularities was included in the past into LinkEHR editor to support HL7 CDA archetype definition and thus supporting them for MML required no further additions.

Figure 21 shows an example of a created RMA for a MedXML MML module.

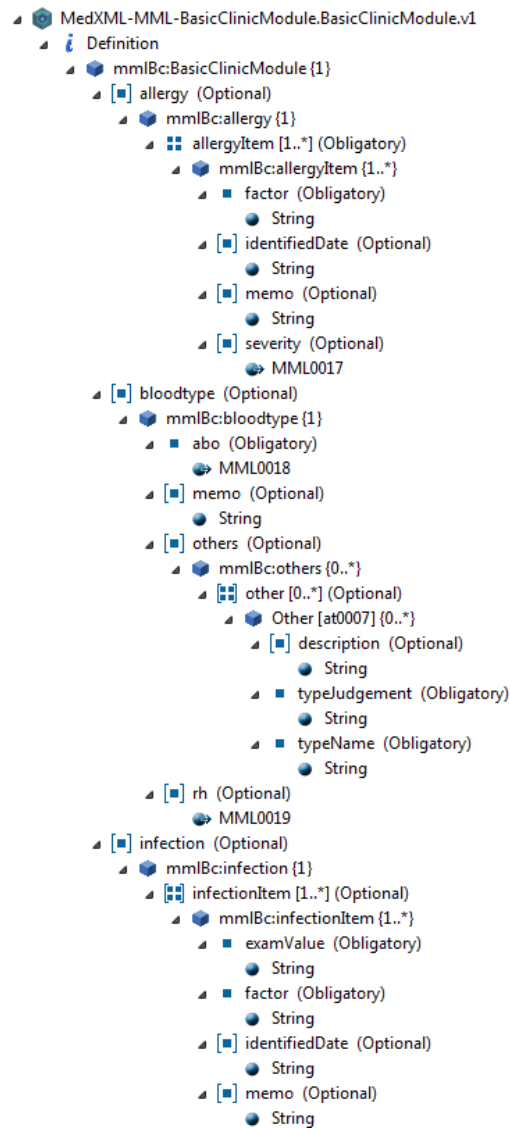


FIGURE 21 EXAMPLE OF A MML BASICCLINICMODULE REFERENCE MODEL ARCHETYPE

Although creating an automatic process for the generation of RMA could be more time consuming than just defining a set of RMAs, if the format for the definition of the schema is shared by a set of models, it is preferable to create an automatic process.

2.4. Advanced archetype editing

Specific archetype editors for a particular reference model such as openEHR archetype editor can be regarded as “concept centric” since they hide most of the complexity of the underlying reference model. Therefore, they are suitable to be used by health domain specialists even with moderate knowledge of the underlying reference model and archetype model. The main drawback of this approach is that the reference model must be hard coded into the editor making it difficult both to keep in pace with its evolution and to support multiple models.

The “raw” use of RMA brings to the front the reference model during the editing process. This makes the editor “structure-centric”, i.e. domain concepts are defined by directly constraining the data structures present in the reference model according to the archetype formalism. Obviously, this approach forces users to have a deeper knowledge of the reference model, but facilitates working with multiple models. Figure 22 shows this “structure-centric” edition in a FHIR Adverse Reaction archetype. In order to make the editor, in our case the LinkEHR editor, more accessible and aligned with users’ knowledge of current standards, different approaches and methodologies have been investigated, namely plug-ins, mapping to other standards, semantic patterns, archetype creation from sample instances, and syntactic clinical model transformation between standards. They are discussed next.

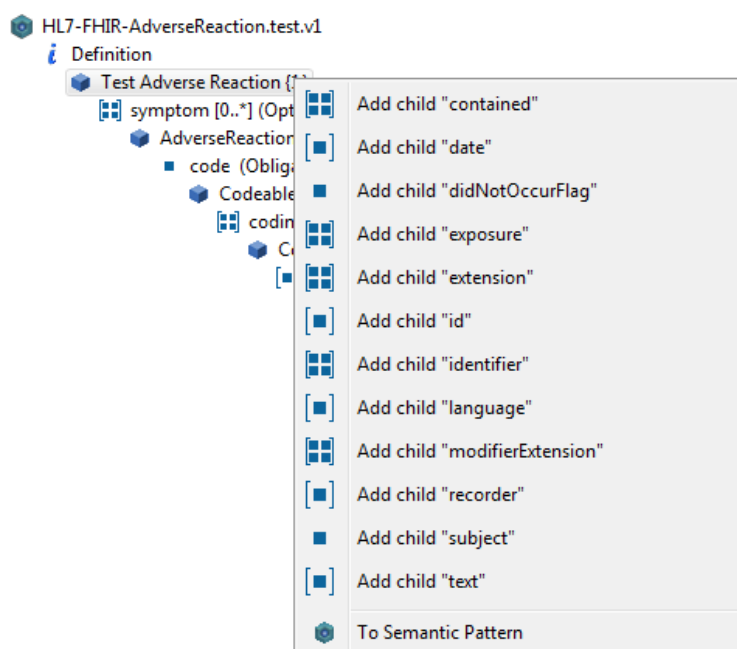


FIGURE 22 EDITING A FHIR ADVERSE REACTION ARCHETYPE

2.4.1. Plugin archetype editors

LinkEHR editor allows the creation of archetypes for a given reference model. As a basis LinkEHR Editor assumes that user has knowledge over the full reference model. In order to allow users with less specific expertise and knowledge of the underlying reference model to edit archetypes, we developed specific editors that use knowledge about a given reference model to hide archetype editing complexity. We have developed this kind of editors for ISO13606, openEHR, and HL7 CDA (130). Figure 23 shows a HL7 CDA archetype being edited with a custom editor.

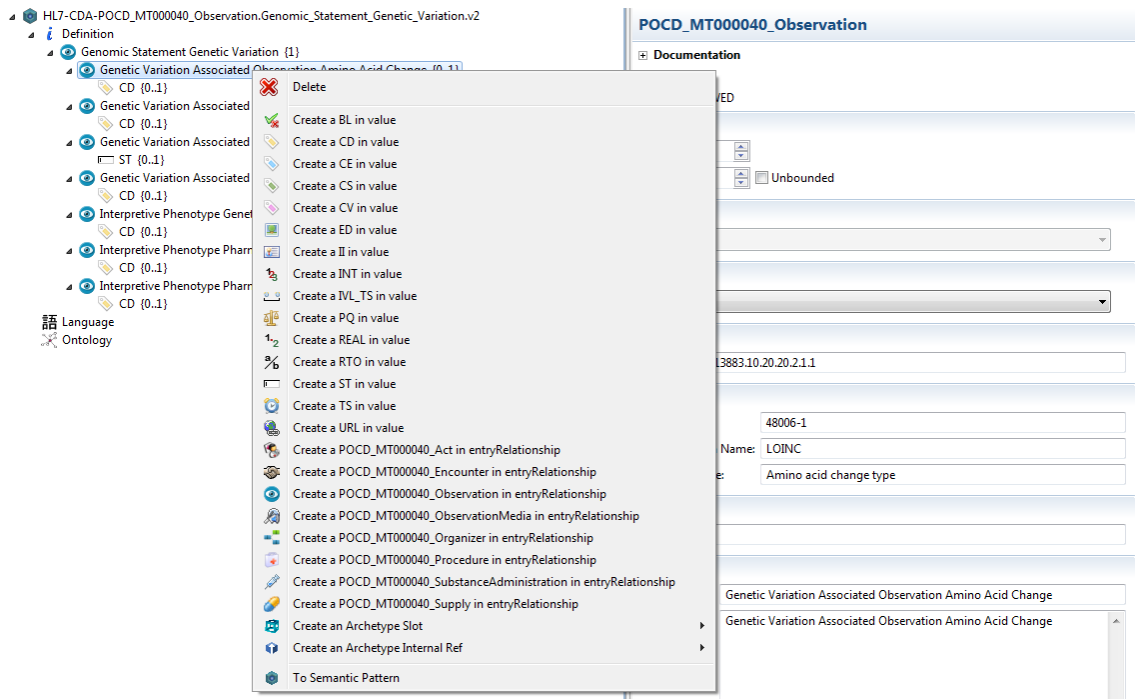


FIGURE 23 EDITING AN HL7 CDA ARCHETYPE WITH SPECIFIC EDITOR

Custom editors are created on the fly following a documentation file. This documentation file contains the complete details specific to a given reference model. These details include technical details (e.g. which is the attribute that contains the node identifier in a given standard: `archetype_id` in ISO13606, `archetype_node_id` in openEHR, or even `templateId` in HL7 CDA), properties related with class diagrams (e.g. which attributes contains a given class, which types are allowed in given attributes), multilingual description of classes, UI related properties (e.g. which icon should be used for this class or which transformation should be used when showing the user a sample form), and editor related properties (e.g. which class should be used to edit a given type or if an attribute is interesting enough to be shown in a specific editor). The class diagram related properties and technical details are generated when the model is imported. By default, the specific editor tree always hides the attributes, being the HL7 types the basic blocks in the edition process. E.g. A user that creates a Composition can include Sections or Entries, without worrying or knowing which attribute needs to navigate in order to include them. If an attribute is found interesting, it can be selected as navigation attribute. Navigation attribute child types will be the alternatives shown in the editor contextual menu. Reference models can contain long navigation trees that traverse classes that do not need to be constrained. For this use case, the ability to hide classes from the archetype tree edition was added. An example of this can be seen in Figure 23, as HL7 CDA Observations are related to other HL7 CDA Observations by an `EntryRelationship` class, which is

not shown in the specific editor but is stored in the archetype. This process can also hide from the edition process full archetype branches, even if they are still being created, e.g. when creating a Physical Quantity (PQ), both the units and the value are created, but only the PQ will be shown on the editor tree. These specific editors can be distributed as plugins and be included into the editor without changing the application source code. This documentation file is automatically generated on RM import, but can be updated in the documentation manager called Reference Model Manager. Figure 24 shows the reference model documentation being edited in the Reference Model Manager.

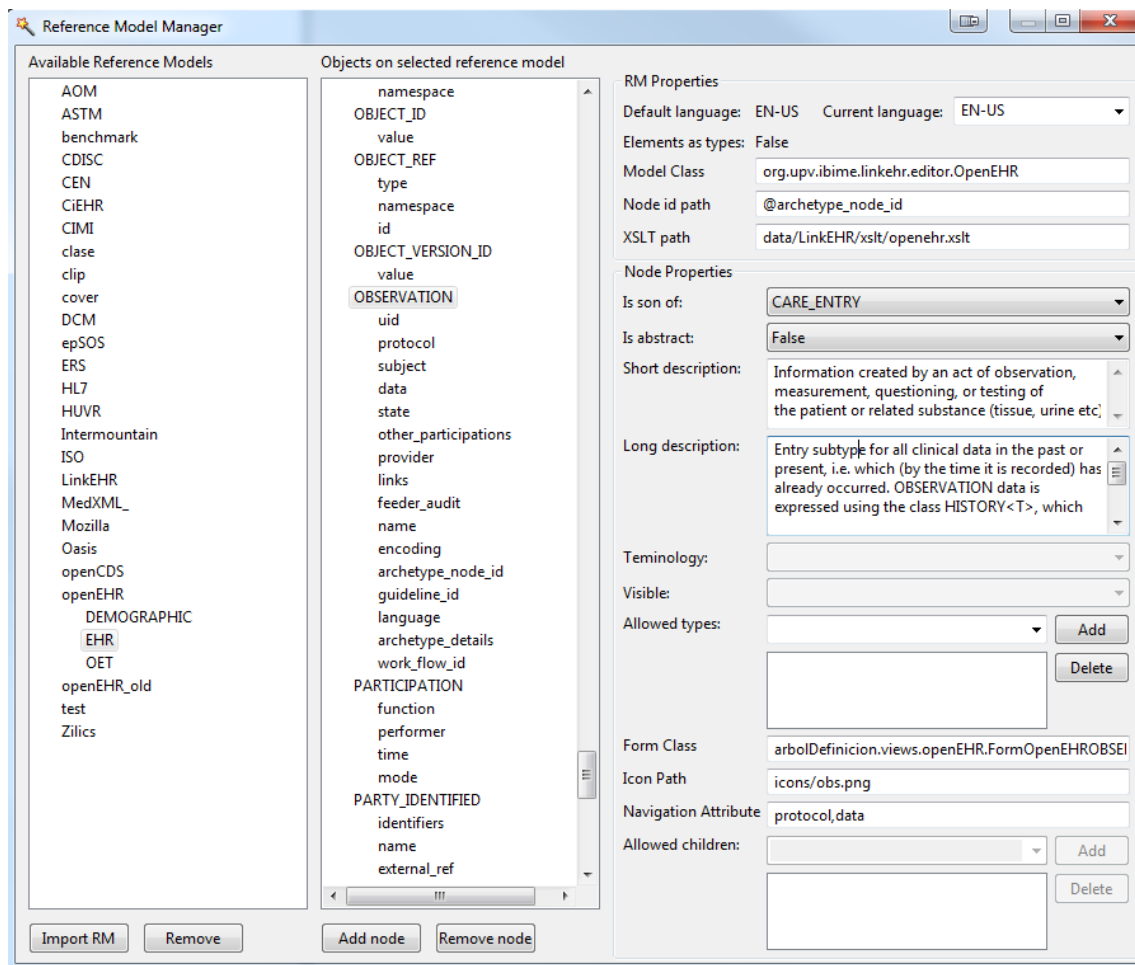


FIGURE 24 REFERENCE MODEL MANAGER INSIDE LINKEHR EDITOR

2.4.2. Mapping to non-dual models archetypes

One of the advantages of including a new reference model into LinkEHR is to generate transformation programs from the archetypes. As described in (131), the mapping process is as follows: First, an archetype with the use case specific constraints must be created. Second, LinkEHR merges the archetype with the underlying RMA in order to assure that the constraints from the archetype and the reference model will be in the final instance. Third, transformation

functions can be assigned to the atomic values from this merged archetype in order to get the correct values. These mappings can be either from another archetype (from any RM) or in form of XML Paths. Fourth, when the defined mappings are enough to generate at least the mandatory information in a given RM, LinkEHR can generate a transformation program from it. This transformation program is an XQuery program that transforms source data into an XML instance compliant with the merged archetype (and thus compliant with both the defined archetype and the underlying RM). Source and target archetypes can be from dual model standards (such as ISO13606 or openEHR) or non-dual model standards (e.g. HL7 CDA, HL7 FHIR, MedXML MML, etc.). Mapping process is explained in detail in chapter 3.

2.4.3. Semantic patterns

Archetype reuse is not limited to archetype slots and internal references. More advanced reuse patterns can be defined for the reuse of complex structures with clear meaning. SemanticHealthNet project (132) proposed the use of a kind of structure called semantic patterns. Semantic patterns are reusable solutions to recurring modeling problems based on an ontological framework in order to bridge between the EHR modeling community and the semantic and formal ontology communities. Their purpose is to guide and standardize the representation of the information meaning encoded by clinical models. For SemanticHealthNet, semantic pattern is a concept that combines structural, terminological, and ontological representations to enable multiple clinical models to be recognized as overlapping and primarily aligned (even between different standards). SemanticHealthNet proposed an OWL representation of semantic patterns. Other interoperability projects such as CIMI have also analyzed, identified, and suggested the use of semantic patterns (also called modelling patterns in CIMI) as part of their modeling approach (133).

Based on this notion, we propose one way of implementing the semantic patterns to help and guide archetype creation. The use of these patterns will help in the achievement of SemanticHealthNet's original purpose, as the structural fragments we define can be translated and used in other standards. In our view, semantic patterns are archetype fragments with known semantics (i.e. usually bind to clinical terminologies to express its meaning). They are designed to be reusable and thus have general meaning, as opposed to slots which fully define a given clinical model on their own. I.e. patterns are expected to be modified upon inclusion on the archetype to better represent the current clinical model use case. The kind of semantic patterns we define are useful for creating equivalences between different reference models, e.g. how an openEHR Observation is represented in ISO13606 reference model. Semantic

patterns provide a way for including predefined meaningful structures, as the semantic pattern themselves provide semantic bindings for a given archetype node. Examples of semantic patterns include general use structures such as table, tree, or panels, semantic structures such as observation, event, or history, and complex semantic patterns such as exam.

For the creation, reuse, and version of semantic patterns LinkEHR editor allows selecting any node of a given archetype and transforming it to a semantic pattern. These patterns are stored as archetypes and thus contain both metadata and vocabulary sections that are used for the correct description of the pattern. This also enables the pattern versioning mechanism. Semantic patterns can be edited as normal as they were normal archetypes. For the reuse of semantic patterns, the process allows to include them in any place the reference model allows that given type, in the same way as normal archetype creation does. When a semantic pattern is selected, it is included in current archetype. A terminology binding is added in order to know exactly which patterns were used when creating a given archetype. An example of the edition is shown in Figure 25. In this figure, two ENTRY semantic patterns (Observation or SubjectOfCare) can be included inside of a COMPOSITION EN13606 class.

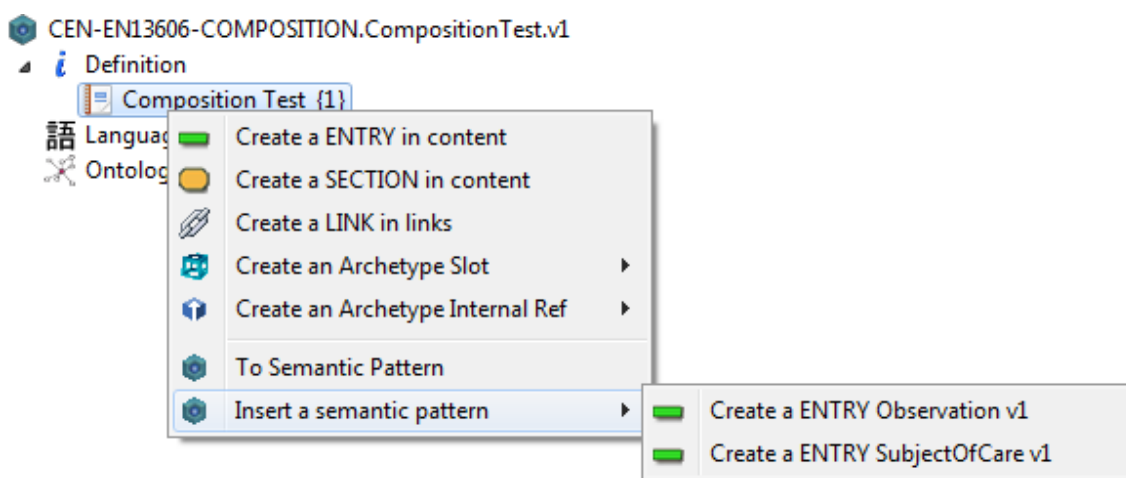


FIGURE 25 INCLUDING A SEMANTIC PATTERN IN CURRENT ARCHETYPE

2.4.4. Archetype creation from instances

Probably one of the biggest challenges when applying archetypes to a non-dual reference model is that defining a minimum set of archetypes is required in order to start taking advantage of archetype-based methodologies and tools. This archetype creation task can be time consuming and requires a deep knowledge of the system. However, as usually the system is already deployed, it is feasible to have access to sample data instances. We developed an

automatic process that traverses a set of data instances on a given reference model and merges them into an archetype with the constraints detected on data. The process is based on the same principle as the generation of XML instances, i.e. an XML instance is equivalent to an archetype with the constraints fixed to constants (we call this constant archetypes, see chapter 4 for a full explanation of constant archetypes). If each XML instance can be seen as a constant archetype, then the problem is reduced to merging the different constant archetypes into a single one. This process of merging two constant archetypes traverses the constraints in each archetype and loosens them according to their constraints values (e.g. widens ranges or makes data types less restrictive).

Archetypes allow the definition of three kinds of constraints: object constraints (namely occurrences), attribute constraints (existence and cardinality) and primitive constraints (which depend on the data type). Occurrences, existence, and cardinality constraints can be guessed by measuring what parts of the instance appear/disappear or repeat. For the first appearance of an XML element or attribute, a mandatory attribute is created on the archetype. Each successive instance is checked to see if a created attribute still exists on it. If the attribute exists just once, nothing is done. If the attribute path exists more than once, the attribute is converted into a multiple attribute (container attribute) and is given a cardinality according to the number of times it appears in the instance. If the attribute does not exist in the instance, then the attribute is made optional on the archetype. The process is similar for objects. Archetype objects are created by looking at the `xsi:type` attributes, which are XML attributes with special meaning containing the object class of a given attribute. If the underlying reference model is known, archetype node identifier is used instead of the `xsi:type` (e.g. `archetype_node_id` attribute in openEHR). For any new type (or node identifier) detected, a new object with mandatory occurrences is created. Occurrences constraint will be relaxed with new instances accordingly.

Getting meaningful results from unknown reference models is harder, as `xsi:type` attributes are optional and do not need to be included in the data instances (and even if they do exist, sibling nodes sharing the same type are impossible to distinguish). However, if the reference model is available, not having an `xsi:type` can be worked out as missing types usually mean that the attribute only has one allowed type, and thus can be automatically selected. It is also worth noticing that with our methodology is impossible to guess 'not allowed' attributes (existence 0..0) or prohibited objects (occurrences 0..0), as they will never appear in data instances.

Primitive constraints provide an interesting challenge. In this case, not only the value constraint may be relaxed with every new instance, but also the type of the value constraint can be modified into a more generic data type. When a new value is detected, it is checked against patterns created to detect each type. Data types with very specific patterns such as `DateTime` or `Duration` are checked first. Every other pattern is checked in descending complexity order, and if nothing matches then the data type is considered to be of `String` data type. In case of strings, two different string values create list constraints (e.g. if first we found the value 'low' and the next instance contains the value 'high', a string constraint with the list {'low','high'} is created. A similar process is used for the different data types.

As an example, if the first data instance had a value of "123.4" then it is interpreted as a real value and a constraint to that exact value will be created on the archetype. If the next analyzed data instance has another real value such as "543.2", then the created real constraint is modified to define a range from [123.4..543.2]. If the next instance includes another range it will again modify the range accordingly. If another instance has any value that is incompatible with the current guessed constraint type, then the type is changed by a more general data type. E.g. a detected integer constraint can be modified into a real constraint (assuming that it was a real with no decimal part) or a string constraint if the received value is from an incompatible type.

As stated above, this instance creation process is more efficient if the information model has been previously imported as a RM, as data types and existence constraints can be also be obtained from the RM.

Once every instance has been merged, the generated archetype is offered to the user in order to fine adjust the value, occurrences, existence, and cardinality constraints.

2.4.5. Syntactic clinical model transformation between standards

Transformation of clinical models between different standards is a difficult problem. One of the most promising approaches is the ontology-based transformation of archetypes and data instances between different reference models. This kind of transformation is already present in the literature (58). However, most of the time class equivalences are clear enough or are even part of the standards themselves (such as ISO13606 (32) or HL7 FHIR(15)) and thus transformation of the archetype to ontology languages for reasoning is not needed. A completely syntactic transformation is feasible as the translation of clinical models from one reference model to another can be done with a number of rules in the same order of

magnitude as the maximum number of classes of the reference model with more classes. For this syntactic clinical model transformation, rules to transform the class name and attributes of each class are created. Generated rules allow both the creation and removal of attributes and classes that have no transformation, e.g. an openEHR ITEM_LIST is transformed into a ISO13606 CLUSTER and adds both 'meaning' and 'structure_type' attributes (in an ISO13606 to openEHR transformation these attributes and classes are removed). In addition to that, rules for the transformation of the archetype metadata are also created.

Rules for class transformation can be categorized in three different kinds depending on what triggers them: rules based on the structure of the source archetype, rules based on the terminology binding of the class, and rules for the generic transformation of the type.

- Structure-based rule: this kind of rule uses conditions over the structure of the source archetype to choose the corresponding class. These conditions cover at least one class type, but can be made dependent of any number of other classes and values present in the source archetype. E.g. to translate an ISO13606 ENTRY to openEHR Observation it is necessary to create rules that search for an ENTRY object which 'meaning' attribute has 'OE-01' code. This is the rule with highest priority from the three.
- Terminology binding-based rule: As archetype nodes support terminology bindings for their complete description, we can use them in order to categorize the class. This however requires that the semantics of target classes are fully described in a given terminology (typically SNOMED-CT). This is the rule with the second highest priority from the three.
- Generic transformation: Health information standards, such as ISO13606, openEHR, or even HL7 CDA, contain a generic class intended to accommodate data coming from a different standard. This last rule, which has the lowest priority, ensures that every class translates to at least to a generic class.

Any given set of source classes can be translated into any set of classes in the target (using insertion, deletion, and substitution operations).

We demonstrated the usefulness of this approach by studying the ISO13606 to openEHR model and data automatic bidirectional transformation. On the one hand, as the ISO13606 part 3 standard already defines the class equivalences and what codes should be put, the openEHR to ISO13606 was easier to develop. As the equivalences are already known, only structural rules were created for openEHR to ISO13606 transformation, which makes the

process completely deterministic. We generated a set of 57 rules for the translation of openEHR archetypes to ISO13606 (46 rules for class translation and 11 for the translation of archetype metadata). These rules transform types, attribute names, and also add the corresponding 'meaning' attribute. This attribute is available in all record component classes in ISO13606 and can be used in this context to provide a code describing the transformation source. We applied these rules to a dump of all available openEHR archetypes in the international CKM. This generated a set of 415 ISO13606 archetypes¹. These archetypes were validated against the ISO13606 reference model and all were compliant with it. For a deeper explanation of the validation of archetypes see (134). The creation of the openEHR to ISO13606 transformation is straightforward as ISO13606 model is more generic than openEHR and usually a set of classes is transformed into a single class (e.g. in openEHR the Observation, Instruction, Action, Evaluation, Admin_entry, and Generic_entry are transformed into the ISO13606 Entry class). However, the inverse transformation (i.e. ISO13606 to openEHR) poses a challenge, as a single class in the origin can be transformed into a given class from a set of classes.

For the translation of ISO13606 classes into openEHR classes, a structural and a terminology binding rule were defined for each target class the source class could be translated to. An additional generic rule was created for each source class, to allow the class to be translated to a class from the target reference model even if no other rule is launched.

In this case, for the structural rules, the 'meaning' attribute (which as stated above may contain a code from ISO13606 part 3 terminology) was used. For the terminology binding rules, we used SNOMED-CT expression constraint language (135) to provide tentative descriptions of the openEHR entry classes. The expression is designed to return the tested code itself if the binding is included in the expression, and to return nothing if it is not included on it. Codes used in these expressions were extracted from current CKM archetypes terminology bindings. As few archetypes contain bindings and there is not yet a unified way of defining these bindings, created expressions may be incomplete. Table 3 shows the expressions used to test if the terminology binding can be identified as a target openEHR class. In these expressions, the unary operator “<<” stands for “descendants of the specified concept plus the specified concept itself”. These expressions are evaluated by using web services provided by the Snomed CT expression constraint execution engine SNQuery (136).

¹ Transformed archetypes can be downloaded from <http://tiny.cc/ISO13606archetypes>

Target class	SNOMED CT subset
Observation	<< 363787002 Observable entity (observable entity) OR << 284365007 Examination of body site (procedure) OR << 122869004 Measurement procedure (procedure)
Evaluation	<< 243814003 Interpretation of findings (observable entity)
Instruction	<< 243120004 Regimes and therapies (regime/therapy) OR << 400999005 Procedure requested (situation)
Action	<< 129264002 Action (qualifier value) OR << 416118004 Administration (procedure) OR << 443938003 Procedure carried out on subject (situation) OR << 71388002 Procedure (procedure)
Admin_entry	<< 14734007 Administrative procedure (procedure) OR << 304784009 Administrative form (record artifact)
Event	<< 272379006 Event (event)

TABLE 3 SNOMED CT GRAMMAR EXPRESSION FOR CATEGORIZATION OF OPENEHR CLASSES

The ISO13606 to openEHR translation direction was limitedly tested, as currently available archetypes usually do not have associated 'meaning' or terminology binding, which in the end causes generated archetypes to be transformed using the generic transformation.

In addition to guide the model transformation, rules also generate the attribute mapping equivalence between the original archetype and the transformed one. This allows generating automatically a transformation program to convert data instances based in a reference model to another reference model (see chapter 3).

2.5. Improvements in LinkEHR Editor

The above methodologies have been incorporated into LinkEHR editor. The definition of plugin editors provides a high level of customization to users and allows covering exactly their needs. These plugins contain the reference model, their documentation, and the Java compiled UI classes to specifically edit given reference model types. For the mapping of archetypes to data sources, a new perspective was added in order to import data sources, edit mappings, and generate transformation programs from within the tool. For the use of semantic patterns, the

options to include and generate semantic patterns were added, as well as a semantic pattern manager to manage them from inside the tool. For the generation of archetypes from instances a wizard was added that uses both a set of provided instances and the name of the reference model and reference model entity for creating the archetype. For the syntactic clinical model transformation between standards an export option was added to provide the translation of current archetype. This export option changes depending on current archetype reference model and gives access to an output transformation (if defined for the selected reference model).

In addition to these changes, other improvements were included in the tool in order to deal with new reference models and being able to use the tool into different standards workflow.

2.5.1. Connection to external repositories

Knowledge sharing is one of the key principles of archetypes. Several countries (such as Norway, Brazil, or Spain) have deployed archetype repositories to store and publish their clinical models. These archetypes establish national requirements for the semantic interoperability of data transferred within a country. In order to reuse these clinical models, an access to the repositories was provided. Both openEHR CKM and LinkEHR CM provide a set of web services (SOAP or REST) to query and retrieve archetypes available in them. LinkEHR Editor was improved to access both kinds of web services.

2.5.2. Template import

In dual model development cycle, templates provide ways to specialize an archetype for a local use. They define local use constraints and complete the archetype slots in the original archetype with their actual representation. In early openEHR days, these templates came in a format that is deprecated in ADL 1.5. In order to provide a way of reusing these templates in modern tools, a process to import templates and transform them into archetypes was created. This process analyzes the template (in either OET or OPT format) and provides an equivalent archetype from them. Templates in legacy format omit part of the structure of the clinical model, as they are assumed to be based on openEHR RM. This makes a requirement that openEHR RM is imported in the tool, as omitted parts are inferred from the remaining structure. This method can also connect to remote or local archetype repositories in order to fulfill all the archetypes referenced by the template. The resulting archetype can be used for mapping and reference material generation.

2.5.3. Export JSON schema

As JSON (137) implementations become more common, there is also the need to validate data and structure constraints in JSON. To allow this, a new method for the creation of JSON Schema (138) from the archetype was implemented. This allows the validation of JSON instances based on a given archetype in the same way a XML Schema or Schematron validate XML. By generating JSON Schema, other derived materials can be automatically created such as Java classes, API documentation, data sources schemas, forms, fake data instances, or editors with autocomplete.

2.5.4. Export FHIR profiles

FHIR uses a mechanism similar to the archetypes and specializations that is called 'profile'. In order to allow the created archetypes to be used in FHIR servers and clients a method to export FHIR archetypes to FHIR DSTU profiles was created. Archetype constraints are translated into profile constraints. This process also allows the selection of which paths of the archetype can be queried, providing a name for this parameter, the type, and path. With this information, these query parameters are translated into profile query parameters and the profile is generated.

2.6. Conclusions

Using archetypes for non-dual model based standards allows the use of all available methodologies and tools, from archetype editors to repositories (both data and archetype repositories). This can be really beneficial for all involved parties, as it allows the use of high quality clinical information models in current data workflows and EHR systems. It also allows the improvement of clinical models with new use cases and expertise by different people, which in the end greatly enriches the original archetypes.

There are several advantages of using dual model methodology with non-dual model standards, such as being able to check if a resource is valid against the reference model. This is especially useful not only with draft standards such as FHIR or standards with few dedicated tools such as MML, but also in the renewal of reference models like ISO13606, openEHR, or HL7 CDA. The changes on the reference model could potentially invalidate the already developed archetypes for the past reference model version. Our methodology allows regenerating the RM when new versions are released and use established mechanisms for archetypes like subsumption (134) to check if available archetypes are still valid with the evolved model, i.e. to check for the consistency of the archetypes and reference models.

The involvement of clinicians in systems creation is another of the clear advantages of using clinical models. Most of the problems with health information systems evolution and use can be tracked down to systems that were designed with clinical knowledge embedded into the system underlying information model (e.g. deciding what parts are recorded in a given clinical process). The use of clinical models allows for the separation of clinical knowledge from the structure of the system. Using archetypes eases the communication between clinicians and technical staff. They also allow for a better requirements capture, as clinicians themselves put their requirements in the archetypes, avoiding miscommunication problems.

One of the advantages of archetypes is that they provide a way of linking clinical models with clinical terminologies and vocabularies. This allows the clinical models to be clearly and unambiguously described. Archetyped data can also be easily included in the system, as the meaning and semantics are well known.

Multilinguality is one of key advantages of using archetypes on these standards. In fact, the advantages are clear for each standard we have studied. In the case of FHIR, there is no explicit support for multi-language representation in resources (139). FHIR-based archetypes can be used to translate FHIR Resources to other languages without the need of creating profiles to extend the resources. This is similar to the use case of MedXML MML. In MML, the need of translations was already pointed out, as a modification of the original schemas was proposed to give support to Chinese language (71). This arguably could be done by translating our developed MML archetypes. When dealing with HL7 CDA, an implementation guide is typically released for each language, which is harder to maintain and use. Having multilingual source archetypes would allow the generation of reference materials such as implementation guides for a given language automatically. Archetype terminology bindings also provide their own mechanism to support multilinguality by looking up translations in a terminology server.

Another key advantages of using archetypes are the knowledge reuse, the ability to lock down modelling optionality and vocabularies, the ability to generate derived reference materials from the archetypes (such as Schematron, sample instances, sample formularies, and implementation guides), the use of AQL(39) for data query, and even the transformation to OWL to run SPARQL queries (61,140).

In particular, defining archetypes for non-dual model standards also allows us the mapping of existing systems and standards from and to dual model standards like ISO13606 or openEHR. This allows us to seamless include existing archetype systems into the workflow of current

standards and vice versa. Archetypes can also be mapped from legacy data sources in order to generate valid data for current systems. This methodology has been tested for other non-dual model standards such as HL7 CDA (141) and CDISC ODM (142).

In addition to that, dual model approach allows the use of modelling methodologies for the creation of clinical models in any given standard. These modelling methodologies are independent of the chosen standard and put the reuse of validated clinical models as a key phase of the process (46).

There are also a few disadvantages when using archetypes with non-dual models. In general, narrative parts (as the ones available in HL7 CDA or HL7 FHIR) are difficult to handle in model transformation (143). This is still true in the case of archetypes. In addition to that, each standard usually has already defined a workflow which would need to be adjusted to also use archetypes, which is not always possible.

One of the biggest drawbacks of using archetypes in non-dual model standards is that some models have business concepts with low variability and thus using archetypes on them has very little added value (e.g. ASTM CCR model, some HL7 FHIR resources, or even ISO13606 current demographic model). Even in this case, using archetypes provide added value such as formally supporting multilinguality, knowledge reuse, and terminology bindings.

The advantages of a joint use of archetypes with non-dual model standards outweigh the disadvantages. Creating archetypes for these standards allows us to reuse all the tools and methodologies developed for dual model standards, and using archetypes for currently used reference models such as HL7 CDA, HL7 FHIR, or MedXML MML will help to the rapid adoption of both the original standard and the dual model approach.

Chapter 3.

Migration of Health Information Systems – Data perspective

3.1. Introduction

Health care is a sector where the need of sharing information is the norm rather than the exception. However, the health data of one patient is usually scattered among the different health facilities where they have been attended. As a consequence, it becomes increasingly important to combine and communicate seamlessly all the distributed information with minimal additional support or intervention from end users. Due to the special sensitivity of health data and the wide range of ethical and legal constraints, health data communication must be done in a meaningful way, avoiding all possibility of misunderstanding or misinterpretation. This crucially depends on the standardization of the EHR architecture.

This chapter deals with one of the main problems when adopting EHR-related standards: how to standardize existing data. In our scenario, this involves transforming EHR content into data structure compliant with reference models and archetypes. We face a problem known in the literature as the data exchange (translation or transformation) problem (4). This problem is a difficult one, since it deals with differences and mismatches between heterogeneous data formats and models. In the EHR scenario this problem is even more complex. On one side, we have the legacy data that conform to a particular schema and with local semantics. On the other side, we have EHR architectures and archetypes that have been defined without any consideration regarding the internal architecture or database design of EHR systems. Our objective is to create an instance of the target schema (archetype) taking data structured under the source schema (legacy EHR). For this purpose, we require an explicit representation

of how the source schema and target schema are related to each other. These explicit representations are called (schema) mappings (5,6).

The effort required to create and manage mappings is considerable since it involves writing and managing complex data transformations programs. A simple approach is to write intricate custom and non-reusable software in a general purpose language to perform the required transformations. A more elaborated alternative is to use a “specify-generate” approach where high-level declarative assertions are used to specify the relationship between the source and target schemas. The assertions are then compiled automatically into executable scripts (such as SQL/XQuery). This approach makes it possible to separate the design of the relationship between schemas from its implementation.

In this chapter, we describe a declarative approach to specify mappings and from them to generate automatically data transformation scripts expressed in XQuery that may be used to integrate and communicate EHR systems. Our solution is based on the large body of research on data exchange. The existing formalisms has been studied and adapted to cope with the special requirements of archetypes.

We will also explain how this approach has been implemented in the LinkEHR platform to support the mapping of archetypes based on any reference model. Our main requirements on the target instances are that they shall be compliant with the target standard, be non-redundant and contain all the available source information (144,145). Furthermore, we will study how to incorporate expressive mappings that: a) not only cope with value couplings but also with structural mappings b) take into account the wide range of constraints that can be specified in archetypes.

Different parts of this chapter have previously published in two papers in the Journal of Biomedical Informatics: “Using the ResearchEHR platform to facilitate the practical application of the EHR standards”¹ and “Interoperability of clinical decision-support systems and electronic health records using archetypes: a case study in clinical trial eligibility”².

3.2. Data Model

We need first to introduce the data model that is used to represent the source and target schemas in our mapping framework. The source schemas may be either a XML Schema or an

¹ Available at <http://www.sciencedirect.com/science/article/pii/S1532046411001924>

² Available at <http://www.sciencedirect.com/science/article/pii/S1532046413000701>

archetype expressed in ADL whereas the target schema is an archetype expressed in ADL. Thus, the data model shall be capable of representing both formalisms.

A data model is a collection of concepts that can be used to describe a data schema, i.e. the data types, relationships, and constraints that should apply on the data. Since archetypes (ADL) impose a hierarchical structure to the EHR we have chosen the nested relational (NR) model (146). Furthermore, the NR model is the base of the existing mapping formalisms for hierarchical data transformation. Therefore, its use will allow us to apply the existing formalisms and methodologies to archetypes.

The NR model generalizes the relational model where tuples and relations are modeled as records and set of records respectively. In the NR model, a non-atomic element (either records or set of records) can be nested inside another element to build complex hierarchies as those defined by archetypes. The proposed data model is similar to the data model described in (147) but it has been adapted to deal with archetype data definition capabilities.

The set of atomic data types of our model are those supported by ADL archetypes, namely: string, integer, real, date, time, date and time, duration and Boolean. Non-atomic types are record types of the form $\text{Rcd}[a_1^{(e_1:f_1)}:\tau_1, \dots, a_n^{(e_n:f_n)}:\tau_n]$, set types of the form $\text{SetOf}[\tau_1^{(l_1:u_1)} \dots \tau_n^{(l_n:u_n)}]^{(l_\tau:u_\tau)}$ and choice types of the form $\text{ChoiceOf}[a_1:\tau_1 \dots a_n:\tau_n]$ where:

1. τ represents either an atomic, set, or record type
2. $n \geq 1$, $e_i \in \{0,1\}$, $f_i \in \{0,1\}$ and $e_i \leq f_i$
3. l_i is a natural number, u_i is a natural number or ∞ and $l_i \leq u_i$
4. $\sum_{i=1}^n l_i \leq u_\tau$ and $\sum_{i=1}^n u_i \leq l_\tau$

The symbols a_1, \dots, a_n are called label or attributes.

Record values of type $\text{Rcd}[a_1^{(e_1:f_1)}:\tau_1, \dots, a_n^{(e_n:f_n)}:\tau_n]$ are ordered tuples of attribute-value pairs: $[a_1 = v_1, \dots, a_n = v_n]$ where v_1, \dots, v_n must be of types τ_1, \dots, τ_n respectively. In record types $(e_1:f_1)$ represents the existence constraints, for instance (1:1) means the attribute is mandatory.

Set values of type $\text{SetOf}[\tau_1^{(l_1:u_1)} \dots \tau_m^{(l_m:u_m)}]^{(l_\tau:u_\tau)}$ are set of values of one of the types τ_1, \dots, τ_n . In set types, $(l_\tau:u_\tau)$ represents the cardinality of the corresponding attribute

whereas $(l_i:u_i)$ represents the occurrences constraints (how many times an instance of a given type can occur).

Choice values of type $\text{ChoiceOf}[a_1:\tau_1 \dots a_n:\tau_n]$ are an attribute-value pair $a_i = v_i$ where v_i must be of type τ_i . Note that the ChoiceOf type models alternatives of attributes a feature of XML Schema that is not supported by archetypes. However, archetypes support alternatives of types which are modeled by set types with upper cardinality equal to 1 in our formalism. For instance, the following type definition models an alternative of two types QUANTITY[at0005] and QUANTITY[at0006] :

$$\text{typeValue} ::= \text{SetOf}[\text{QUANTITYat0005} \text{QUANTITYat0006}]^{(1:1)}$$

For simplicity of presentation, we will assume strict alternation of set/Choice and records types in a schema and default values in ADL 1.4 for existence, cardinality and occurrences. Table 4 contains the representation of several archetypes constraints using our data model. Note that we use SetOf to model the content of attributes (both mono-valued and multi-valued).

In Figure 26, we show a simple ADL excerpt that may be represented as:

$$\text{HISTORY} ::= \text{Rcd}[\text{periodic: Boolean value: typeValue events: typeEvent}]$$

$$\text{typeValue} ::= \text{SetOf}[\text{QUANTITYat0005} \text{QUANTITYat0006}]^{(1:1)}$$

$$\text{QUANTITYat0005} ::= \text{Rcd}[\text{magnitude: int property: string units: string}]$$

$$\text{QUANTITYat0006} ::= \text{Rcd}[\text{magnitude: int property: string units: string}]$$

$$\text{typeEvent} ::= \text{SetOf}[\text{Event0002}^{(0:1)} \text{Event0003}^{(1:2)} \text{Event0004}^{(0:\infty)}]^{(0:\infty)}$$


```

HISTORY occurrences ∈ {1} ∈ {
  periodic ∈ {False}
  value matches {
    QUANTITY[at0005] matches {
      magnitude matches {0..55}
      property matches {"velocity"}
      units matches {"mph"} -- miles per hour
    }
    QUANTITY[at0006] matches {
      magnitude matches {0..100}
      property matches {"velocity"}
      units matches {"km/h"} -- km per hour
    }
  }
  events cardinality ∈ {*} ∈ {
    EVENT[at0002] occurrences ∈ {1..1} ∈ {}
    EVENT[at0003] occurrences ∈ {1..2} ∈ {}
    EVENT[at0004] occurrences ∈ {0..*} ∈ {}
  }
}

```

FIGURE 26 ADL EXCERPT

Archetype constraint	Representation in NR model
Mandatory primitive attribute attr	$attr^{(1:1)}$: atomic type
Optional primitive attribute attr	$attr^{(0:1)}$: atomic type
Class with attributes $attr_1 \dots attr_n$	$Rcd [attr_1^{(e_1:f_1)}: \tau_1, \dots, attr_n^{(e_n:f_n)}: \tau_n]$
Mandatory mono-valued attribute attr. In case $m=1$, we model an alternative.	$attr ::= SetOf [\tau_1^{(l_1:u_1)} \dots \tau_m^{(l_m:u_m)}]^{(1:1)}$
Optional mono-valued attribute attr. In case $m=1$, we model an alternative.	$attr ::= SetOf [\tau_1^{(l_1:u_1)} \dots \tau_m^{(l_m:u_m)}]^{(0:1)}$
Multi-valued attribute attr with cardinality = $\{c_l..c_u\}$	$attr ::= SetOf [\tau_1^{(l_1:u_1)} \dots \tau_n^{(l_n:u_n)}]^{(c_l:c_u)}$
Class τ_i with occurrences = $\{l_i:u_i\}$	$\tau_i^{(l_i:u_i)}$

TABLE 4 REPRESENTATION OF ARCHETYPE CONSTRAINTS IN THE PROPOSED DATA MODEL

3.3. Source and target schemas

3.3.1. XML Schemas

Type definitions in XML Schema can be reused in multiple places what may hinder the mapping definition process. It becomes necessary to generate the exhaustive nested schema for instance data by “unfolding” all the types at the place they are used in order to univocally reference source data element. For this purpose, we have developed an algorithm that analyses the schema and generates a visual tree that contains all possible paths from the root to the atomic elements/attributes that may appear in instance XML documents. Figure 27 depicts an example, where the asterisk denotes a SetOf type. As a result, users do not have to deal with the complexity of XML Schema when mapping archetypes. Note that for each XML Schema the user must indicate the entity (element/attribute) that contains the patient identifier. This information is vital in order not to mix data of multiple patients in a single EHR extract.

The resulting mapping tool, LinkEHR, can also handle source relational schemas, as long as they are converted in a canonical way into a W3C XML Schema. The tool extracts an existing relational schema from a relational database using JDBC drivers. The imported schemas, stored in an internal format, are used as the basis for GUI management of such source schemas. Once a schema has been imported, users can select the portion of the database (tables and relationships) that is relevant for a particular archetype. Only those tables that are related directly (by means of a foreign key) or indirectly (by means of a foreign key path) to the table that contains the patient identifier can be used in a mapping specification. Based on the relationships between the root table and other tables, the tool generates a hierarchical view expressed as a W3C XML Schema. Since the root entity corresponds to the table that contains the patient identifier, all the patient data is nested inside this root node. The user selection may contain cycles; the tool assists the users in their elimination before generating the hierarchical view.

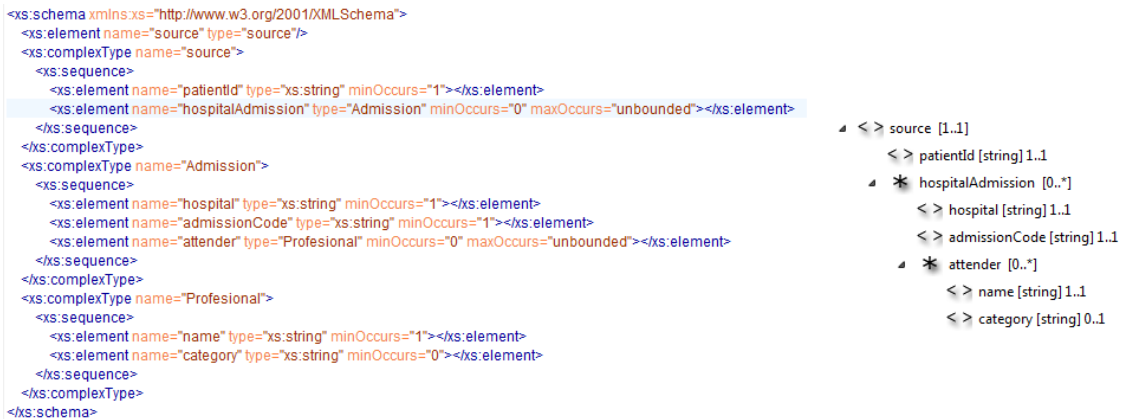


FIGURE 27 EXAMPLE OF TREE VIEW OF AN XML SCHEMA

3.3.2. Archetypes

In ADL only the constrained entities (classes and attributes) of the reference model need to appear in the archetype definition. It is supposed that the constraints imposed by the underlying reference model are implicit constraints. The same occurs when specializing an archetype, all the constraints in the parent archetype are implicit constraints of its specializations. Note, that this is consistent with the object-oriented paradigm, where attributes and methods of a superclass are automatically inherited by all its subclasses. The main advantage of this rule is that archetype definitions in ADL are kept simple. For instance, if all the classes and attributes of the reference model were to be included, archetypes would have many constraints (hundreds in EN13606) making the archetype definition unnecessarily complex.

This rule poses a difficulty when an archetype is to be mapped to a data source. In many cases it would be necessary to map an unconstrained attribute, hence not present in the archetype. Note that our final objective is to generate XML documents compliant with the reference model. Thus, when an archetype needs to be mapped it becomes necessary to complete the archetype definition with the reference model. We have implemented a merge function that takes an archetype and the underlying reference model as inputs and outputs what we call a comprehensive archetype. A comprehensive archetype includes all the explicit constraints (those defined by the archetype to be mapped) and all the implicit ones (those defined by the reference model) that data instances must satisfy. Figure 28 shows an example of comprehensive archetype. On the left-hand side the original CEN/ISO 13606 archetype is depicted, whereas the corresponding comprehensive archetype is shown on the right-hand side. As it can be observed the comprehensive archetype contains all the constraints of the

original archetype as well as all the unconstrained entities from the reference model such as `act_status` and `archetype_id`.

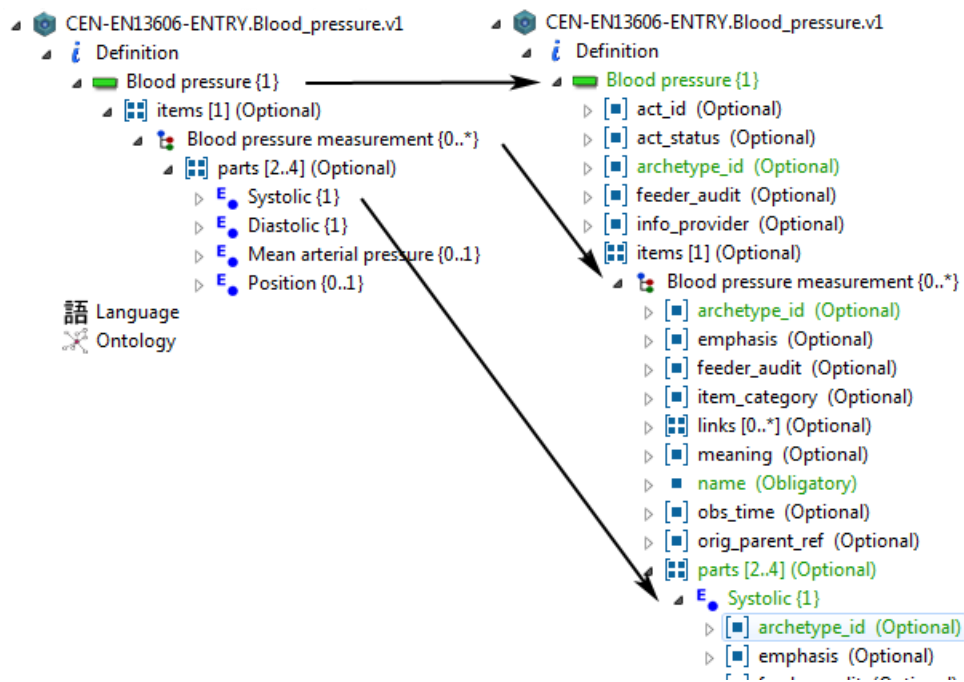


FIGURE 28 BLOOD PRESSURE ARCHETYPE AND AN EXCERPT OF THE COMPREHENSIVE ARCHETYPE SIDE BY SIDE

3.4. Mapping Language

The mapping language is based on the tgds (tuple-generating-dependencies) (4) proposed in (78,147). They are expressive enough to represent, in a declarative way, many of the schema mappings of interest (147). The tgds basically define a value correspondence, i.e. how to compute a value for an atomic attribute of the target schema (archetype) by using a set of atomic elements from the data source. In our setting a value correspondence is defined by a set of pairs, consisting of a transformation function and a filter. The latter contains the conditions that source data must satisfy to be used in the transformation function. All the atomic values, either from source or target data, eventually devolve to instances of the primitive types of the archetype model, namely String, Integer, Real, String, Date, Time, DateTime, Period and Boolean. Filters expression must yield a Boolean value whereas transformation functions must produce a value compatible with the type of the target attribute. Value correspondences allow us to hide much of the structural complexity of archetypes and reference model. Users do not need to specify the logical relations between the entities of the source and target schemas. It is only necessary to specify the navigation path of the attributes used in the mapping.

The simplest kind of transformation function is the identity function which copies a single source value into a target value. But quite often it is necessary to specify arbitrary complex functions which transform a set of source values into a target value. To achieve this, a wide range of transformation functions are supported. They can be divided into nine categories as described in Table 5. The example in Figure 29 illustrates a simple value correspondence for transforming gender codes. It transforms the local gender code in the path `/patient/gender` of an XML EHR fragment (source data) into a normalized code to be stored by somewhere within an archetype (target data). Note that the order is relevant and only one mapping function is applied. Therefore, this correspondence should be interpreted as:

```

If (/patient/gender='M' OR /patient/gender='m') then 0
Else if (/patient/gender='W' OR /patient/gender='w') then 1
Else if (/patient/gender=0 OR /patient/gender=1) then /patient/gender
Else 9

```

Condition	Mapping function
<code>/patient/gender = 'M' OR /patient/gender = 'm'</code>	0
<code>/patient/gender = 'W' OR /patient/gender = 'w'</code>	1
<code>/patient/gender = 0 OR /patient/gender = 1</code>	<code>/patient/gender</code>
true	9

FIGURE 29 EXAMPLE OF VALUE CORRESPONDENCE TRANSFORMING THE GENDER CODES FROM A XML SOURCE.

Category	Description	Examples	Sample mapping
Set value	Enable to set a fixed value to an archetype atomic attribute. The value can be either a constant or an expression involving several constants.		<code>/source/value</code>
Type conversion	Set of functions for the conversion from one type to another.	<code>toString</code> , <code>toInteger</code> , <code>toFloat</code>	<code>toString(58.7)</code>
Mathematical	Standard mathematical functions for numerical calculation using several	<code>+</code> , <code>div</code> , <code>mod</code> , <code>round</code> , <code>ceiling</code>	<code>(10 + 25) mod 30</code>

	source values.		
Logical	Main logical functions including comparison operators.	AND, OR, TRUE, FALSE, <, >, <>, NOT	/source/value < 30
String	Main operator for handling string values.	concat, matches, contains, substring_before	matches("abc","[a-z]+")
Date and time	Transformation of source values into values conforming to the international standard ISO 8601 for date and time representation or the extraction of portions of date or time expressions.	toISODate, toISOTime, day-From-DateTime, minutes-From-Time	toISODate(/source/date,"yyyy-MM-dd")
Archetype vocabulary	Allow the access to archetype metadata such as entity identification or the text and description attached to archetype entities	Code, id, description, text	description("at0010")
Terminology	Functions that allow terminology abstraction by reasoning over the acyclic taxonomic (is-a) hierarchy of SNOMED CT,	In, ascendants, descendants, union, intersection	in(/diagnostic/code, @descendants ("128462008"))
Grouping	Functions applied to a source path representing pointing to a set. Takes into consideration this set of paths to perform operations using all set values.	Count, max, min, average	max(/source/value)

TABLE 5 FUNCTIONS SUPPORTED BY LINKEHR MAPPING

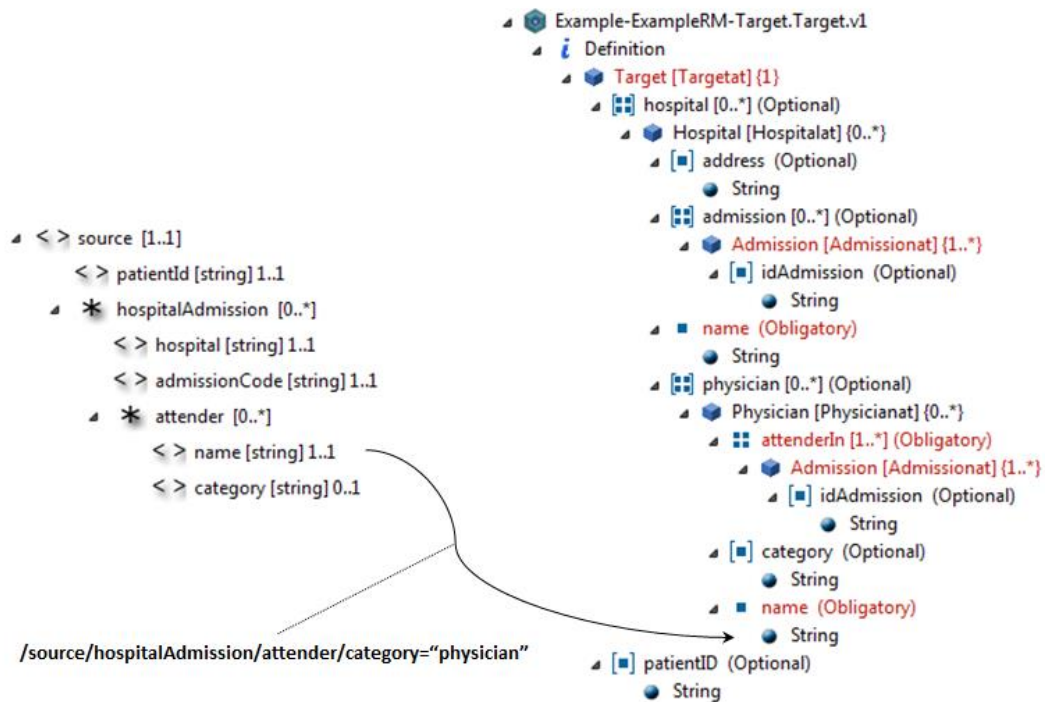


FIGURE 30 AN EXAMPLE OF VALUE CORRESPONDENCE

Value correspondences can be translated into source-to-target tgds. For instance the value correspondence depicted graphically in Figure 30 translates to the following tgd:

$$\forall h \in \text{source.hospitalAdmission}, a \in h.\text{attender} \mid a.\text{category} = \text{"physician"} \rightarrow$$

$$\exists h' \in \text{Target.hospital}, p \in h'.\text{physician} \mid p.\text{name} = a.\text{name}$$

Note that the above tgd is essentially a GLAV mapping, where the left hand side of the implication is a query over the source (Q^S) and the right hand side is a query over the target (Q^t). The expression specifies a containment assertion: for each record value returned by Q^S there must exist a corresponding record value in Q^t . We also point out that the above tgd (and any tgd) does not capture the existence (the attribute “name” is mandatory in the target), cardinality and occurrence constraints. We enforce these constraints in the generated XQuery.

One of the main problems of using value correspondences is that they must be combined in order to generate a complete mapping. Value correspondences lack expressive power and some mapping details must be worked out (146). The main problem is related to the grouping semantics. Grouping semantics describes when instances should be grouped and nested into a SetOf instance. Consider Figure 31 where a very simple, but yet illustrative, mapping scenario is depicted. The target schema is nested on an extra level (procedures set). The value correspondences require all patientId and SurgicalProcedure that can be found in the source to

be transferred to the target. However, the intended semantics dictates that all the different SurgicalProcedure shall be grouped together, for the same value of patientId. This behavior cannot be captured by a tdg which is stated at the level of flat record instances. As a result we will obtain a different target instance for every different combination of (patientID,SurgicalProcedure) in the source and every target instance will have just one nested procedure. This output satisfies the tgds but it is not the correct one.

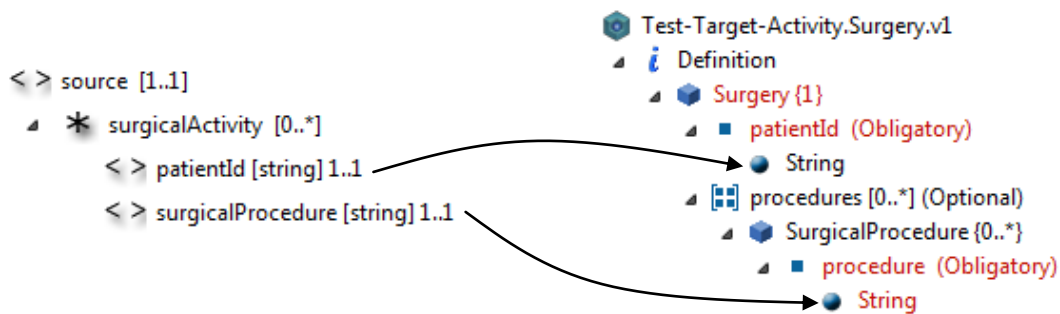


FIGURE 31 GROUPING MISMATCH BETWEEN SOURCE AND TARGET SCHEMAS

Our default grouping semantics is heavily inspired on CLIO (81). It is based on the Partition Normal Form (PNF), i.e. we impose that the resulting target instances will satisfy the PNF (148). This forces that in any target nested relation there cannot exist two distinct records that coincide on all the atomic elements, in other words the non-multivalued attributes are the key of nested relations. In Figure 32 the nested relations A and C are in PNF unlike B. As can be observed C contains the same information as B, in fact it is the normalized version of (b).

Entity A	Entity B	Entity C
a	{1,2,3}	b
a	{3,5}	c

(a)

Entity A	Entity B	Entity C
a	{1,2,3}	b
a	{3,5}	c

Entity A	Entity B	Entity C
a	{2,4,7}	b

(b)

Entity A	Entity B	Entity C
a	{1,2,3,4,7}	b
a	{3,5}	c

(c)

FIGURE 32 NESTED RELATIONS EXAMPLES, (A) AND (C) ARE IN PNF AND (B) IS NOT

This default grouping strategy has been proven suitable for most use cases. Since much of context information (such dates or authors) is mono-valued, therefore clinical data that share this context are grouped together. To achieve PNF on the target instances, we use skolem functions. A skolem function returns a different value (identifier) for each combination of parameters. A new instance will be only created if another instance with the same identifier does not exist.

Our skolemization algorithm is based on a schema that associates to each type definition in the comprehensive archetype a subset of the atomic attributes of the comprehensive archetype. This subset controls the creation of fresh values in the target instances: they are the parameters of the corresponding skolem function. The algorithm is as follows:

Input: a comprehensive archetype A and a set of correspondences of values S

Output: skolem function parameters for each set type node in A controlling the creation of instances of archetype complex types

Propagate each atomic attribute of the comprehensive archetype to the complex type nodes, in any of the following two ways:

1. Propagate up the atomic attributes with a correspondence of value until a multivalued attribute is found.

2. Propagate down the atomic attributes with a correspondence of value.

The propagation rules are applied recursively until no rules can be applied.

The propagation process ends in a unique configuration in which each type node in the comprehensive archetype has a list of the atomic attributes.

Figure 33 shows the result of the application of the previous algorithm to the example in Figure 31. Note that the skolem function of the type “procedures” has two parameters “/patientId” and “/procedures[at0001]/procedure”. Therefore, a different instance of SurgicalProcedure will be created for each combination of patientId and procedure values. Note that the archetype attribute /patientId is mapped to source path /source/surgicalActivity/patientId and the attribute /procedures[at0001]/procedure to /source/surgicalActivity/surgicalProcedure as shown in Figure 31.

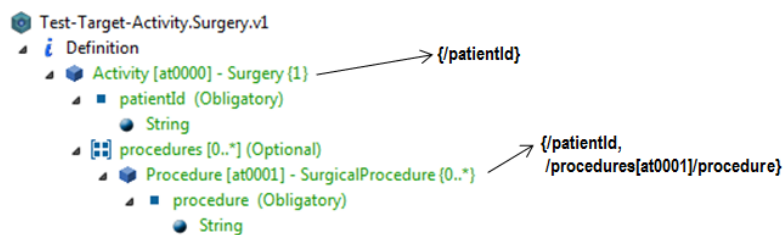


FIGURE 33 ANNOTATED ARCHETYPE WITH SKOLEM PARAMETERS

Figure 34 depicts a more complex example of the application of the previous algorithm. For instance, a fresh value in the set type Physician will be created for every combination of patientId, hospital name, physician name and physician category.

3.5. Customization of grouping semantics

The semantic of the data transformation is defined in two parts. First it is defined in the set of value correspondences and then in the default (implicit) grouping semantics. The nested nature of archetypes makes the grouping semantics a key aspect. An important limitation is that the default grouping semantic is not specified declaratively. Therefore, it cannot be customized when it is not the desired semantics. In order to address this issue we propose two different mechanisms to allow users to control the structural transformation.

3.5.1. Modification of skolem functions

Skolem functions can be modified in order to change the default grouping semantics, although in a quite limited way. As can be observed, the set of parameters of outer types are included in

the set of inner nested types. This is important since it assures that the grouping context is propagated downwards. In Figure 34 the skolem function of the archetype root (Target) has a single parameter (/patientId). Intuitively this means that all the nested data is about a single patient. The nested type Hospital has an extra parameter (/hospital[hospitalat]/name). Therefore, the instances of this type will contain information for a patient in a particular hospital. The same rationale is applied to the remaining types. This containment relationship must be maintained in order to assure data coherence. For instance, if “/patientId” is removed from the parameter set of the Hospital type, it will be possible to have data about different patients in a single Hospital instance. In order to prevent this situation, the deletion of a parameter is propagated to the remaining parameter sets.

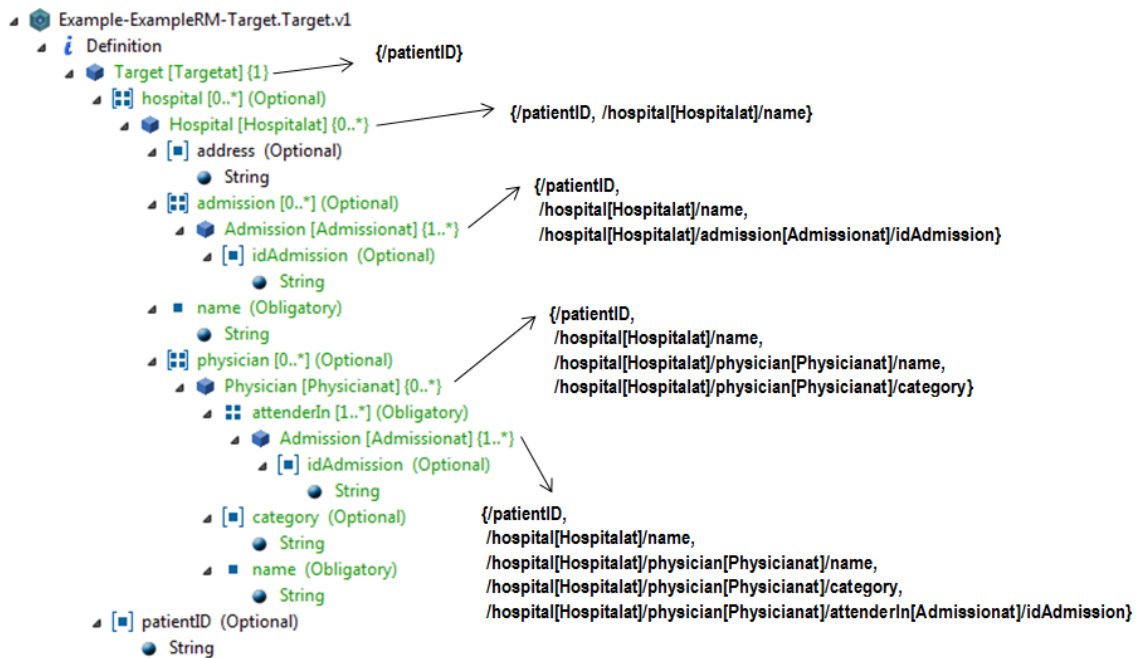


FIGURE 34 PARAMETER OF THE SKOLEM FUNCTION TO CONTROL THE CREATION OF VALUES IN SET TYPES

3.5.2. Object Builders

As stated before default grouping semantic is not always adequate. To illustrate this, consider the mapping example shown in Figure 35. The source schema is a nested schema describing departments with their set of employees and project. The target schema is an archetype which is a slight variation of the source schema. It also groups together employees and projects per department but the department name is not included. The value correspondences relate source project and employee names with target project and employee names. The skolemization algorithms yields the empty set for the archetype root (Target) and Dept[at0001]. In this case and since there is not information available about the generation of

Dept instances a single instance is generated. The single Dept instance encloses all the project and employees, not preserving the containment and sibling relationships in the source. What we need is to control the creation of target instances, in such a way that a new Dept instance is constructed for each source department node.

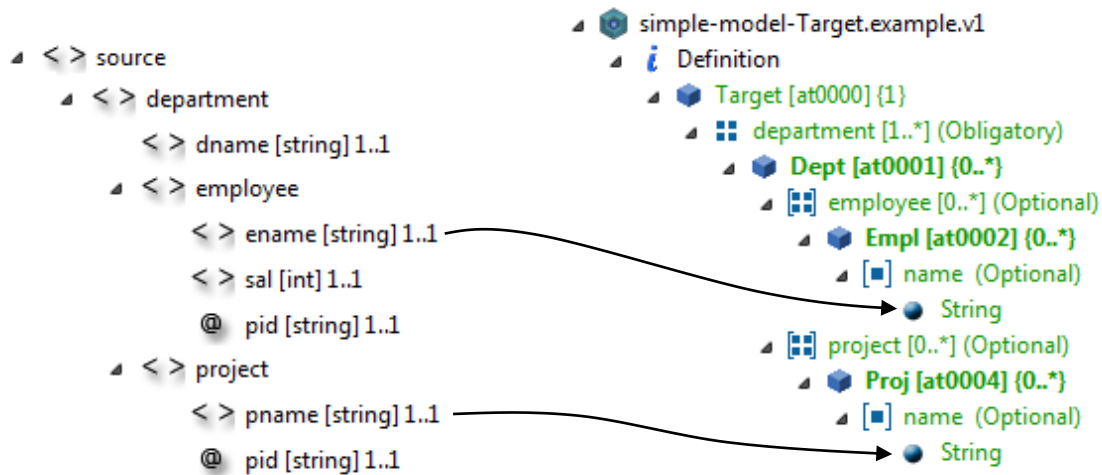


FIGURE 35 SIMPLE MAPPING SCENARIO

A natural extension to the mapping language is the inclusion of assertions that connect complex schema elements (instead of atomic elements as value correspondences). This type of assertion are called object builders (78). An object builder has:

- A set of incoming source complex elements that can be tagged with variable names.
- One or none outgoing CComplexObject element of the target archetype. The upper occurrences of the CComplexObject must be greater than 1. The archetype root is an exception to this rule.
- A filter on the source data.
- One or none incoming object builder.
- Zero to many outgoing object builders.

Intuitively, an object builder defines an iterator on the source nodes they are drawn from (a), in each iteration, a new target element (b) is generated for each combination of source values that satisfy the filter (c). It is possible to define hierarchies of builders where parent builders propagate its context to its children. Figure 36 illustrates an example of mapping scenario with three object builders. This mapping solves the problem discussed previously. The topmost builder generates target Dept elements and its context is propagated to the remaining

builders. One of them generates Empl elements while the other generates proj elements considering the current topmost mapping into Dept. Furthermore, only those employees with a salary (source attribute 'sal') greater than 10000 are selected.

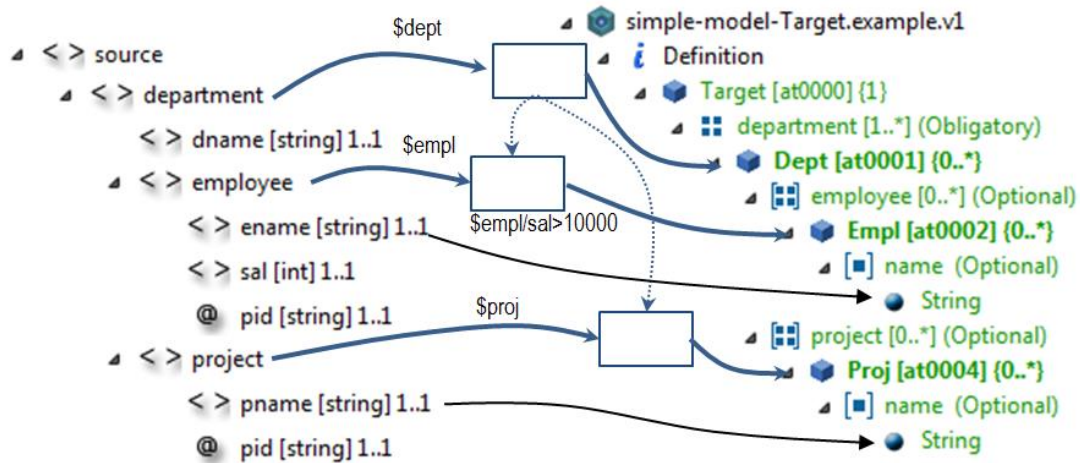


FIGURE 36 SIMPLE MAPPING SCENARIO WITH OBJECT BUILDERS

3.6. XQuery generation

3.6.1. Mapping covers

When two or more complex types in ADL archetypes appear nested inside an attribute which is not a container (i.e. for which there is no cardinality constraint) are taken to be alternative constraints, only one of which needs to be matched by the data. In Figure 26, `QUANTITY[at0005]` and `QUANTITY[at0006]` are alternatives for the value attribute, leading to the result that only one instance of one of both types can appear in runtime. This useful feature provides an additional challenge in the generation of transformation programs as only one of the alternatives can be present in target data. If more than one alternative has a complete mapping (i.e. there are enough value correspondences to generate a valid instance) then we have different ways of producing valid target instances. We overcome this problem by generating what we call cover archetypes. A cover archetype of archetype A is generated by removing all but one of the child type constraints of single-valued attributes, i.e. by selecting one of the alternatives and removing the rest. If we take as example the archetype in Figure 26 two different covers can be generated, one where only `QUANTITY[at0005]` remains and other where `QUANTITY[at0006]` remains. We note that this might lead to an explosion in the number of covers if many alternatives are defined in the archetype. To alleviate this problem, we

automatically remove those alternatives that have not enough value correspondences to create valid target instances.

3.6.2. Representation of domain constraints

Comprehensive archetypes may contain constraints on primitive types that may come either from the reference model or the archetype itself. This kind of constraint is used to limit the set of possible values that an atomic attribute can hold. For instance, Integers can be constrained by using a list of integer values or an interval whereas Strings can be constrained in two ways: by using a list of fixed strings or a regular expression. Since they are applied to atomic attributes, these constraints must be taken into account in value correspondences.

Fixed values are the most restricted type of domain constraint. We transform automatically fixed values into value correspondences, whose filter is the true value (it is always applied) and the transformation function is a set value function which assigns the fixed value to the corresponding atomic attribute. Therefore, the fixed value will always appear in instance data.

Constraints on primitive types that do not impose a single fixed value are treated in a different way. User needs to specify value correspondence since there are multiple possible values. The domain constraints are enforced in the generated XQuery. We will discuss our generation of XQuery scripts in the following section. Table 6 shows two examples of how constraints are automatically compiled into XQuery transformation script. First row contains an example of a constraint that imposes the fixed value 1 to the attribute number. Whereas the second row describe how a range constraint on an integer attribute is compiled into the XQuery transformation script. In the example, the value is contained in the path `/data/measurements/value` of the input XML document.

<i>Archetype constraint</i>	<i>Value correspondence</i>	<i>XQuery extract</i>
number matches {1}	if true then 1 (automatically generated)	Return <number> 1 </number>
number matches { 1..10 }	if true then /data/measurements/value (specified by user)	For \$val in /data/measurements/value Where \$val≥0 and \$val≤10 Return <number> {data(\$val)}</number>

TABLE 6 EXAMPLES OF HOW CONSTRAINTS ARE PROCESSED

3.6.3. Translation of mappings into XQuery

It is important to note that XQuery is a possibility among others for implementing the mapping specification. Since the proposed mapping language is declarative the mapping specification can be “compiled” in other languages such as XSLT. We chose XQuery due to its rich set of features that allow many different types of operations on XML documents such as selecting information based on specific criteria, filtering out unwanted information, sorting, grouping and aggregating data, performing arithmetic calculation on number and dates, manipulating string and transforming and restructuring XML data into another structure.

Taking into account the abstract mapping specification, the archetype constraints and the source schema an XQuery script is generated for each cover archetype. The resulting transformation script takes as input an instance of the source EHR data and generates a XML document that is compliant both with the archetype and the underlying reference model. The template of such XQuery programs is shown in Figure 37.

For each object node (specialization of a reference model class) in the archetype a nested XQuery FLWOR expression (149) is produced which contains:

- a) An uppermost LET clause that contains all the combinations of a flat view of all the data to be structured according the target archetype object node. This LET clause includes a set of FOR clauses that iterates over the relevant source elements(for instance incoming nodes of object builders), a set of LET clauses calculates target values using values correspondences, a WHERE clause that includes the correlation with parent object node and domain constrains on single-valued attributes, and a RETURN clause that projects on the calculated target values;
- b) A FOR clause (\$combo) that iterates over the distinct values of the flat view of target values. Note that this flat view contains the values of the mono-valuated attributes, i.e. the parameters of skolem functions.
- c) A WHERE clause that captures the cardinality and occurrence constraints
- d) A RETURN clause that outputs the XML elements for the object node and contains any other nested mapping.

Let \$context***for*** \$var_for **in*****let*** \$var_let (: value transformation :)***where*** (: domain constraints, correlation with parent object node :)***return***

<combo node_ID="...">

(: flat view of data, it uses \$var_for and \$var_let's :)

</combo>

for \$combo **in** ((ibimeFunction:distinct-deep(\$context)))

(: elimination of duplicates :)

FIGURE 37 XQUERY TEMPLATE FOR BUILDING ARCHETYPE INSTANCES FROM SOURCE DATA

Figure 38 shows the complete XQuery that is generated for the object node Admission[Admissionat] in Figure 34. To begin, the values of the skolem parameters are obtained (lines 3, 4 and 5). Note that in this case we bind one variable to each parameter by a *for clause* since the value correspondences just copy the values in the source. Otherwise we should use a *let clause* to calculate the target values. The *where clause* (line 6) contains the correlation with the parameters (patientId and hospital name) of the upper object node (Hospital). The *return clause* at line 7 outputs the flat view of the atomic attributes (skolem parameters in this case), the code in line 8 deletes duplicates and the code in line 9 enforces the occurrence constraint (at least one Admission instance must be exist). Finally, data is structured according to the target schema; this is done in line 10.


```

1  let $aux_4 :=
2    for $source_LinkEHRVar_0 in (/source)
3    for $hospital_LinkEHRVar_0 in ($source_LinkEHRVar_0/admission/hospital)
4    for $admission_LinkEHRVar_0 in (/source/admission)
5    for $idadmission_LinkEHRVar_0 in
6      ibimeFunction:if-empty($admission_LinkEHRVar_0/idadmission,xs:QName("idadmission"))
7    where $source_LinkEHRVar_0/patID=$patID_LinkEHRVar_0 and
8          $admission_LinkEHRVar_0/hospital=$hospital_LinkEHRVar_0 and
9          $combo_1/hospitalHospitalat__name=$hospital_LinkEHRVar_0
10   return
11     <combo archetype_ID="Example-ExampleRM-Target.Target.v1::Admissionat">
12       <hospitalHospitalat__admissionAdmissionat__idAdmission>
13         {data($idadmission_LinkEHRVar_0)}
14       </hospitalHospitalat__admissionAdmissionat__idAdmission>
15     </combo>
16 for $combo_2 in ((ibimeFunction:distinct-deep($aux_4)))
17 where count(ibimeFunction:distinct-deep($aux_4))>=1
18 return
19   <admission xsi:type="Admission" archetype_ID="Example-ExampleRM-Target.Target.v1::Admissionat">
20     <idAdmission>
21       {data($combo_2/hospitalHospitalat__admissionAdmissionat__idAdmission)}
22     </idAdmission>
23   </admission>

```

FIGURE 38 EXAMPLE OF GENERATED XQUERY

3.7. LinkEHR mapping module

LinkEHR can be considered as a high-level schema mapping tool. In LinkEHR, users are responsible of defining an abstract non-procedural mapping specification. This abstract representation is specified using a set of value correspondences between the atomic attributes of archetypes and source schemas that can be complemented with object builders. The abstract specification is then compiled into an executable transformation script expressed in XQuery. Figure 39 shows the overall architecture of LinkEHR mapping module.

In order to ease the definition of mappings, a completely new mapping perspective was added to the tool. This perspective allows the edition of complex attribute mappings without the need of typing text. This perspective also adds a set of specific visual interfaces for defining attribute and object mappings, and a data source manager to import, edit, and delete data sources in the tool.

Data source manager allows the import and use of relational data sources, XML schemas, and archetypes as source schemas for the transformation. In case of sources based on XML Schema, users must provide the location of the schema or schemas, an alias, the path to the patient identifier, and the root entity from all the entities available in the XML schema. For archetype import as a source, no extra parameter must be configured, as archetypes from supported reference models already have all the needed information in the documentation file

(see chapter 2). In the case of relational data sources, they are included as LinkEHR Integration Engine source definition files and do not need any additional configuration parameter.

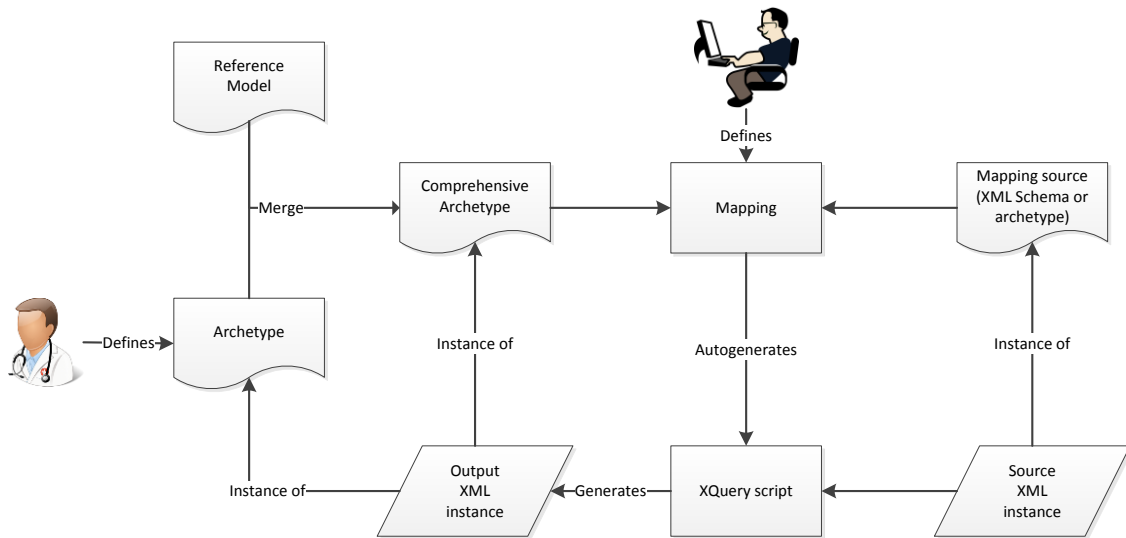


FIGURE 39 MAPPING AND XQUERY GENERATION

For the specification of the transformation target, the starting point is the archetype being edited in LinkEHR editor. Once the user has finished specializing the constraints that apply to a particular use case, which is typically called 'templating' or 'creating a template' in dual model architectures, the editor can be switched to mapping perspective and mapping process can begin.

This mapping perspective completes the archetype with the underlying reference model, transforming current archetype into a comprehensive archetype (see Figure 28 example) containing both the constraints coming from the archetype and the ones coming from the reference model. This perspective also freezes the archetype preventing their defined constraints to be edited, although they can still be reviewed. As comprehensive archetypes become very big and complex we discarded in the use of a graphical approach (e.g. showing both source and target structures, lines for the mappings, and boxes for the operators). In mapping editing perspective, only the comprehensive archetype is presented to the user and specific attribute mappings and object mappings are shown when the user selects a suitable node in the comprehensive archetype, mimicking the original LinkEHR editor perspective.

3.7.1. Mapping management

As mentioned above, comprehensive archetype can become big and complex. To alleviate this, mapping perspective was provided with a set of mechanisms and visual clues to ease mapping definition.

Filters

To reduce the quantity of information presented to the user, a filtering mechanism was included. This filtering mechanism is defined for both label filters, i.e. filter nodes by their label or node identifier, and source filters, i.e. filter nodes by certain conditions on the mapping source, such as mappings to constant values, mappings to a given source, nodes containing object mappings, or nodes with multiple mappings (mapping alternative). Both filters can be used simultaneously to filter by two different criteria.

Coloring

As archetypes define complex structures with combinations of mandatory and optional objects and attributes, it could be difficult to know when there are enough defined mappings to define a correct data instance (i.e. enough mappings to generate a transformation program that complies with both the archetype and the reference model). In order to ease mapping and transformation process visual metaphors were introduced into the archetype visual tree. Three different colors are used to provide feedback to the user: Green, meaning that the branch has all the needed mappings to generate a correct instance, red, meaning that the branch is missing a needed mapping to be able to generate a correct instance, and black, meaning that this branch is optional and has no mapping assigned. Once the user defines a mapping, the archetype tree is tested to be instantiable (i.e. able to generate a correct instance), and updates the entire tree with the corresponding colors. Once the root node of the archetype has been green colored (i.e. at least a data instance containing the root node can be generated) the process to generate the transformation program can be started. There are also visual metaphors to tell if an object contains object mappings. In this case the font from the node is put into bold format if the object has an object mapping attached to it.

Entity cloning

Archetypes allow the definition of multiple occurrences to objects (e.g. to define a set or list of medications, allergies, phones, or surnames). In mapping perspective, objects with multiple occurrences can be cloned in order to define mappings to other paths from the same source. Already defined mappings are also copied and can be modified later on. Object clones created this way allow the definition of different object builders for the original object and the cloned one.

3.7.2. Mapping reuse

Mapping propagation

When the comprehensive archetype is generated all the context attributes from the underlying model are included. More often than not, the mappings of the same context attribute in two different classes from the same type is exactly the same (e.g. 'name', 'archetype_id', and 'synthesised' attributes from ENTRY class share the same kinds of mappings). An option was added to automatically copy all the mappings of a given type to all the entities of the same type in the comprehensive archetype.

Attribute mapping copy

In addition to mapping propagation, an option to reuse individual attribute mappings was introduced, which we called "favorite mappings". These mappings have aliases in order to easily identify them. These mappings are kept from one mapping session to another. Available favorite mapping list only shows the mappings that come from the selected data source. Mappings that can be evaluated to a constant are always included in this list.

Archetype slot mapping reuse

Dual model allows reusing archetypes externally defined with the slot mechanism. These archetypes can already have mappings defined. The transformation program generation process allows reusing these mappings when the slot in the parent archetype is solved, as long as the mappings from the archetype and the included slot use the same data source. The process takes as input a set of schema mappings between the same source and target schema and returns a schema mapping that correlates them, recalculating the target paths of the archetype without the need of intervention by the user. The archetypes included in an archetype slot can also contain slots to other archetypes. In this case, the process composes and correlates the schemas recursively to generate valid mapping paths. To control how the data will be generated, object mappings can also be applied to the slot object.

3.7.3. Value correspondence editing

Value correspondences can be assigned to every leaf node in the archetype (i.e. primitive types). As seen in Figure 40, value correspondence mapping editor presents the user with a filter/function table. This figure defines the same mapping as Figure 29. The edition of both the filters and functions is made in an expression editor, which contains all the aforementioned transformation functions (see Table 5), a view of the selected data source, and the list of fixed value constraints from the archetype if applicable. Every transformation function contains an

associated help that explains the function behavior and provides examples of its use. Expression can be typed or created by pressing the desired function button. Figure 41 shows the edition of a mapping condition. When a mapping expression is accepted or validated, the editor checks that the expression is both syntactically valid and that it is correct for the source and target schemas, e.g. that defined source paths exist in the source, that conditions return a boolean, and that the function returns the correct type for the archetype primitive object.

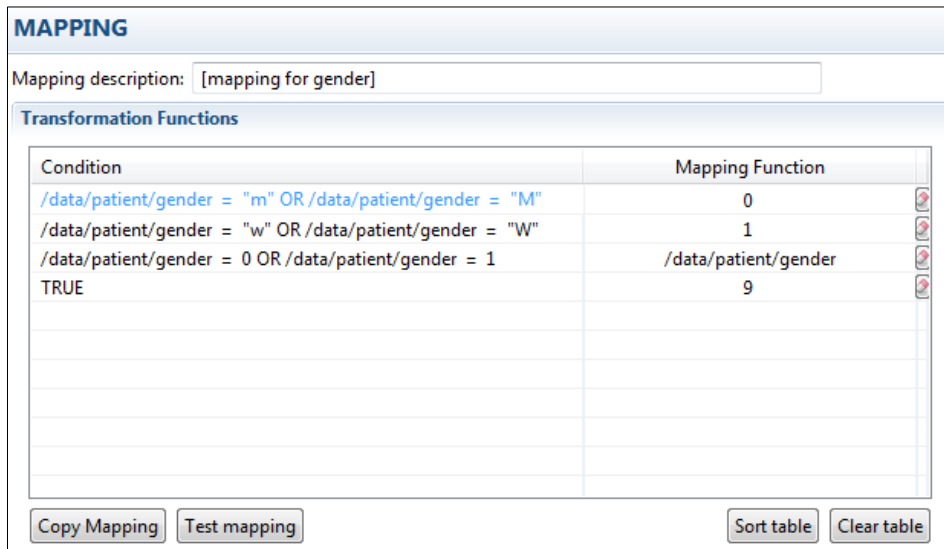


FIGURE 40 EDITING A MAPPING FUNCTION TABLE

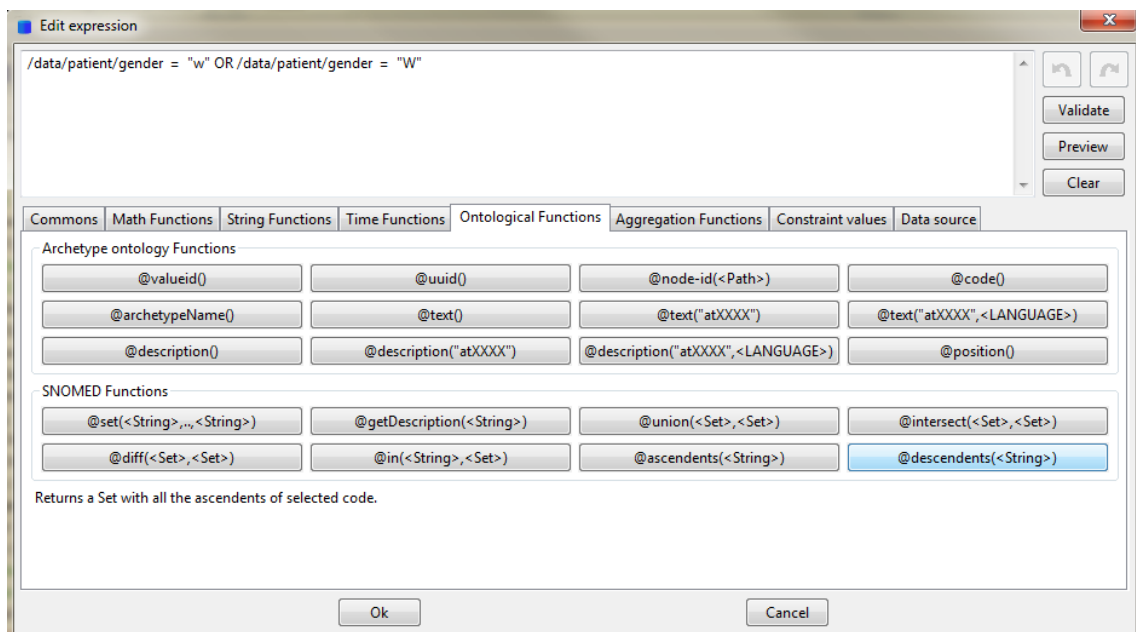


FIGURE 41 EXPRESSION MAPPING EDITOR

3.7.4. Object builder editing

Object mapping editor allows editing the set of incoming source nodes (represented as variables), the filter (which uses an editor similar to the one used in attribute mappings), and the incoming object builder, which is selected from the ones already defined. The main difference between the editor used in value correspondence editing and the object builder filter editing is that condition paths are always dependent of the variables and return type is always boolean. Figure 42 shows an object builder form with an incoming source node '\$pat', and a condition based on it. Parent available object builders are presented in 'Available parents' combo and can be selected to specify the object builder hierarchy.

The screenshot displays the 'Context Builder' interface. It is divided into two main sections: 'Variables' and 'Object Mapping Properties'.
The 'Variables' section contains a text area with the value '\$pat=/data/patient'. Below this text area are three buttons: 'Add variable', 'Edit variable', and 'Delete variable'.
The 'Object Mapping Properties' section includes a 'Condition:' field with the value '\$pat/systolic > 160'. Below the condition field is a checkbox labeled 'Has bag semantics' which is currently unchecked. Underneath is an 'Available parents:' dropdown menu. At the bottom left of this section is the text 'Create context mapping without target path:' and at the bottom right is a button labeled 'Generate context mapping'.

FIGURE 42 OBJECT BUILDER EDITING FORM

3.7.5. Generation of XQuery and Testing

Once there are enough mappings to generate at least a correct instance of the root node of the archetype (and consequently the node is painted green) we can generate the transformation program. A set of covers is generated from the value mappings. Users are presented with that list of covers in order to found the valid combinations. Each cover is identified by the set of the alternatives it contains. The covers can be previewed in the same dialog. Figure 43 shows two different covers for an archetype, with a preview to the actual archetype and their mappings.

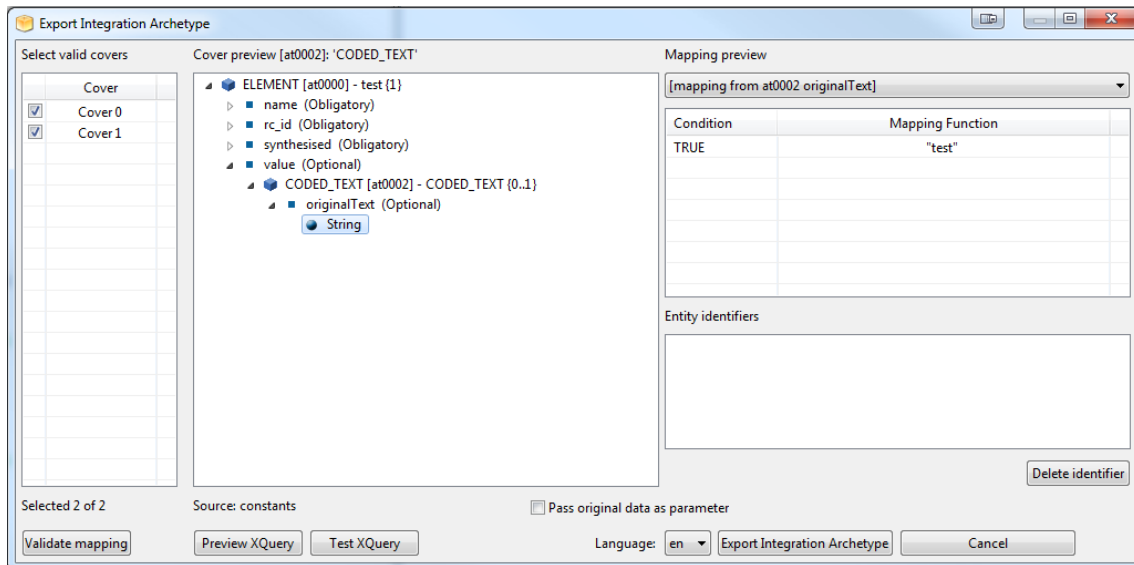


FIGURE 43 EXPORT INTEGRATION ARCHETYPE DIALOG SHOWING A SET OF COVERS FOR AN ARCHETYPE

Mapping testing

Archetype mappings can become complex, which in the end means that errors can easily occur. A set of validation and testing mechanisms were included in the mapping perspective to ensure mapping quality.

Value testing

Value correspondences can be long and complicated on their own, e.g. invoking several operations on a row or the output being dependent of a set of source values. Thus, obtaining the desired result or testing all edge cases can become difficult. Creators need an immediate connection to what they are creating (150), in this case, they need to be able to check the mapping expression as they are building it. To ease this, a functionality to test the mapping expressions in real time was included in the tool. The process provides the user with a field for every path to the source. Values are evaluated as the user types and the result is previewed. In case an error is thrown (e.g. division by zero, or trying to make an illegal type cast) the user is informed so the mapping expression can be fixed.

Transformation validation

For the users, reviewing the output transformation program in XQuery can be complicated. In order to test if the program works as intended a way of testing the XQuery with a given XML was added to the export dialog. This process executes the transformation program and returns a set of XML output instances. For selected reference models (namely ISO13606, openEHR, and HL7 CDA) an additional HTML view is shown. These HTML view is the result of applying an XSLT transformation to each one of the XML outputs. This XSLT files usually are provided by

the standards bodies themselves. Figure 44 shows an example of a HL7 CDA transformation program test in HTML.

The screenshot shows a web browser window titled 'XQuery test' displaying the output of an HL7 CDA transformation program. The output is rendered in HTML and includes the following information:

Patient:	Henry Levin	Patient-ID:	123456
Contact details:			
DOB:	February 12, 1944	AdministrativeGender:	Male
Attending Physician:			
	Created on:	April 15, 2011	

Clinical Document

History of Present Illness

Henry Levin, the 7th is a 67 year old male referred for further asthma management. Onset of asthma in his teens. He was hospitalized twice last year, and already twice this year. He has not been able to be weaned off steroids for the past several months.

Author:	Gregory House on April 15, 2011
Informant:	
Performer:	
Supporting author :	
Payer:	

At the bottom of the browser window, there are buttons for 'Save XML' and 'Copy to clipboard'.

FIGURE 44 HL7 CDA TRANSFORMATION PROGRAM TEST OUTPUT

Branch validation

In order to generate the transformation program, all the mandatory parts must have an assigned value correspondence. In some cases, this means that the only way of testing a transformation program is defining all mappings contained in mandatory paths first (i.e. mappings where the archetype path has all required types and mandatory attributes). However, if the transformation is not correct, it can become hard to fix it as all branches have their own mapping. In order to provide an easy way to test this, the ability to test a single branch of the archetype was included. This process prunes the comprehensive archetype of all the branches except the selected one and generates the transformation program. The XQuery generated this way can also be tested in order to see which part of the mapping needs to be changed.

Cover mapping validation

In addition to all the above validation processes, the export dialog also includes an automatic mapping validation. This process checks the cover archetypes and returns the errors found in them. It detects errors such as wrong hierarchy of object builders, wrong return types for a given function, or having mappings from more than a single data source in a single cover.

3.8. Validation

3.8.1. Technical Evaluation

The first validation addresses the general mapping capabilities of LinkEHR. For this purpose, we used the STBenchmark, a benchmark for evaluating mapping systems (151)¹ At the time of writing this document STBenchmark describes 17 basic mapping scenarios that are commonly used in practice and therefore should be supported by mapping systems. This means that the user should obtain from the mapping specification the desired executable code (in our case an XQuery script) without having to modify the executable code. Each scenario contains a source and target schema expressed as an XML Schema, an instance of the source schema, and a visual and textual description of the scenario.

In our validation, we represented the target schemas as archetypes and source schemas as common XML data sources. We tried all the mapping scenarios whose target schema could be modeled as an archetype. Two scenarios could not be tested: self-join and order. The former due to the presence of foreign keys in the target schema since foreign keys are not supported by archetypes. The latter could not be tested due to limitations in the expressive power of LinkEHR mapping language. In the order scenario only a subset of the source records must be copied to the target, e.g. the 5 first records. The other 15 scenarios could be tested successfully. The results were positive; the expected target instances were generated. Although our XQuery scripts were more verbose, mainly due to our grouping semantics that requires creating a flat view of data that is then nested according to the archetype structure.

3.8.2. Evaluation in real settings

The evaluation study was to use the platform in two real setting. The first one was a project for medicines reconciliation between primary health and hospital care. The second one demonstrates how Clinical Decision Support Systems (CDSS) and EHR interoperability could be improved by the use of archetypes and mappings.

The medicines reconciliation project was carried out in the Hospital of Fuenlabrada (Spain). The objective of this project was to obtain and evaluate a complete medication list of patients regardless of where the medication came from (primary or hospital care). Obtaining an up-to date and complete medication will avoid errors such as medication omission, duplication, dosing errors, and drug interactions. For the solution, a set of ISO13606 patient summary and medication archetypes were developed using the specifications from epSOS European Project (68) and openEHR Clinical Knowledge Manager (CKM) (16) archetypes. Archetypes were

¹ Benchmark is available at <http://db.disi.unitn.eu/pages/stbenchmark/>.

validated by a clinical team from the hospital composed by the Medical Director, the head of Pharmacy Service and other clinical and technical staff. Archetypes were mapped to the different data sources by using LinkEHR. The mapping was used for the generation of XQuery scripts that were deployed in both hospital and primary care that is used to generate normalized data from legacy systems.

Hospital EHR was upgraded to include a new tab containing the patient summary. With this new view, clinicians have access to the full medication list including data from both the hospital and primary care. Figure 45 shows the organization of the platform. The system is currently being used by over 430 physicians and 600 nurses and has access to the patient summary of more than 230,000 patients. This project was awarded with the Spanish Ministry of Health quality award, in transparency category.

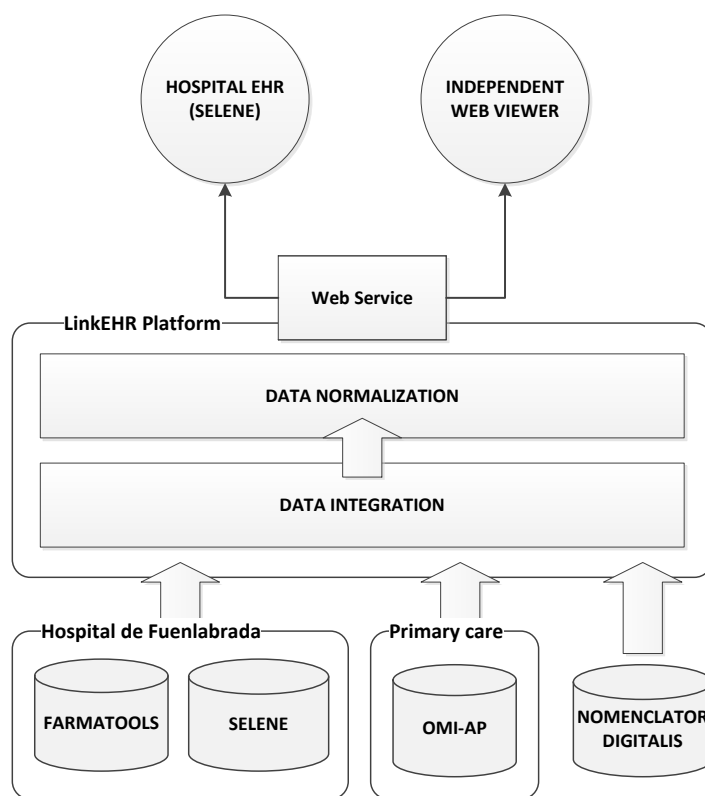


FIGURE 45 PLATFORM FOR THE MEDICATION RECONCILIATION PROJECT

For the second project, archetypes and mappings at different levels of abstraction were used in order to obtain if a patient was eligible for a given cancer clinical trial. In our use case, a patient will not be eligible for the clinical trial if it has a severe comorbidity, has (in his history or family history) any colorectal adenoma, colorectal cancer, colorectal polyposis, any inflammatory bowel disease, any lynch syndrome, or has had any total colectomy.

For this project, the archetype 'openEHR-EHR-EVALUATION.problem.v1' from CKM was specialized to include information about the presence or absence of a problem and an associated score for the comorbidity (called 'openEHR-EHR-EVALUATION.problem-DS.v1'). This specialized archetype was then specialized into a series of archetypes for each one of the identified concepts that were detected on our case study dealing with the determination of patient eligibility in a clinical trial. Mappings are created from a given level to the levels below him, i.e. level 1 archetypes are mapped to the health summary archetype, level 2 archetypes are mapped to level 1 archetypes and level 3 archetypes are mapped to level 2 and 1 archetypes. Figure 46 shows the dependences for the mappings of each level.

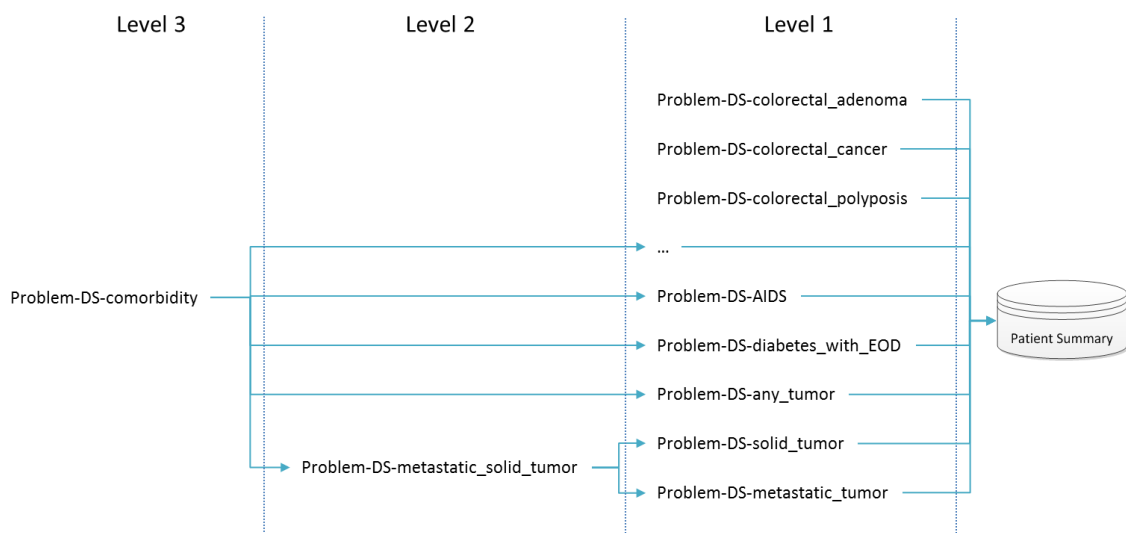


FIGURE 46 GRAPHICAL REPRESENTATION OF ARCHETYPE MAPPING DEPENDENCIES

This project makes heavy use of advanced mapping functions, such as grouping or terminology (in this use case, based in SNOMED-CT). Table 7 shows an example of a mapping in a readable form to know if a patient has a metastatic tumor (SNOMED-CT code 128462008). Also, some mappings take into account more than one archetype as a source (e.g. a metastatic solid tumor is present if both a metastatic tumor and a solid tumor are present). This example can be seen in Table 8.

<i>Condition</i>	<i>Mapping</i>
<code>(@count(summary/problems/problem, @in(summary/problems/problem/code, @descendants("128462008")))>0</code>	TRUE
TRUE	FALSE

TABLE 7 ATTRIBUTE MAPPING FOR THE PRESENCE/ABSENCE OF A METASTATIC TUMOR IN A LEVEL 1 ARCHETYPE

<i>Condition</i>	<i>Mapping</i>
<code>(Evaluation_problem_DS_metastatic_tumor_v1/structure/present = TRUE) AND (Evaluation_problem_DS_solid_tumor_v1/structure/present = TRUE)</code>	TRUE
TRUE	FALSE

TABLE 8 ATTRIBUTE MAPPING FOR THE PRESENCE/ABSENCE OF A METASTATIC SOLID TUMOR FROM TWO DIFFERENT ARCHETYPES

Output values of this process can then be used directly in the different steps of the clinical guideline.

3.9. Conclusions

This chapter deals with one of the main problems when adopting dual model EHR standards: how to transform existing clinical data to meet the data structures and constraints defined by reference models and archetypes. We face a problem known in the literature as the data exchange. Data exchange at the schema level requires an explicit representation of how the source schema (legacy data schema/archetype) and target schema (archetype) are related to each other; these explicit representations are called mappings. The effort required to create and manage such data transformations is considerable. This is even more complex in the case of archetypes, since they generally define highly nested complex data structures and model generic concepts without any regard to the internal structure of the EHR systems. Our solution separates the specification of the relationships between the schemas from the implementation of the actual transformation. The standardization process comprises four main tasks: i) generation of an XML view of the local EHRs; ii) mapping specification between source and target schemas; iii) compilation of the mapping specification into an executable program; and iv) execution of the resulting program over the source instances.

We have presented a set of integrated innovative tools to help current systems to achieve semantic interoperability by normalizing data based on clear semantically-rich clinical models. We provide methods and tools that help in automating and managing the problem of EHR data normalization. We have provided methodologies to define mappings between relational, XML, and archetypes. We deal with the complex constraints in the target such as cardinality, occurrences, and existence. We have also provided a wide range of transformation functions, including grouping and terminology functions.

We have also presented the mapping capabilities of the LinkEHR platform. The mapping module of LinkEHR is a visual programming environment for defining and managing declarative mappings and from them generating and validating the transformation scripts. This approach brings about one important advantage for users since the declarative specification is independent of the logical design of archetypes and data sources. Therefore, users do not have to specify the grouping semantics (when entities shall be nested inside other entities) or how attributes involved in a mapping are related to one another, for instance by means of parent-child relationships (in the case of XML schemas). In contrast the generation of a correct transformation scripts becomes a more difficult problem since the associations among the attributes must be discovered in an automatic way and grouping, i.e. the creation and nesting elements of the target, must be inferred from simple value correspondences. Although the interpretation of the mapping specification must cover all correspondences there usually are several alternatives, the deduction of semantics indented by the user is generally a matter of heuristics. For this reason, apart from value correspondences, LinkEHR provides more complex mappings constructs that allow users to customize the grouping semantics. Nevertheless, default semantics for inferring grouping is provided. It is based on the clinical context of data, i.e. data that share the same clinical context are grouped together.

LinkEHR allows the utilization of archetypes for upgrading already deployed systems in order to make them compatible with an EHR standard. The overall objective is to maintain in-production systems and applications without any changes while providing a mean for publication of clinical information in the form of standardized EHR extracts, hiding technical details, location, and heterogeneity of data repositories. Therefore, archetypes could be used as a semantic layer over the underlying databases associating them with domain specific concepts and therefore upgrading the semantics of the data they hold.

In the two described validation scenarios the mapping capabilities of LinkEHR were enough to generate normalized extracts with the intended semantics in a short period of time. As a

performance indicator, we considered the response time for serving a patient summary request. Transformations of source data for a single patient into an ISO/CEN EN13606 XML documents using the generated XQuery script took a fraction of a second, which was found to be insignificant with respect to the data-retrieving time. These scenarios also show that the use of these tools needs the involvement of the clinical users, which are the ones who can precisely describe their requirements. Archetypes are a suitable mechanism to improve communication between clinicians and technical staff. In our experience, even with dedicated tools the involvement of clinicians it is still difficult, probably due to the fact that clinical users still need to be aware of the dual model architecture and their implications. In order to involve clinicians, additional actions must be taken: On the one hand, editors can be improved not only to hide the complexity of the underlying models, but also provide mechanisms with higher level of abstraction (such as step by step wizards) to allow users to use the tool right away. On the other hand, in order to bring the archetypes to their original systems, archetypes must be derived into materials and artifacts, such as forms, excel spreadsheets, or mindmaps, that can be understood by clinicians and the technical staff already present in the organizations.

Once the information and models are correctly represented, new research areas open up. Areas such as application generation, semantic querying based on archetypes, clinical research, or advanced Clinical Decision Support Systems (CDSS) would benefit from having big quantities of normalized data based on formal models with clear semantics.

Chapter 4.

Archetypes for generation, validation, and use in EHR Systems

An updated version of this chapter was published as:

Diego Boscá Tomás; Jose Alberto Maldonado Segura; David Moner Cano; Montserrat Robles Viejo. Automatic generation of computable implementation guides from clinical information models. *Journal of Biomedical Informatics*. 55 - 1, pp. 143 - 152. 2015 DOI: 10.1016/j.jbi.2015.04.002

Abstract

Clinical information models are increasingly used to describe the contents of Electronic Health Records. Implementation guides are a common specification mechanism used to define such models. They contain, among other reference materials, all the constraints and rules that clinical information must obey. However, these implementation guides typically are human-readable, and thus cannot be processed by computers. As a consequence, they must be reinterpreted and transformed manually into an executable language such as Schematron or Object Constraint Language (OCL). This task can be difficult and error prone due to the big gap between both representations. The challenge is to develop a methodology for the specification of implementation guides in such a way that humans can read and understand easily and at the same time can be processed by computers. In this paper, we propose and describe a novel methodology that uses archetypes as basis for generation of implementation guides. We use archetypes to generate formal rules expressed in Natural Rule Language (NRL) and other reference materials usually included in implementation guides such as sample XML instances. We also generate Schematron rules from NRL rules to be used for the validation of data instances. We have implemented these methods in LinkEHR, an archetype editing platform, and exemplify our approach by generating NRL rules and implementation guides from EN ISO 13606, openEHR, and HL7 CDA archetypes.

Keywords: Archetype, Natural Rule Language, Implementation Guide, Data Validation, Clinical Information Model.

4.1. Introduction

Capturing requirements in the clinical domain is a difficult task (152). Traditional requirements capture methodologies fail due to the continuous evolution of clinical knowledge, the different vocabularies of clinicians and implementers, and the implicit definition of domain concepts (3). Typically clinicians rely on non-formal approaches (such as Excel or Word files) to document

their domain requirements. This kind of approach is not suitable for cooperative and long term use as it is prone to errors and version control problems. In order to solve these problems several methodologies have been proposed.

Templates are the mechanism used by HL7 CDA (153) for the specification of clinical information models. In spite of not being computable, CDA Implementation guides are the most common way for the specification of such templates in an understandable way. They usually include an introductory section describing purpose, scope, intended audience, conventions used in the guide, and separated sections for each kind of CDA components (mainly document, section, and clinical statement templates). Each one of these sections contains all the relevant templates for a given clinical model. For each template, a template identifier, a description, a set of constraints over the attributes of a given CDA component, and an XML example are provided. The implementation guide is usually completed with terminological value sets and bibliographic references. Implementation Guides play a central role in HL7 world. As an example, they have been adopted for the definition of the Consolidated CDA (C-CDA) Templates (154), which are being used to help providers to meet the applicable Meaningful Use objectives (155). However, the interpretation of the constraints in an implementation guide may differ from person to person (156), therefore limiting semantic interoperability.

Another type of resource for the specification of clinical domain requirements are archetypes. Archetypes are a key part of the dual model approach on which the EN ISO 13606 norm (12) and openEHR specification (14) are based. The dual model approach is a recent paradigm for the specification of EHR Architectures (EHRA). It distinguishes two models: the Reference Model (RM) and archetype model. In a broad sense, a reference model is an abstract representation of the generic and stable entities and relationships of a given domain. It is designed to provide a basis for the development of more concrete models and implementations. In the domain of Electronic Health Records (EHR), a reference model defines the framework for describing all EHR entries or clinical statements, the way how they are aggregated, and the context information needed to meet ethical, legal and provenance requirements. The generality of the reference model is completed by the particularity of archetypes. Archetypes are detailed and domain-specific definitions of clinical concepts in the form of structured and constrained combinations of the entities of the reference model. Archetypes may logically include other archetypes, and can be specialized to better fit the specific requirements of each use case. They can be bound to clinical terminologies and

ontologies to semantically describe the elements of information. What is important here is that for each domain concept, a definition can be developed in terms of constraints on the reference model entities. Each domain concept is also given an archetype node identifier (following the 'atNNNN' pattern where N stands for a digit) and a textual label. ADL (Archetype Definition Language) (29) is a formal language developed by openEHR for expressing archetypes that has been adopted by EN ISO 13606 standard. Even if archetypes are based on a formal language (ADL) understandable by computers, users still need specific tools and knowledge of the underlying reference model to define and understand the clinical models completely.

To allow users unfamiliar with the archetype methodology or a particular reference model to understand clinical models without using specific tools, a formal document similar to the implementation guides is required. What we need is formal document that has at least the same expressiveness than an archetype and at the same time is easily understandable even by non-technical users.

Our proposal, as described in Figure 47, aims to achieve the automatic generation of computable implementation guides from archetypes. Our objectives are twofold:

1. To generate implementation guides that can be used in the development of computer systems by IT technical staff. For this purpose, we use archetype texts, descriptions, and terminology bindings. We also include other automatically generated materials such as sample XML instances and validators.
2. To document archetypes or templates in order to ease their understanding by health professionals without the need of specific tools. For this purpose, we transform the potentially complex archetype constraints into English-like rules. This is achieved by the use of Natural Rules Language (NRL) (100). We also include additional reference materials in the implementation guide, such as a mindmaps, value sets and bibliographic references.

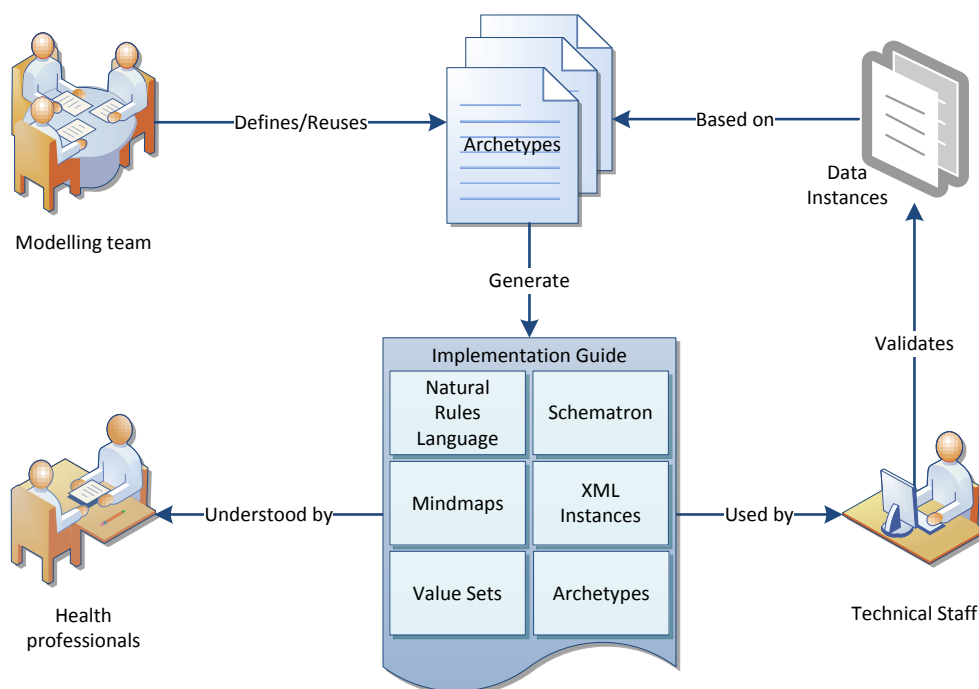


FIGURE 47 – PROPOSED ARCHITECTURE FOR THE GENERATION OF IMPLEMENTATION GUIDES FROM ARCHETYPES

We will exemplify our approach by generating implementation guides from an EN ISO 13606 archetype, openEHR archetypes from the Clinical Knowledge Manager (CKM) (157) and HL7 CDA archetypes from the Genetic Testing Report.

4.2. Background and related work

There exists a wide range of formal rule languages for the definition of constraints on data. One of the most known is the Object Constraint Language (OCL) (30), an OMG (89) standard for the definition of rules over UML models (95). There are also languages for defining Horn-like rules for the Ontology Web Language (OWL) (96), such as Semantic Web Rule Language (SWRL) (97) or RuleML (98). The widespread use of rules, formal or not, has caused the creation of proposals, like the W3C Rule Interchange Format (RIF) (99), for the exchange of rules between different rules languages. The main disadvantage with most rule languages is that rules are not easily understood by non-technical staff. To solve this problem, some rule languages with natural language-like syntax have been proposed. Two main examples are Natural Rule Language (100) and Attempto Controlled English (101). Each one of them addresses the problem of natural language rules representation from a different perspective.

Natural Rule Language (NRL) is a formal language for specifying constraints and rules in a human readable way. The main feature of this language is the capacity of defining constraints in a way that facilitates their understanding by non-technical people. Moreover, NRL also defines an extension to deal with actions, such as the creation or deletion of objects, or setting values when certain conditions are met. Although we will not use this extension, it could be used to complete the rules with actions, for instance to calculate derived values. To the authors' knowledge there is only one prior use of NRL in the clinical domain, concretely for the representation of clinical practice guidelines and its evaluation in a real world case (102). Rules drawn from a hypertension guideline were translated into NRL in order to be validated by clinicians and subsequently they were transformed into OCL and finally used in the system. The NRL rules were generated by hand which can be a time-consuming task.

Attempto Controlled English (ACE) (101) is a controlled natural language, which means that it is a subset of Standard English with a restricted syntax. ACE can be translated into other languages, such as RuleML, OWL, or SWRL. The meaning of words in ACE is not predefined and must be defined in an existing ontology or in additional ACE sentences. Although ACE has been in use for more than ten years, to the authors' knowledge it only has been used once applied to the clinical domain (103), specifically for clinical guidelines readability. In this work, rules from a pediatric clinical guideline were expressed in ACE, although they were not applied to real data.

There also exist formal languages for the validation of XML documents such as Schematron, DTD or XML Schema. Schematron (104) is a rule-based validation language for making assertions about patterns in XML trees that is an ISO norm since 2006. Since it is a path based validation language, Schematron can express constraints that neither XML Schema nor DTD can express. Each rule can be associated with a descriptive text of the type of error or warning encountered. Schematron plays a key role on current CDA implementations as Schematron rules are typically attached to implementation guides alongside sample XML instances. It has been proved that Schematron rules can be directly generated from NRL rules (105) as well as from archetypes (106). Advanced features of archetype methodology, such as reuse of internal or external types can be also reproduced with Schematron.

The generation of reference materials from formal model definitions is also one of the goals of other initiatives such as Open Health Tools (OHT) Model-Driven Health Tools (MDHT) Project (94). MDHT is an open source effort for the promotion of shared artifacts between related standards and the creation of modelling tools for their seamless integration. The project is

supported by the US Veteran's Health Administration (VHA), IBM, and the US Office of the National Coordinator (ONC). Their original focus was to develop HL7v3 specifications via UML, but they later moved to work in the specification of HL7 CDA Implementation Guides. They have provided models and reference implementations for several HL7 C-CDA Implementation Guides. They are planning to support other standards besides HL7 CDA, for instance by using UML for the specification of archetypes. A UML profile (Archetype Modeling Language, AML) has been proposed to OMG to deal with the specific requirements of the archetype modeling. MDHT is also working in the generation of Schematron for XML instance validation.

4.3. Material and Methods

4.3.1. LinkEHR platform

LinkEHR® (158) is software tool for the integration and normalization of health data (134). LinkEHR employs archetypes for both the semantic description of the clinical concepts to be shared and the transformation of existing clinical information into standardized EHR extracts. It comprises two main modules that allow (i) the editing of archetypes based on different RMs (several RMs have been tested successfully: EN ISO 13606, openEHR, HL7 CDA, CDISC ODM and ASTM CCR); and (ii) the specification of declarative mappings between archetypes and data sources, and from these mappings the automatic generation of XQuery scripts which translate source data into archetype compliant XML documents. In our scenario, a crucial tool is the LinkEHR archetype editor. During archetype editing, the tool provides support to ensure that the archetype being edited is valid with respect to the reference model (and parent archetype, if any), e.g. by showing the valid elements at any point. When the user wishes to add a new entity to an archetype the editor displays the valid entities and the user must select one of them. All the functionalities described in this paper have been added to LinkEHR archetype editor.

4.3.2. Generation of implementation guides from archetypes

In order to generate a complete implementation guide we produce five different reference materials from archetypes: Natural language rules, mindmap, XML instances, Schematron rules, and value set tables. Although typical implementation guides are designed to be printed, they can be improved with interactive elements such as sample data entry forms or mindmaps that can be rendered on a computer screen. Reference materials used for the creation of the implementation guide depend on its final purpose and use, e.g. the inclusion of mindmaps may be very useful in an interactive implementation guide, but it may be not as useful in a printed one.

As stated before, an implementation guide contains an introductory section, separated sections for document, section, and clinical statements templates, and a final section with the value sets used in the guide. All these sections are generated by combining the archetype definitions with the generated reference materials.

Introductory section is generated from the archetype metadata, which includes the purpose, keywords, intended use, references, etc. In archetypes, the entities of the clinical model can be attached with a text label, a description, and a terminology binding. All this information is organized into their own subsection of the implementation guide. The text label of each entity becomes the template name of that entity and the description and terminology binding become the template description. Archetype entities also include information about their occurrences, cardinality and existence that are used to control what will be generated. For instance, a mandatory entity must appear in XML instances and must be checked to exist with a specific rule in Schematron, but could be hidden in a mindmap or form if it is not considered interesting to the final user. The last section of implementation guides are value set tables. These tables specify a set of codes drawn from one or more code systems. As archetypes already contain this kind of information in the constraint binding part of the ontology section, we generate all the tables directly from there. This table is built by querying a terminology server to obtain all codes from a given subset and all the codes descriptions. Finally, we also generate a Table of Contents to easily navigate the implementation guide.

4.3.3. Generation of NRL rules

In the archetype methodology, archetype entities are created by constraining a reference model type (3), concretely by constraining the values, structure, and/or terminology bindings. New entities include implicitly all the constraints imposed by the reference model type that have not been explicitly narrowed in the archetype. This supposition is consistent with the object-oriented paradigm, where attributes and methods of a superclass are automatically inherited by all its subclasses. If we were to create rules directly over the reference model types, they would not be easily understandable because rules would refer to a given type and a node identifier (e.g. "at least one ENTRY where archetype_id='at0000' exists"). This is the reason why we create variables in NRL using the textual labels attached to the archetype entities as variable names. When no label is defined (e.g. a data type) a label is derived from their parent entity. If there is a label clash, the entity identifier is also used for the generation of the readable name. The expression that defines the variable is built using entities identifiers, i.e. the archetype node identifier if we are using an archetype-based standard or the

templateId if using HL7 CDA as reference model. As an example, the above rule is rewritten as “at least one BloodPressure exists” which uses “BloodPressure” variable defined as “BloodPressure is the ENTRY where archetype_id='at0000'”. We exemplify this approach with the generation of NRL rules from a blood pressure EN ISO 13606 archetype shown in Figure 48. In Figure 49 we show how variables for each one of the reference model types are declared and reused in other rules. The readable label is used as a variable that will be applied when node identifier is found in data.

```

ENTRY[at0000] occurrences matches {1..1} matches { -- Blood pressure
  items existence matches {0..1} cardinality matches {1..1; ordered; unique} matches {
    ELEMENT[at0001] occurrences matches {1..1} matches { -- Systolic
      value existence matches {1..1} matches {
        PQ[at0005] occurrences matches {1..1} matches { -- Systolic measure
          units existence matches {1..1} matches {
            CS[at0009] occurrences matches {1..1} matches { -- CS
              codeValue existence matches {0..1} matches {"mm[Hg]"}
              codingSchemeName existence matches {0..1} matches {"UCUM"}
            }
          }
        }
      value existence matches {1..1} matches {0.0..<1000.0}
    }
  }
}

ELEMENT[at0002] occurrences matches {1..1} matches { -- Diastolic
  value existence matches {1..1} matches {
    PQ[at0006] occurrences matches {1..1} matches { -- Diastolic measure
      units existence matches {1..1} matches {
        CS[at0010] occurrences matches {1..1} matches { -- CS
          codeValue existence matches {0..1} matches {"mm[Hg]"}
          codingSchemeName existence matches {0..1} matches {"UCUM"}
        }
      }
    value existence matches {1..1} matches {0.0..<1000.0}
  }
}

ELEMENT[at0003] occurrences matches {0..1} matches { -- Mean arterial pressure
  value existence matches {0..1} matches {
    PQ[at0007] occurrences matches {1..1} matches { -- Mean pressure measure
      units existence matches {1..1} matches {
        CS[at0011] occurrences matches {1..1} matches { -- CS
          codeValue existence matches {0..1} matches {"mm[Hg]"}
          codingSchemeName existence matches {0..1} matches {"UCUM"}
        }
      }
    value existence matches {1..1} matches {0.0..750.0}
  }
}

ELEMENT[at0004] occurrences matches {0..1} matches { -- Position
  value existence matches {0..1} matches {
    SIMPLE_TEXT[at0012] occurrences matches {0..1} matches { -- Position
      originalText existence matches {0..1} matches {"Standing","Sitting","Reclining","Lying"}
    }
  }
}

```

FIGURE 48 BLOOD PRESSURE EN ISO 13606 SAMPLE ARCHETYPE

"BloodPressure" is the ENTRY where archetype_id = 'at0000'
 "Systolic" is the ELEMENT where archetype_id = 'at0001'
 "Diastolic" is the ELEMENT where archetype_id = 'at0002'
 "MeanArterialPressure" is the ELEMENT where archetype_id = 'at0003'
 "Position" is the ELEMENT where archetype_id = 'at0004'

FIGURE 49 DECLARATION OF VARIABLES IN NRL TO ALLOW THE GENERATION OF READABLE RULES


```

Context: BloodPressureMeasurement

Validation Rule "Children occurrences of 'parts' attribute from Blood Pressure"
exactly one of parts is a kind of Systolic and
exactly one of parts is a kind of Diastolic and
at most one of parts is a kind of MeanArterialPressure and
at most one of parts is a kind of Position

```

FIGURE 50 SAMPLE RULE FOR OCCURRENCES CONSTRAINT USING THE READABLE VARIABLES

Once we have created a variable for each archetype entity we are ready to create rules for the archetype constraints. We create rules for each one of the constraints defined on the archetype, such as entity occurrences (as shown in Figure 50), attributes existence and cardinality (shown in Figure 51), and on data values (shown in Figure 52). Each rule has a readable name to identify it. We can also generate comments to help even more with the understanding of the rules. Comments are generated from entity constraints. Any part of a rule line starting by '--' is considered a comment. For instance, in Figure 51 the rule "Cardinality of 'parts' attribute from BloodPressureMeasurement" has an additional comment stating the cardinality with an array notation, which can be easier to understand for people used to work with archetypes.

```

-- Cardinality [2..4]
Context: BloodPressureMeasurement

Validation Rule "Cardinality of 'parts' attribute from BloodPressureMeasurement"
at least two parts exist and at most four parts exist

-- Existence of structure_type is mandatory
Context: BloodPressureMeasurement

Validation Rule "Existence of 'structure_type' attribute from BloodPressureMeasurement"
structure_type is present

```

FIGURE 51 SAMPLE RULES FOR CHECKING CARDINALITY AND EXISTENCE OF AN ATTRIBUTE

```
Context: CSFromDiastolicMeasure

Validation Rule "Constraint of 'codeValue' attribute from CSFromDiastolicMeasure"
The codeValue is equal to 'mm[Hg]'

Context: DiastolicMeasure

Validation Rule "Constraint of 'value' attribute from DiastolicMeasure"
value is greater than or equal to 0.0 and value is less than 1000.0

Context: Position

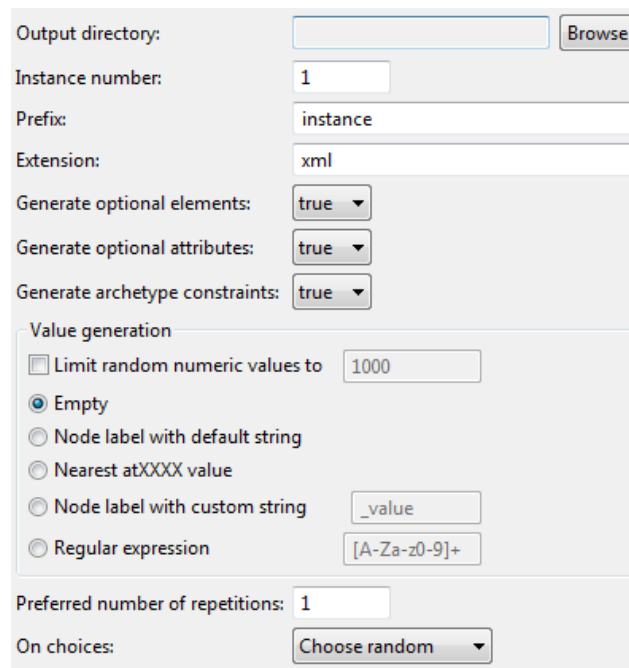
Validation Rule "Constraint of 'originalText' attribute from Position"
The originalText is one of 'Standing', 'Sitting', 'Reclining', 'Lying'
```

FIGURE 52 SAMPLE RULES FOR CHECKING DIFFERENT KINDS OF DATA VALUE CONSTRAINTS

The set of automatically generated rules can be extended with additional user-defined natural language rules, for instance to express constraints that are not supported by ADL, e.g. constraints such as “Mean blood pressure is calculated by adding to the systolic pressure two times the diastolic pressure and dividing the result by three” that involves more than one entity from the archetype.

4.3.4. XML instances Generation

For the generation of XML instances we use LinkEHR mapping capabilities in order to generate valid XML sample instances compliant with the archetype and the underlying reference model. As stated before, in archetypes only entities (classes and attributes) of the reference model which are actually constrained need to appear in the archetype definition. It is supposed that the constraints defined in the underlying reference model are implicit constraints for the derived archetypes. As a consequence, it is necessary to complete (“merge”) the archetype with the constraints defined in the underlying reference model in order to generate complete XML data instances. A constant mapping, i.e. a mapping function that assigns a constant value, is automatically generated for each leaf node of this “merged” archetype. Using this constant mapping, we generate a XQuery transformation program on the fly whose output will be an XML instance compliant with the original archetype and the underlying reference model (134). The instance generation process can be tuned by several parameters, such as the inclusion of optional attributes, selection of alternatives, or the contents and ranges of primitive types. The aforementioned parameters can be set in LinkEHR Editor as shown in Figure 53.



Output directory: Browse

Instance number:

Prefix:

Extension:

Generate optional elements: true

Generate optional attributes: true

Generate archetype constraints: true

Value generation

Limit random numeric values to

Empty

Node label with default string

Nearest atXXXX value

Node label with custom string

Regular expression

Preferred number of repetitions:

On choices:

FIGURE 53 OPTIONS FOR XML INSTANCE GENERATION IN LINKEHR

4.3.5. Schematron Generation

As stated before, NRL rules can be translated to Schematron for the validation of XML instances with respect to archetypes and reference models. Schematron rules are based on path conditions that specify where the assertion must be tested. The process traverses the entities in the archetype recursively and generates a rule for each entity with an assertion for each one of the tests (namely tests for occurrences, cardinality, existence, and values). In Figure 54 we show the equivalent Schematron rule to the NRL rule described in Figure 50. As it can be observed, Schematron rules are by far less understandable than NRL rules.

In addition to the Schematron rules generated from the explicit archetype constraints, we also generate optional Schematron rules for checking the implicit constraints, i.e. the constraints coming from the reference model. This is necessary for instance to assure that an archetype type does not contain attributes that are not allowed by the reference model or that the type of an unconstrained entity is one of the types allowed by the reference model.

```

<rule context="EN13606:ENTRY/EN13606:items">
<!-- Rule for archetype path /items[Blood pressure measurement] -->
<!-- Occurrences for the children of attribute 'parts' -->
  <assert test="count(EN13606:parts[archetype_id='at0001'])=1"/>
  <assert test="count(EN13606:parts[archetype_id='at0002'])=1"/>
  <assert test="count(EN13606:parts[archetype_id='at0003'])>=0 and
count(EN13606:parts[archetype_id='at0003'])<=1"/>
  <assert test="count(EN13606:parts[archetype_id='at0004'])>=0 and
count(EN13606:parts[archetype_id='at0004'])<=1"/>
  <assert test="EN13606:items[@xsi:type='CLUSTER']"/>
</rule>

```

FIGURE 54 SCHEMATRON RULE FOR BLOOD PRESSURE MEASUREMENT OCCURRENCES

4.3.6. Generation of Additional Reference Materials

In addition to the aforementioned reference materials, we generate other materials that have not been traditionally included in implementation guides such as mindmaps or sample input forms. These artefacts are interactive, and thus they lose part of their potential usefulness in printed implementation guides. However they can be really useful when the implementation guide is displayed on a computer screen. Mindmaps mimic the archetype structure but omit non-clinical parts to make it easier for clinicians to understand the clinical meaning. Forms are generated in a similar way, but their transformation from archetypes is reference model dependent. Currently we are only able to generate sample forms for EN ISO 13606 archetypes.

4.4. Results

We have implemented our solution in several software modules in Java, each one producing a different type of reference material from an archetype expressed in ADL. Both mindmap and Schematron outputs are XML representations that are generated from the archetype definition. NRL rules are also generated from the archetype following the process described above. Sample instances are produced by generating a constant mapping and creating from it an XQuery whose output is the data instance. Finally, Sample forms are created by applying an XSLT transformation to the XML representation of archetypes and are displayed in a web browser.

In addition to the previous modules, another module was implemented to combine all the output into a complete implementation guide expressed in HTML. Mindmap interactive visualization is included in the HTML page using an Adobe Flash plugin. We have defined different CSS style sheets to render on-screen and print views. Printed views can also be generated as PDF files to ease their distribution. Regarding terminology bindings, we employed

Indizen IT Server (159) to retrieve the concept text descriptions and get all codes in a terminology subsets.

The load and generation time, i.e. the time to read and parse the archetype and the time to generate the implementation guide respectively, closely reflects the archetype size in terms of number of constraints as would be expected. On simple archetype, the generation time is almost negligible while for large archetypes it can take as much as several seconds. In any case, the time is negligible when compared with the time required to generate an implementation guide manually.

All the developed modules have been included in the LinkEHR platform in order to provide different export formats for archetypes. Each module used both a set of configuration parameters and documentation about the reference model being to control the generation process and output appearance of the corresponding material (XML instances, Schematron rules, NRL rules, mindmaps, or sample input forms). In the case of implementation guides, this set of parameters is predefined in such a way that the output resembles a real implementation guide.

To exemplify the generation of implementation guides, we show two different examples. In the first one we automatically generated implementation guides from a subset of CKM archetypes created in (57) with improved terminology bindings. In the second example, we generated an implementation guide from an HL7 CDA archetype (160). The complete examples can be found in the supplementary material.

The first example exemplifies all the generated subsections included into a section of the implementation guide: Description, terminology binding (looking up the terminology code in an external terminology), a set of readable rules, an XML sample instance, and the Schematron validation for this specific entity. Figure 55 shows an excerpt the output implementation guide subsection for Heart rate entity from the Apgar score archetype. This contains the archetype entity text as section header, a description of the entity, their terminology binding (along with the text obtained from the terminology server), entity constraints stated as NRL rules, an XML example section, and a Schematron section.

5.4 Heart rate

Assessment of heart function in the new born

5.4.1 Terminology Binding

[SNOMED-CT::364075005] - 364075005[Heart rate (observable entity)]

5.4.2 Rules

Context: HeartRate

Validation Rule "Alternatives of 'value' from HeartRate"

every value is Over100BeatsPerMinute or

every value is Absent or

every value is Slow(below100BeatsPerMinute)

5.4.3 XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ELEMENT archetype_node_id="at0005">
  <name xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="DV_CODED_TEXT">
    <value>Heart rate</value>
    <defining_code xsi:type="CODE_PHRASE">
      <terminology_id xsi:type="TERMINOLOGY_ID">
        <value>Heart rate</value>
      </terminology_id>
      <code_string>Heart rate</code_string>
    </defining_code>
  </name>
  <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="DV_ORDINAL">
    <value>0</value>
    <symbol xsi:type="DV_CODED_TEXT">
      <value/>
      <defining_code xsi:type="CODE_PHRASE">
        <terminology_id xsi:type="TERMINOLOGY_ID">
          <value>local</value>
        </terminology_id>
        <code_string>at0006</code_string>
      </defining_code>
    </symbol>
  </value>
</ELEMENT>
```

5.4.4 Schematron

```
<rule context="EHR:OBSERVATION/EHR:data/EHR:events[@archetype_node_id='at0003']/EHR:data/EHR:items[@archetype_node_id='at0005']">
  <!-- Existence for attribute 'value' is optional, so no rule is created-->
  <!-- Occurrences for the children of attribute 'value' -->
```

FIGURE 55 EXCERPT OF AN AUTOMATICALLY GENERATED IMPLEMENTATION GUIDE FROM AN APGAR SCORE OPENEHR ARCHETYPE

In the second example we employed an archetype (160) created from the Genetic Testing Report (GTR) HL7 implementation guide (161). The HL7 CDA archetype contains all the data constraints defined by the GTR implementation guide. Figure 56 shows an excerpt of the original implementation guide, whereas Figure 57 shows the same excerpt represented in the automatically generated guide.

Cytogenetics Associated Observation Cells Karyotyped Count

[Observation: templateId 2.16.840.1.113883.10.20.20.2.2.3]

The ClinicalGenomicStatementCytogeneticsCellsKaryotypedCount template is a sub-template of ClinicalGenomicStatement and is used to carry the no. of cells karyotyped in a cytogenetics test.

1. **SHALL** conform to *Genomic Associated Observation* template (templateId: 2.16.840.1.113883.10.20.20.4)
2. **MAY** contain exactly one [1..1] `code/@code="55199-4" Cells karyotyped.total [#] in Blood` (CodeSystem: 2.16.840.1.113883.6.1 LOINC)
3. **SHALL** contain zero or one [0..1] `value`, where its data type is INT

FIGURE 56 CELLS KARYOTYPED COUNT FROM THE ORIGINAL GENETIC TESTING REPORT IMPLEMENTATION GUIDE

5.1 GTR Clinical Genomic Statement Cytogenetics Cells Karyotyped Count

The ClinicalGenomicStatementCytogeneticsCellsKaryotypedCount template is a sub-template of ClinicalGenomicStatement and is used to carry the no. of cells karyotyped in a cytogenetics test.

5.1.1 Rules

```
Context: GtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount
Validation Rule "Rules for GtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount"
classCode is present -- Existence of classCode is mandatory
at most one value exist -- Cardinality [0..1]
exactly one of value is INTFromGtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount
templateId is present -- Existence of templateId is mandatory
exactly one templateId exist -- Cardinality [1]
exactly one of templateId is IIFromGtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount
code is present -- Existence of code is mandatory
exactly one of code is CVFromGtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount
The classCode is equal to 'OBS'

Context: IIFromGtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount
Validation Rule "Constraint of root from IIFromGtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount"
The root is equal to '2.16.840.1.113883.10.20.20.2.2.3'

Context: CVFromGtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount
Validation Rule "Rules for CVFromGtrClinicalGenomicStatementCytogeneticsCellsKaryotypedCount"
The code is equal to '55199-4'
The codeSystemName is equal to 'LOINC'
The displayName is equal to 'Cells karyotyped total [#] in Blood'
The codeSystem is equal to '2.16.840.1.113883.6.1'
```

FIGURE 57 EXECUTABLE RULES FROM CELLS KARYOTYPED COUNT AUTOMATICALLY GENERATED FROM THE HL7 CDA GENETIC TESTING REPORT ARCHETYPE

The generated rules express exactly the same constraints as the original implementation guide, but they can be executed directly over data instances. We can express rules following two alternatives, grouping the rules by context to ease their understanding, or generating an individual rule for each kind of constraint (occurrences, existence, cardinality, etc.) to know exactly which constraint fails. In Figure 57 we have followed the first approach.

4.5. Discussion

In this paper, we emphasize on the usefulness of archetypes for the generation of implementation guides and reference materials. The generated reference materials include human readable definition of clinical models for clinicians or computable artifacts for technical staff. For instance, NRL rules facilitate the involvement of clinicians in the definition of clinical models, ensuring that the systems to be developed satisfy their requirements. At the same time, and since NRL is a formal language, the rules can be used to support the implementation of EHR systems, for instance for data validation purposes.

If we compare our approach to other initiatives dealing with the generation of derived artifacts such as MDHT, the main difference is we employ archetypes instead of UML as the formal approach to model clinical data structures. MDHT Project also generates implementation guides with alternative content structure depending on the target audience, e.g. by generating ballot documents or implementer views of an UML diagram. Our proposal aims to deal with both target audiences by creating formal, computable human-readable implementation

guides. Another important feature of our approach is it can deal with any reference model or EHR standard. Since it is based on archetypes it is possible to generate implementation guides for a wide range of EHR standards. The only requirement is to be able to define archetypes based on the information model defined by the standard. This not only includes standards that are “archetype native” such as openEHR, EN ISO 13606, or CIMI reference model, but also non-archetype based standards such as HL7 CDA, HL7 FHIR, CDISC ODM, openCDS VMR, Intermountain CEML, ASTM CCR, or MedXML MML, all of them already supported by LinkEHR archetype editor. In the case of CDA, CDA archetypes are equivalent to a template fully compliant with the HL7 Reference Information Model (RIM). Using archetypes as a basis for implementation guides generation may seem unfitting for the HL7 world. However, using archetypes over HL7 CDA model has already been proved useful in real life projects (160)(141). This approach also solves common HL7 CDA problems (162) such as extensions of the CDA standard, namespaces changes and element sequencing

It is important to notice that archetypes are multilingual, which means that the target Implementation Guide can be automatically generated in any language supported by the archetype. This is also true for the archetype terminological bindings, as long as a translation of the terminology to the target language exists. Due the fact that NRL is a controlled grammar, it is also feasible to translate the rules to different languages, and render them in the language of the user. We used NRL over ACE because one of our objectives was data validation. Formal rule validation languages, such as OCL or Schematron, are easily derived from NRL rules. The fact that NRL can be directly transformed to OCL means that we can automatically generate implementation guides with OCL rules instead of NRL rules to mimic current implementation guides. We used NRL instead of OCL in order to make these implementation guides readable to non-technical staff. ACE rules could still be used for expressing the constraints if we give priority to OWL and SWRL transformations or we want to use some kind of ontology reasoning, as ACE terms are defined in OWL. For this use case, every word in an ACE rule should be mapped to an ontology concept, which needs to be done, or at least supervised, by a domain expert. This turns the automatic rule generation process into a semiautomatic one, which is not feasible for our use case.

NRL rules can also be used alongside archetypes, as archetypes can accommodate rules to define further constraints. NRL may be used as the rules language for the Archetype Definition Language (ADL). For instance, advanced constraints like the ones provided by the upcoming ADL 2.0 (such as grouping or sorting) could be expressed with NRL rules in order to increase

the expressivity of ADL 1.4 to match up to ADL 2.0. NRL also includes an action grammar, which can be used not only to set values in the clinical model, but also for the creation of rules for data transformation.

One of the main disadvantages of using NRL rules is that the vocabulary in current HL7 CDA implementation guides differs from the one used by NRL grammar. HL7 CDA implementation guides use specific reserved words for the definition of constraints such as “SHALL”, “SHOULD”, “MAY”, etc. If needed, NRL rules could be transformed to generate rules using this particular vocabulary. Furthermore, taking a generic approach means that there will be some misalignments between an HL7 implementation guide and an automatically generated implementation guide. Misalignments are caused by the explicit generation of constraints from the archetype. It can be argued if it is preferable to check parts of the template (e.g. the `templateId` or the entity type) with explicit rules to ensure correct data instances, even if they are normally assumed for a given data instance (e.g. in Figure 56, both the “Observation” type and the `templateId` are not explicitly defined with a “SHALL” rule).

Regarding Schematron, our generation process is similar to the one described in (106) for Schematron generation from HL7 CDA archetypes, but we are able to generate Schematron for any EHR standard. Also, our generated Schematron solution can distinguish if a rule should be applied over XML elements or attributes. The final advantage of our solution is that we also provide a set of optional rules to check the implicit constraints coming from the reference model. This allows us to define different validation scenarios for the same archetype, like validating only archetype constraints, or archetype and reference model constraints.

Finally, we can extend this methodology to generate implementation guides from Clinical Practice Guidelines. There are several examples of representation of Clinical Practice Guidelines with archetypes and formal rules (163–169). Usually the information required by the clinical guide is modeled as archetypes, and rules and pathways are normally modeled in languages such as CLIPS, Drools, or PROforma. These rules are not human-readable, but as demonstrated in (102), they can be also expressed in NRL. Archetypes created by these methods can be transformed into implementation guides using our methodology, and rules and pathways transformed into NRL and then included in the resulting implementation guide. This transformation will make clinical practice guidelines suitable to be used directly in data validation and eases its understanding by computers and clinical staff alike.

4.6. Conclusion

Implementation guides are one of the most common documents for the provision of clinical specifications for particular domain. In this paper we have shown how it is possible to generate automatically from archetypes all the parts and reference materials that are usually included in implementation guides. In addition to that, other interesting materials that are not usually included such as Schematron rules, mindmaps or sample forms can also be generated for their distribution alongside implementation guides. The quality of the output implementation guide and derived reference materials is directly related to the quality and completeness of the source archetype. Missing or incomplete sections of the archetype (e.g. poor or no metadata defined, or missing terminology bindings) will cause the generation of empty or incomplete sections in the implementation guide. The quality of the resulting implementation guide provides a measurement of the quality of the source archetype.

The proposed methodology promotes the involvement of clinical staff in the modeling and validation process. Any possible misinterpretation is avoided as constraints and rules definitions can be automatically translated into formal validation rule languages that can be applied directly in the final system.

Reuse is one of the core principles of archetype methodology. When the same archetype is included in other archetypes we can reuse this generated implementation guides. This not only eases their generation, but also provides coherence between the different implementation guides that reuse the same clinical models.

The presented methodology puts the emphasis on the generation of implementation guides that humans can read and understand easily and at the same time can be processed by computers. This approach may promote the adoption of clinical information models in the development of EHR systems, thus increasing the quality of clinical data and its semantic interoperability.

Chapter 5.

Final Conclusions and Future Work

5.1. Final conclusions

Semantic interoperability has always been the Holy Grail in health informatics. Sending information that could be completely understood by the receiver is one of the biggest challenges in health informatics. This mainly requires capturing faithfully the original meaning of health data. The intrinsic complexity and variability of health data makes standardization crucial to achieve high levels of semantic interoperability. Fortunately, there is a mature body of EHR standards that cover apart from generic information models, clinical models (such as archetypes) as mechanisms to formalize both the syntax and the semantics of specific clinical concepts. The principal purpose of clinical models in general and archetypes in particular is to provide a powerful way of managing the description, creation, validation, communication and querying of EHRs. The definition and sharing of clinical models is acknowledged as a big step forward in reaching high levels of semantic interoperability.

It must be noted that eHealth interoperability standards are usually defined as documented specifications that must be brought to life by system designers and implementers. This initial implementation effort requires deep expertise, often not accessible for many organizations. Therefore, the current challenge is not the lack of standards, but its competitive implementation at a reasonable effort and return of investment for all the actors involved either clinical or technological.

This thesis intends to provide methodologies and tools to facilitate the adoption of EHR standards and archetypes as a secure mechanism to achieve higher levels of semantic interoperability for the EHR. Concretely, the main contributions of this thesis are:

1. A set of methodologies for applying the dual model approach to non-dual EHR information models and a set of tools for the definition of archetypes based on these EHR models. The definition of a set of clinical models of an already deployed system allows for their formal description. These clinical models allow taking advantage of all available archetype-based methodologies and tools for their use in current systems.
2. A methodology for the definition of mappings between legacy systems and archetypes. This archetype-based mapping methodology generates transformation programs to translate data contained in these legacy or non-legacy systems into data in any other standard or specification. The presented mapping methodology has been validated by its use in several academic and real world projects. This methodology was implemented in

LinkEHR platform. Mapping definition still needs deep knowledge of the standards and systems involved, which we try to ease as much as possible with the presented tools.

3. A methodology for the automatic generation of a set of reference materials from archetypes. The methodology was implemented in a set of tools that allow the automatic generation of each reference material. These tools can be used independently or combined to generate rich implementation guides. The creation of reference materials allows the validation of the data and workflows in the systems and the technical validation of requirements. Different validation scenarios make use of different sets of reference materials to check the systems. Validating if systems follow the requirements is eased as clinical-defined archetypes are also the ones used in by technical people in the creation of systems. This allows for a better communication of the requirements between clinical and technical people, as archetypes act as a common language.

The presented methodologies and tools ease the joint use of the three layers of semantic interoperability to achieve it (reference models, clinical models, and clinical terminologies). Assuring that the archetypes have good quality allows for the meaningful communication of legacy health data, and will allow true technology-independent, patient centric EHR.

This will make available for analysis more patient health data than ever. These big amounts of data need to be curated and processed for being able to get the most out of it. Data normalization and standardization is a basic step for measuring and increasing data quality (170). In fact, as seen in the medication conciliation use case, archetype-based normalization can improve data coherence among different systems by agreeing in a common set of archetypes. This is useful in any multicentre research project. Advanced data analysis greatly benefits from knowing the semantics of data, as do clinical research, clinical guidelines, and public health (2).

5.2. Future work

Once the barrier of integrating dual model systems with legacy health information systems has been broken, new and interesting challenges unfold.

From the data perspective, we have detected several challenges. Variability in original sources continuously provides new challenges for the transformation definition. New methodologies must be researched in order to not lose any significant clinical data from legacy systems. Our methodology assumes that all data needed for the normalization comes from a single data source, which is not always the case. The methodology must be improved in order to support

multiple source schemas. In addition to that, the generation and managing of mappings requires a considerable effort. Automatic schema matching techniques can potentially be used to detect relationships between health information standards and generate automatic mappings between clinical information models. Tree-matching algorithms are specially promising due to the structured nature of these models (171,172). These mappings will use model semantics and archetype terminology bindings, so defining precise and coherent terminology bindings must be a requirement in archetype creation.

In fact, the use of clinical vocabularies and ontologies gives a challenge on its own. The integration of ontologies, such as SNOMED CT, in queries and reasoning used in real world systems will benefit both patient and public health. The presented solution currently provides a basic support to SNOMED CT queries and subsets. More complex and powerful languages such as the SNOMED CT expression constraint language (135) must be evaluated and incorporated into our mapping language. The frontier between archetypes, clinical guidelines, and clinical terminologies and ontologies is still an open issue and must be studied.

From the reference material perspective, every reference material can benefit from terminology bindings, as the generation of implementation guides, rules, and validators can be improved by reusing these bindings. Other reference materials as the example data instances can be greatly improved to better mimic real data values and distributions, e.g. how the clinical diagnosis are distributed in a real world EHR system. Recent developments have provided ways of simulating better looking fake data instances (173) and our work can greatly benefit from them. The use of these materials helps in the improvement of data quality (106,174,175). However, presented methodologies only measure the quality of the systems limited to their compliance with standards, being impossible to calculate general compliance degrees or where are the errors typically located. Meaningful formal definitions of clinical data will allow for more advanced quality measurements, such as correctness, completeness, multi-source stability, predictive value, and temporal stability (176,177).

The archetype-based mapping mechanism over legacy data allows us to obtain big quantities of patient data based on archetypes. We can take advantage of having meaningful patient historic data by generating research and public health repositories. These new repositories should be able to query data based on archetype information. Big data and parallelization issues must be solved in order to create useful clinical data repositories.

5.3. Journal and congress contributions

5.3.1. *Journal contributions and book chapters*

- **D. Boscá**, J. A. Maldonado, D. Moner, and M. Robles, “Automatic generation of computable implementation guides from clinical information models,” *J Biomed Inform*, vol. 55, pp. 143–152, Jun. 2015.
- S. Kobayashi, **D. Boscá**, N. Kume, and H. Yoshihara, “Reforming MML (Medical Markup Language) Standard with Archetype Technology,” *Indian Association for Medical Informatics*, vol. 91, p. 57, 2014.
- M. Marcos, J. A. Maldonado, B. Martínez-Salvador, **D. Boscá**, and M. Robles, “Interoperability of clinical decision-support systems and electronic health records using archetypes: a case study in clinical trial eligibility,” *Journal of biomedical informatics*, vol. 46, no. 4, pp. 676–689, 2013.
- J. A. Maldonado, **D. Boscá**, D. Moner, and M. Robles, “LinkEHR: A Platform for the Normalization of Legacy Clinical Data Based on Archetypes,” *Interoperability in Healthcare Information Systems: Standards, Management, and Technology: Standards, Management, and Technology*, p. 45, 2013.
- J. A. Maldonado, C. M. Costa, D. Moner, M. Menárguez-Tortosa, **D. Boscá**, J. A. M. Giménez, J. T. Fernández-Breis, and M. Robles, “Using the ResearchEHR platform to facilitate the practical application of the EHR standards,” *Journal of biomedical informatics*, vol. 45, no. 4, pp. 746–762, 2012.
- M. Marcos, J. A. Maldonado, B. Martínez-Salvador, D. Moner, **D. Boscá**, and M. Robles, “An archetype-based solution for the interoperability of computerised guidelines and electronic health records,” in *Artificial Intelligence in Medicine*, Springer Berlin Heidelberg, 2011, pp. 276–285.
- J. A. Maldonado, D. Moner, **D. Boscá**, J. T. Fernández-Breis, C. Angulo, and M. Robles, “LinkEHR-Ed: A multi-reference model archetype editor based on formal semantics,” *International journal of medical informatics*, vol. 78, no. 8, pp. 559–570, 2009.

5.3.2. *Congress contributions*

- **D. Boscá**, D. Moner, J. A. Maldonado, and M. Robles, “Combining Archetypes with Fast Health Interoperability Resources in Future-proof Health Information Systems.,” *Studies in health technology and informatics*, vol. 210, pp. 180–184, 2014.
- **D. Boscá**, L. Marco, D. Moner, J. A. Maldonado, L. Insa, and M. Robles, “Detailed Clinical Models Governance System in a Regional EHR Project,” in *XIII Mediterranean*

- Conference on Medical and Biological Engineering and Computing 2013*, Springer, 2014, pp. 1266–1269.
- D. Moner, J. A. Maldonado, **D. Boscá**, A. Mañas, and M. Robles, “Development of a Visual Editor for the Definition of HL7 CDA Archetypes,” in *XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013*, Springer, 2014, pp. 1258–1261.
 - **D. Boscá**, L. Marco, V. Burriel, T. Jaijo, J. M. Millán, A. M. Levin, O. Pastor, M. Robles, and J. A. Maldonado, “Genetic testing information standardization in HL7 CDA and ISO13606,” in *MedInfo*, 2013, pp. 338–342.
 - C. Martínez-Costa, **D. Bosca**, M. C. Legaz-García, C. Tao, B. J. Fernández, S. Schulz, and C. G. Chute, “Iseosemantic rendering of clinical information using formal ontologies and RDF.,” *Studies in health technology and informatics*, vol. 192, pp. 1085–1085, 2012.
 - **D. Boscá**, J. A. Maldonado, D. Moner, and M. Robles, “Detailed clinical models to facilitate interstandard interoperability of data types,” *23rd International Conference of the European Federation for Medical Informatics. Oslo, Norway: European Federation for Medical Informatics*, 2011
 - J. Maldonado, D. Moner, **D. Boscá**, C. Angulo, L. Marco, E. Reig, M. Robles, and others, “Concept-based exchange of healthcare information: The LinkEHR approach,” in *Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 First IEEE International Conference on*, 2011, pp. 150–157.
 - D. Moner, J. A. Maldonado, **D. Boscá**, C. Angulo, M. Robles, D. Pérez, and P. Serrano, “CEN EN13606 normalisation framework implementation experiences,” *Seamless Care, Safe Care: The Challenges of Interoperability and Patient Safety in Health Care: Proceedings of the EFMI Special Topic Conference, June 2-4, 2010, Reykjavik, Iceland*, vol. 155, p. 136, 2010.
 - M. Robles, J. T. Fernández-Breis, J. A. Maldonado, D. Moner, C. Martínez-Costa, **D. Boscá**, and M. Menárguez-Tortosa, “ResearchEHR: Use of semantic web technologies and archetypes for the description of EHRs,” *Studies in health technology and informatics*, vol. 155, p. 129, 2010

Bibliography

1. Menachemi N, Collum TH. Benefits and drawbacks of electronic health record systems. *Risk Manag Healthc Policy*. 2011 May 11;4:47–55.
2. Stroetman V, Kalra D, Lewalle P, Rector A, Rodrigues J, Stroetman K, et al. Semantic Interoperability for Better health and Safer Healthcare [Internet]. The European Commission; 2009 [cited 2015 Sep 7] p. 1–34. Available from: <http://discovery.ucl.ac.uk/66190/>
3. Beale T. Archetypes Constraint-based Domain Models for Futureproof Information Systems. 2000.
4. Fagin R, Kolaitis PG, Miller RJ, Popa L. Data exchange: semantics and query answering. *Theor Comput Sci*. 2005 May 25;336(1):89–124.
5. Lenzerini M. Data Integration: A Theoretical Perspective. In: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems [Internet]. New York, NY, USA: ACM; 2002 [cited 2015 Sep 4]. p. 233–46. Available from: <http://doi.acm.org/10.1145/543613.543644>
6. Ten Cate B, Kolaitis PG. Structural characterizations of schema-mapping languages. *Commun ACM*. 2010 Jan 1;53(1):101.
7. ISO/TR 20514:2015 Health informatics - Electronic health record - Definition, scope and context. 2005.
8. Jensen PB, Jensen LJ, Brunak S. Mining electronic health records: towards better research applications and clinical care. *Nat Rev Genet*. 2012 Jun;13(6):395–405.
9. Kohane IS. Using electronic health records to drive discovery in disease genomics. *Nat Rev Genet*. 2011 Jun;12(6):417–28.
10. Luciano JS, Andersson B, Batchelor C, Bodenreider O, Clark T, Denney CK, et al. The Translational Medicine Ontology and Knowledge Base: driving personalized medicine by bridging the gap between bench and bedside. *J Biomed Semant*. 2011;2 Suppl 2:S1.
11. Prokosch HU, Ganslandt T. Perspectives for medical informatics. Reusing the electronic medical record for clinical research. *Methods Inf Med*. 2009;48(1):38–44.
12. ISO 13606 Health informatics - Electronic health record communication - Part 1: Reference model and Part 2: Archetype interchange specification. 2008.
13. HL7 Standards Product Brief - CDA® Release 2 [Internet]. 2005 [cited 2015 Aug 7]. Available from: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7
14. The openEHR Foundation [Internet]. 2000 [cited 2015 Nov 2]. Available from: <http://www.openehr.org>
15. openEHR foundation. Clinical Knowledge Manager - CKM [Internet]. 2007 [cited 2015 Aug 17]. Available from: <http://www.openehr.org/ckm/>

16. Parker CG, Rocha RA, Campbell JR, Tu SW, Huff SM. Detailed clinical models for sharable, executable guidelines. *Stud Health Technol Inform*. 2004;107:145–8.
17. Tao C, Jiang G, Oniki TA, Freimuth RR, Zhu Q, Sharma D, et al. A semantic-web oriented representation of the clinical element model for secondary use of electronic health records data. *J Am Med Inform Assoc JAMIA*. 2013 May 1;20(3):554–62.
18. Clinical Information Modeling Initiative (CIMI) [Internet]. 2011 [cited 2015 Sep 8]. Available from: <http://www.opencimi.org/>
19. ihtsdo.org [Internet]. SNOMED CT. 2007 [cited 2015 Jul 30]. Available from: <http://www.ihtsdo.org/snomed-ct>
20. World health Organization. The International Classification of Diseases 11th Revision [Internet]. WHO. 2015 [cited 2015 Sep 16]. Available from: <http://www.who.int/classifications/icd/revision/en/>
21. Rector AL. The interface between information, terminology, and inference models. *Stud Health Technol Inform*. 2001;84:246–50.
22. Martínez-Costa C, de Andrade A, Karlsson D, Kalra D, Schulz S. Towards the harmonization of Clinical Information and Terminologies by Formal Representation. *EJBI*. 2012;8(3):3–10.
23. Health Level Seven International [Internet]. 1987 [cited 2015 Sep 16]. Available from: <http://www.hl7.org>
24. ISO 12967 Health informatics - Service architecture. 2009.
25. CDISC [Internet]. 1997 [cited 2015 Sep 16]. Available from: <http://cdisc.org/>
26. Maldonado JA, Costa CM, Moner D, Menárguez-Tortosa M, Boscá D, Miñarro Giménez JA, et al. Using the ResearchEHR platform to facilitate the practical application of the EHR standards. *J Biomed Inform*. 2012 Aug;45(4):746–62.
27. openEHR Architecture Overview [Internet]. 2003 [cited 2015 Sep 3]. Available from: http://openehr.org/releases/BASE/latest/docs/architecture_overview/architecture_overview.html
28. Archetype Definition Language 1.4 (ADL) [Internet]. 2003 [cited 2015 Nov 2]. Available from: <http://www.openehr.org/releases/1.0.2/architecture/am/adl.pdf>
29. Object Constraint Language (OCL) Specification v2.4 [Internet]. 2014 [cited 2014 Oct 23]. Available from: <http://www.omg.org/spec/OCL/2.4/>
30. ISO 13606 Health informatics - Electronic health record communication - Part 2: Archetype interchange specification. 2008.
31. ISO 13606 Health informatics - Electronic health record communication - Part 3: Reference archetypes and term lists. 2009.
32. ISO 13606 Health informatics - Electronic health record communication - Part 4: Security. 2009.

33. ISO 13606 Health informatics - Electronic health record communication - Part 5: Exchange models. 2010.
34. Recursos de Modelado Clínico (arquetipos) [Internet]. 2013 [cited 2015 Jan 16]. Available from: https://www.msssi.gob.es/profesionales/hcdsns/areaRecursosSem/Rec_mod_clinico_arquetipos.htm
35. Helen Broberg. CEN/ISO EN13606 in Sweden [Internet]. EN13606 assembly; 2010 Jun 24 [cited 2015 Nov 2]; Madrid. Available from: <http://www.en13606.org/images/docs/workshop2010/08-EN13606Sweden-Broberg-Rosenalv-CEHIS.pdf>
36. Abreu Maia T, Fernandes De Muylder C, Mendonça Queiroga R. Archetype Development Process of Electronic Health Record of Minas Gerais. *Stud Health Technol Inform.* 2015;216:938.
37. Secretaria de Saúde - B-RES - Arquétipos Publicados [Internet]. 2010 [cited 2015 Sep 14]. Available from: <http://sres.saude.mg.gov.br/arquetipo/listar>
38. Archetype Query Language (AQL) Description [Internet]. 2008 [cited 2015 Nov 2]. Available from: <http://www.openehr.org/wiki/display/spec/Archetype+Query+Language+Description>
39. openEHR Guideline Definition Language (GDL) [Internet]. 2013 [cited 2015 Sep 1]. Available from: <http://www.openehr.org/releases/CDS/latest/docs/GDL/GDL.html>
40. Nasjonal IKT Clinical Knowledge Manager [Internet]. 2014 [cited 2015 Oct 27]. Available from: <http://arketyper.no/ckm/>
41. CENTERMS Clinical Knowledge Manager [Internet]. 2014 [cited 2015 Oct 27]. Available from: <http://www.centerms.org.br:8009/ckm/>
42. UK Clinical Knowledge Manager [Internet]. 2007 [cited 2015 Oct 27]. Available from: <http://www.clinicalmodels.org.uk/ckm/>
43. NEHTA Clinical Knowledge Manager [Internet]. 2005 [cited 2015 Oct 27]. Available from: <http://dcm.nehta.org.au/ckm/>
44. Upravlavec kliničnega znanja [Internet]. 2007 [cited 2015 Oct 27]. Available from: http://ukz.ezdrav.si/ckm/OKM_sl.html
45. Moreno-Conde A, Moner D, da Cruz WD, Santos MR, Maldonado JA, Robles M, et al. Clinical information modeling processes for semantic interoperability of electronic health records: systematic review and inductive analysis. *J Am Med Inform Assoc.* 2015;ocv008.
46. Braun M, Brandt AU, Schulz S, Boeker M. Validating archetypes for the Multiple Sclerosis Functional Composite. *BMC Med Inform Decis Mak.* 2014;14(1).
47. Kalra D, Tapuria A, Austin T, De Moor G. Quality requirements for EHR archetypes. *Stud Health Technol Inform.* 2012;180:48–52.

48. Ahn S, Huff SM, Kim Y, Kalra D. Quality metrics for detailed clinical models. *Int J Med Inf.* 2013 May;82(5):408–17.
49. ISO 18864 Health informatics - Quality metrics for detailed clinical models.
50. Berges I, Bermudez J, Illarramendi A. Binding SNOMED CT terms to archetype elements. Establishing a baseline of results. *Methods Inf Med.* 2015;54(1):45–9.
51. Meizoso M, Allones JL, Taboada M, Martinez D, Tellado S. Automated Mapping of Observation Archetypes to SNOMED CT Concepts. In: Ferrández JM, Sánchez JRÁ, Paz F de la, Toledo FJ, editors. *Foundations on Natural and Artificial Computation* [Internet]. Springer Berlin Heidelberg; 2011 [cited 2015 Oct 27]. p. 550–61. Available from: http://link.springer.com/chapter/10.1007/978-3-642-21344-1_57
52. Sundvall E, Qamar R, Nyström M, Forss M, Petersson H, Karlsson D, et al. Integration of tools for binding archetypes to SNOMED CT. *BMC Med Inform Decis Mak.* 2008 Oct 27;8(Suppl 1):S7.
53. Yu S, Berry D, Bisbal J. An Investigation of Semantic Links to Archetypes in an External Clinical Terminology through the Construction of Terminological“ Shadows.” 2010 [cited 2015 Nov 13]; Available from: http://arrow.dit.ie/teapotcon/9/?utm_source=arrow.dit.ie%2Fteapotcon%2F9&utm_medium=PDF&utm_campaign=PDFCoverPages
54. Menárguez-Tortosa M, Fernández-Breis JT. OWL-based reasoning methods for validating archetypes. *J Biomed Inform.* 2013 Abril;46(2):304–17.
55. Legaz-García MDC, Martínez-Costa C, Fernández-breis JT. Exploitation of ontologies for the management of clinical archetypes in ArchMS. 2012 [cited 2015 Oct 27]; Available from: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.416.6594>
56. Allones JL, Taboada M, Martinez D, Lozano R, Sobrido MJ. SNOMED CT module-driven clinical archetype management. *J Biomed Inform.* 2013 Jun;46(3):388–400.
57. Martínez-Costa C, Menárguez-Tortosa M, Fernández-Breis JT, Maldonado JA. A model-driven approach for representing clinical archetypes for Semantic Web environments. *J Biomed Inform.* 2009 Feb;42(1):150–64.
58. Porn AM, Peres LM, Didonet Del Fabro M. A Process for the Representation of openEHR ADL Archetypes in OWL Ontologies. *Stud Health Technol Inform.* 2015;216:827–31.
59. Martínez Costa C, Menárguez-Tortosa M, Fernández-Breis JT. Clinical data interoperability based on archetype transformation. *J Biomed Inform.* 2011;44(5):869–80.
60. Dentler K, ten Teije A, Cornet R, de Keizer N. Semantic Integration of Patient Data and Quality Indicators Based on openEHR Archetypes. In: Lenz R, Miksch S, Peleg M, Reichert M, Riaño D, ten Teije A, editors. *Process Support and Knowledge Representation in Health Care* [Internet]. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013 [cited 2015 Aug 7]. p. 85–97. Available from: http://link.springer.com/10.1007/978-3-642-36438-9_6

61. Lezcano L, Sicilia M-A, Rodríguez-Solano C. Integrating reasoning and clinical archetypes using OWL ontologies and SWRL rules. *J Biomed Inform.* 2011 Abril;44(2):343–53.
62. Fernández-Breis JT, Maldonado JA, Marcos M, Legaz-García M del C, Moner D, Torres-Sospedra J, et al. Leveraging electronic healthcare record standards and semantic web technologies for the identification of patient cohorts. *J Am Med Inform Assoc JAMIA.* 2013 Dec;20(e2):e288–96.
63. Marco-Ruiz L, Moner D, Maldonado JA, Kolstrup N, Bellika JG. Archetype-based data warehouse environment to enable the reuse of electronic health record data. *Int J Med Inf.* 2015;84(9):702–14.
64. HL7 Standards - HL7 Version 2 [Internet]. 2011 [cited 2015 Sep 14]. Available from: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=185
65. Boone KW. The CDA TM book [Internet]. Springer-Verlag London; 2011 [cited 2015 Nov 2]. Available from: <http://www.springer.com/public+health/book/978-0-85729-335-0>
66. Introduction to Meaningful Use - CDC [Internet]. 2012 [cited 2015 Sep 8]. Available from: <http://www.cdc.gov/ehrmmeaningfuluse/introduction.html>
67. European Patients Smart Open Services (epSOS) [Internet]. 2008 [cited 2015 Aug 17]. Available from: <http://www.epsos.eu/>
68. Fast Healthcare Interoperability Resources (FHIR) Specification [Internet]. 2014 [cited 2015 Nov 2]. Available from: <http://hl7.org/implement/standards/fhir/>
69. Publicly Available FHIR Servers for testing - HL7Wiki [Internet]. 2014 [cited 2015 Sep 14]. Available from: http://wiki.hl7.org/index.php?title=Publicly_Available_FHIR_Servers_for_testing
70. Kobayashi S, Kume N, Yoshihara H. Restructuring an EHR system and the Medical Markup Language (MML) standard to improve interoperability by archetype technology. *Stud Health Technol Inform.* 2015;216:881.
71. Guo J, Takada A, Niu T, He M, Tanaka K, Sato J, et al. Enhancement of MML Medical Data Exchange Standard for a Localized Chinese Version. *J Med Syst.* 2005 Oct;29(5):555–67.
72. IHTSDO. SNOMED CT Compositional Grammar Specification and Guide - July 2015 [Internet]. [cited 2015 Sep 11]. Available from: <http://snomed.org/compprogram.pdf.html>
73. WHO | International Classification of Diseases (ICD) [Internet]. WHO. 1990 [cited 2015 Oct 28]. Available from: <http://www.who.int/classifications/icd/en/>
74. Logical Observation Identifiers Names and Codes (LOINC) [Internet]. 1994 [cited 2015 Sep 7]. Available from: <https://loinc.org/>
75. Ling Liu. Encyclopedia of Database Systems [Internet]. Springer; 2009 [cited 2015 Oct 13]. Available from: <http://www.springer.com/us/book/9780387355443>

76. Gruninger, M, Lee, J. *Ontology Applications and Design*. Commun ACM. 2002;45(2):39–41.
77. Friedman M, Levy AY, Millstein TD, others. *Navigational plans for data integration*. AAAI/IAAI. 1999;1999:67–73.
78. Raffio A, Braga D, Ceri S, Papotti P, Hernández M, others. *Clip: a visual language for explicit schema mappings*. In: *Data Engineering, 2008 ICDE 2008 IEEE 24th International Conference on* [Internet]. IEEE; 2008 [cited 2015 Oct 27]. p. 30–9. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4497411
79. Bojańczyk M, Kołodziejczyk LA, Murlak F. *Solutions in XML data exchange*. J Comput Syst Sci. 2013 Sep;79(6):785–815.
80. *Foundations of Data Exchange*. 1 edition. Cambridge, UK; New York: Cambridge University Press; 2014. 341 p.
81. *Clio: Schema Mapping Creation and Data Exchange - Springer* [Internet]. 2009 [cited 2015 Oct 13]. Available from: http://link.springer.com/chapter/10.1007%2F978-3-642-02463-4_12
82. Bonifati A, Chang EQ, Lakshmanan AVS, Ho T, Pottinger R. *HePToX: Marrying XML and Heterogeneity in Your P2P Databases*. In: *Proceedings of the 31st International Conference on Very Large Data Bases* [Internet]. Trondheim, Norway: VLDB Endowment; 2005 [cited 2015 Oct 13]. p. 1267–70. Available from: <http://dl.acm.org/citation.cfm?id=1083592.1083745>
83. Alexe B, Tan W-C. *A New Framework for Designing Schema Mappings*. In: Tannen V, Wong L, Libkin L, Fan W, Tan W-C, Fourman M, editors. *In Search of Elegance in the Theory and Practice of Computation* [Internet]. Springer Berlin Heidelberg; 2013 [cited 2015 Oct 29]. p. 56–88. Available from: http://link.springer.com/chapter/10.1007/978-3-642-41660-6_4
84. Alexe B, Hernández M, Popa L, Tan W-C. *MapMerge: Correlating Independent Schema Mappings*. VLDB J. 2012 Abril;21(2):191–211.
85. Duftschmid G, Wrba T, Rinner C. *Extraction of standardized archetyped data from Electronic Health Record systems based on the Entity-Attribute-Value Model*. Int J Med Inf. 2010 Aug;79(8):585–97.
86. Duftschmid G, Chaloupka J, Rinner C. *Towards plug-and-play integration of archetypes into legacy electronic health record systems: the ArchiMed experience*. BMC Med Inform Decis Mak. 2013;13:11.
87. *Model Driven Architecture (MDA) Specifications* [Internet]. 2001 [cited 2015 Jan 12]. Available from: <http://www.omg.org/mda/specs.htm>
88. Truyen F. *The Fast Guide to Model Driven Architecture - The basics of Model Driven Architecture*. 2006 Enero [cited 2015 Jan 12]; Available from: http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf
89. *Object Management Group (OMG)* [Internet]. 1989 [cited 2015 Nov 2]. Available from: <http://www.omg.org/>

90. Menárguez Tortosa, Marcos. Modelos de representación de arquetipos en sistemas de información sanitarios [Internet] [Thesis]. Universidad de Murcia; 2013. Available from: <http://www.tdx.cat/bitstream/handle/10803/117386/TMMT.pdf?sequence=1>
91. Atalag K, Yang HY, Tempero E, Warren J. Model driven development of clinical information systems using openEHR. *Stud Health Technol Inform*. 2011;169:849–53.
92. Kobayashi S, Kimura E, Ishihara K. Archetype model-driven development framework for EHR web system. *Healthc Inform Res*. 2013;19(4):271–7.
93. Martínez-Costa C, Menárguez-Tortosa M, Fernández-Breis JT. An approach for the semantic interoperability of ISO EN 13606 and OpenEHR archetypes. *J Biomed Inform*. 2010 Oct;43(5):736–46.
94. Model-Driven Health Tools (MDHT) [Internet]. 2009 [cited 2015 Nov 2]. Available from: <https://www.projects.openhealthtools.org/sf/projects/mdht/>
95. Unified Modeling Language (UML) v2.5 [Internet]. 2015 [cited 2015 Nov 2]. Available from: <http://www.omg.org/spec/UML/>
96. Web Ontology Language (OWL) [Internet]. 2004 [cited 2015 Nov 2]. Available from: <http://www.w3.org/TR/owl-ref/>
97. SWRL: A Semantic Web Rule Language Combining OWL and RuleML [Internet]. 2004 [cited 2015 Nov 2]. Available from: <http://www.w3.org/Submission/SWRL/>
98. Rule Markup Language (RuleML) [Internet]. 2001 [cited 2015 Nov 2]. Available from: <http://www.ruleml.org/>
99. World Wide Web (W3C) Rule Interchange Format [Internet]. 2013 [cited 2014 Oct 23]. Available from: <http://www.w3.org/TR/rif-overview/>
100. Natural Rule Language (NRL) Specification 1.4.0 [Internet]. 2010 [cited 2014 Oct 23]. Available from: <http://nrl.sourceforge.net/spec/>
101. Project Attempto [Internet]. 1995 [cited 2015 Nov 2]. Available from: <http://attempto.ifi.uzh.ch>
102. Farkash A, Timm JTE, Waks Z. A model-driven approach to clinical practice guidelines representation and evaluation using standards. *Stud Health Technol Inform*. 2013;192:200–4.
103. Fuchs NE, Schwertel U, Schwitter R. Attempto Controlled English — Not Just Another Logic Specification Language. In: Flener P, editor. *Logic-Based Program Synthesis and Transformation* [Internet]. Springer Berlin Heidelberg; 1999 [cited 2015 Nov 2]. p. 1–20. Available from: http://link.springer.com/chapter/10.1007/3-540-48958-4_1
104. ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – Schematron [Internet]. 2006 [cited 2015 Nov 2]. Available from: <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>
105. Alexandru A. NRL to Schematron generation tutorial [Internet]. 2011 [cited 2015 Nov 2]. Available from: <http://nrl.sourceforge.net/tutorials/schematron/tutorial.html>

106. Pfeiffer K, Duftschmid G, Rinner C. Validating EHR documents: automatic schematron generation using archetypes. *Stud Health Technol Inform.* 2014;198:101–7.
107. Drools - Business Rules Management System (Java™, Open Source) [Internet]. 2005 [cited 2015 Nov 9]. Available from: <http://www.drools.org/>
108. openCDS [Internet]. 2011 [cited 2015 Sep 1]. Available from: <http://www.opencds.org>
109. SemanticHealthNet [Internet]. 2011 [cited 2015 Sep 8]. Available from: <http://www.semantichealthnet.eu/>
110. Blumenthal D, Tavenner M. The “Meaningful Use” Regulation for Electronic Health Records. *N Engl J Med.* 2010 Agosto;363(6):501–4.
111. Medicare.gov [Internet]. 1965 [cited 2015 Sep 8]. Available from: <https://www.medicare.gov/>
112. Medicaid [Internet]. 1965 [cited 2015 Sep 8]. Available from: <http://www.medicaid.gov/>
113. Trillium Bridge [Internet]. 2013 [cited 2015 Sep 8]. Available from: <http://www.trilliumbridge.eu/>
114. Expand Project [Internet]. 2014 [cited 2015 Sep 22]. Available from: <http://www.expandproject.eu/>
115. Angulo C, Crespo P, Maldonado JA, Moner D, Pérez D, Abad I, et al. Non-invasive lightweight integration engine for building EHR from autonomous distributed systems. *Int J Med Inf.* 2007 Dec;76:S417–24.
116. Kalra D. EHR archetypes in practice: getting feedback from clinicians and the role of EuroRec. [Internet]. 2007 [cited 2015 Aug 13]. Available from: http://www.eurorec.org/news_events/eurorec2007.cfm
117. Kuper GM, Siméon J. Subsumption for XML Types. In: Bussche JV den, Vianu V, editors. *Database Theory — ICDT 2001* [Internet]. Springer Berlin Heidelberg; 2001 [cited 2015 Jan 15]. p. 331–45. Available from: http://link.springer.com/chapter/10.1007/3-540-44503-X_21
118. Maldonado JA, Moner D, Bosca D, Fernández-Breis JT, Angulo C, Robles M. LinkEHR-Ed: A multi-reference model archetype editor based on formal semantics. *Int J Med Inf.* 2009 Aug;78(8):559–70.
119. Bosca D, Marco L, Moner D, Maldonado JA, Insa L, Robles M. Detailed Clinical Models Governance System in a Regional EHR Project. In: Roa Romero LM, editor. *XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013* [Internet]. Cham: Springer International Publishing; 2014 [cited 2015 Aug 31]. p. 1266–9. Available from: http://link.springer.com/10.1007/978-3-319-00846-2_313
120. Lee KP, Hu J. XML Schema Representation of DICOM Structured Reporting. *J Am Med Inform Assoc JAMIA.* 2003 Apr;10(2):213–23.
121. openEHR - Basic Meta-Model (BMM) files [Internet]. 2008 [cited 2015 Aug 5]. Available from: <http://www.openehr.org/releases/1.0.2/reference-models/openEHR/BMM/>

122. XMI v2.5.1 [Internet]. 2015 [cited 2015 Sep 21]. Available from: <http://www.omg.org/spec/XMI/>
123. openEHR/bmm [Internet]. 2008 [cited 2015 Sep 21]. Available from: <https://github.com/openEHR/bmm>
124. Eclipse Modeling Framework [Internet]. 2013 [cited 2015 Sep 21]. Available from: <http://www.eclipse.org/modeling/emf/>
125. FHIR resources ecore definitions [Internet]. 2014 [cited 2014 Oct 15]. Available from: <http://www.hl7.org/implement/standards/fhir/ecoredefinitions.xml>
126. openEHR/odin [Internet]. 2003 [cited 2015 Sep 21]. Available from: <https://github.com/openEHR/odin>
127. MML Version 3.0 [Internet]. 2003 [cited 2015 Aug 5]. Available from: http://www.medxml.net/E_mml30/
128. Cluet S, Siméon J. Data Integration Based on Data Conversion and Restructuring. In: Proceedings of WebDB'98. 1997.
129. Guo J, Takada A, Tanaka K, Sato J, Suzuki M, Suzuki T, et al. The Development of MML (Medical Markup Language) Version 3.0 as a Medical Document Exchange Format for HL7 Messages. *J Med Syst*. 2004 Dec;28(6):523–33.
130. Moner D, Maldonado JA, Boscá D, Mañas A, Robles M. Development of a Visual Editor for the Definition of HL7 CDA Archetypes. In: Roa Romero LM, editor. XIII Mediterranean Conference on Medical and Biological Engineering and Computing 2013 [Internet]. Cham: Springer International Publishing; 2014 [cited 2015 Aug 19]. p. 1258–61. Available from: http://link.springer.com/10.1007/978-3-319-00846-2_311
131. Maldonado JA, Moner D, Tomás D, Angulo C, Robles M, Fernández JT. Framework for clinical data standardization based on archetypes. *Stud Health Technol Inform*. 2007;129(Pt 1):454–8.
132. SemanticHealthNet. Semantic Interoperability for Health Network Deliverable 4.5: Design Guides and Roadmap [Internet]. 2015. Available from: http://www.semantichealthnet.eu/SemanticHealthNet/assets/File/SHN%20288408%20D4_5%20Design%20Guides%20and%20Roadmap.pdf
133. Linda Bird. Modelling Patterns Review & Suggestions [Internet]. 2012 Aug 16; CIMI. Available from: https://drive.google.com/file/d/0B_i5YqHx_cMoMXRac2x3WINLeUE/view
134. Maldonado JA, Moner D, Boscá D, Fernández-Breis JT, Angulo C, Robles M. LinkEHR-Ed: a multi-reference model archetype editor based on formal semantics. *Int J Med Inf*. 2009 Aug;78(8):559–70.
135. IHTSDO. SNOMED CT Expression Constraint Language Specification and Guide [Internet]. 2015 [cited 2015 Nov 10]. Available from: <http://snomed.org/expressionconstraint>

136. V.M. Giménez, J.A. Maldonado, M. Robles. Definición de subconjuntos en SNOMED CT. In Madrid; 2015 [cited 2015 Nov 6]. p. 224–7. Available from: <http://snquery.veratech.es/>
137. JSON [Internet]. 2014 [cited 2015 Sep 9]. Available from: <http://json.org/>
138. JSON Schema and Hyper-Schema [Internet]. 2014 [cited 2015 Sep 9]. Available from: <http://json-schema.org/>
139. Grahame Grieve. Language Localization in #FHIR [Internet]. Health Intersections Pty Ltd. 2015 [cited 2015 Nov 4]. Available from: <http://www.healthintersections.com.au/?p=2382>
140. Martínez Costa C. Modelos de representación y transformación para la interoperabilidad semántica entre estándares de Historia Clínica Electrónica basados en arquitectura de modelo dual / Catalina Martínez Costa; director, Jesualdo Tomás Fernández Breis. [Internet]. 2011 [cited 2015 Aug 7]. Available from: <https://digitum.um.es/xmlui/handle/10201/31557>
141. Moner D, Moreno A, Maldonado JA, Robles M, Parra C. Using archetypes for defining CDA templates. *Stud Health Technol Inform.* 2012;180:53–7.
142. Moner D, Bru, Juan, Maldonado, José A., Robles, Montserrat. Clinical Trials powered by Electronic Health Records. In: CDISC Interchange Europe. Stockholm, Sweden; 2012.
143. Smits M, Kramer E, Harthoorn M, Cornet R. A comparison of two Detailed Clinical Model representations: FHIR and CDA. *EJBI* [Internet]. 2015 [cited 2015 Aug 19];11(2). Available from: http://www.ejbi.org/img/ejbi/2015/2/Smits_en.pdf
144. Fagin R, Kolaitis PG, Popa L. *Data Exchange: Getting to the Core.* 2003.
145. Bonifati A, Mecca G, Papotti P, Velegrakis Y. Discovery and Correctness of Schema Mapping Transformations. In: Bellahsene Z, Bonifati A, Rahm E, editors. *Schema Matching and Mapping* [Internet]. Springer Berlin Heidelberg; 2011 [cited 2015 Aug 14]. p. 111–47. Available from: http://link.springer.com/chapter/10.1007/978-3-642-16518-4_5
146. Popa L, Velegrakis Y, Hernández MA, Miller RJ, Fagin R. Translating Web Data. In: *Proceedings of the 28th International Conference on Very Large Data Bases* [Internet]. Hong Kong, China: VLDB Endowment; 2002 [cited 2015 Aug 14]. p. 598–609. Available from: <http://dl.acm.org/citation.cfm?id=1287369.1287421>
147. Fuxman A, Hernandez MA, Ho H, Miller RJ, Papotti P, Popa L. Nested Mappings: Schema Mapping Reloaded. In: *Proceedings of the 32Nd International Conference on Very Large Data Bases* [Internet]. Seoul, Korea: VLDB Endowment; 2006 [cited 2015 Nov 10]. p. 67–78. Available from: <http://dl.acm.org/citation.cfm?id=1182635.1164135>
148. Abiteboul S, Bidoit N. Non First Normal Form relations: An algebra allowing data restructuring. *J Comput Syst Sci.* 1986 Diciembre;33(3):361–93.
149. Michael Kay. *Blooming FLWOR - An Introduction to the XQuery FLWOR Expression* [Internet]. 2006 [cited 2015 Oct 26]. Available from: http://www.stylusstudio.com/whitepapers/blooming_flwor.pdf

150. Bret Victor. Inventing on Principle [Internet]. CUSEC 2012; 2012 Jan 21 [cited 2015 Nov 2]; Montreal, Quebec. Available from: <https://vimeo.com/36579366>
151. Alexe B, Tan W-C, Velegrakis Y. Comparing and Evaluating Mapping Systems with STBenchmark. In 2008. p. 1468–71. Available from: http://portal.acm.org/ft_gateway.cfm?id=1454203
152. Reddy M, Pratt W, Dourish P, Shabot MM. Sociotechnical requirements analysis for clinical systems. *Methods Inf Med*. 2003;42(4):437–44.
153. Dolin RH, Alschuler L, Boyer S, Beebe C, Behlen FM, Biron PV, et al. HL7 Clinical Document Architecture, Release 2. *J Am Med Inform Assoc JAMIA*. 2006;13(1):30–9.
154. HL7 Implementation Guide for CDA[®] Release 2: IHE Health Story Consolidation, Release 1.1 - US Realm. 2012.
155. Medicare and Medicaid Programs; Electronic Health Record Incentive Program – Stage 2. 171 Sep 4, 2012 p. 53967–4162.
156. D’Amore JD, Mandel JC, Kreda DA, Swain A, Koromia GA, Sundareswaran S, et al. Are Meaningful Use Stage 2 certified EHRs ready for interoperability? Findings from the SMART C-CDA Collaborative. *J Am Med Inform Assoc*. 2014;
157. Garde S, Chen R, Leslie H, Beale T, McNicoll I, Heard S. Archetype-based knowledge management for semantic interoperability of electronic health records. *Stud Health Technol Inform*. 2009;150:1007–11.
158. LinkEHR Platform [Internet]. 2005 [cited 2015 Nov 2]. Available from: <http://www.linkehr.com>
159. Indizen ITServer [Internet]. 2015 [cited 2015 Nov 2]. Available from: <http://www.itserver.es>
160. Bosca D, Marco L, Burriel V, Jaijo T, Millán JM, Levin A, et al. Genetic testing information standardization in HL7 CDA and ISO13606. *Stud Health Technol Inform*. 2013;192:338–42.
161. CDA Implementation Guide for Genetic Testing Report (GTR) (September 2011 Draft). 2011.
162. Rene Spronk, Grahame Grieve. Common issues found in implementations of the HL7 Clinical Document Architecture (CDA) [Internet]. 2008 [cited 2015 Nov 2]. Available from: http://www.ringholm.com/docs/03020_en_HL7_CDA_common_issues_error.htm
163. Chen R, Georgii-Hemming P, Ahlfeldt H. Representing a chemotherapy guideline using openEHR and rules. *Stud Health Technol Inform*. 2009;150:653–7.
164. Anani N, Chen R, Moreira TP, Koch S. Retrospective checking of compliance with practice guidelines for acute stroke care: a novel experiment using openEHR’s Guideline Definition Language. *BMC Med Inform Decis Mak*. 2014 May 10;14(1):39.

165. M. Bacelar-Silva G, Chen R, J. Cruz-Correia R. From Clinical Guideline to openEHR: Converting JNC7 Into Archetypes and Template. In: Anais do XIII Congresso Brasileiro de Informática em Saúde, ISSN: 2178-2857. Curitiba, Brazil; 2012.
166. Barretto SA, Warren J, Goodchild A, Bird L, Heard S, Stumptner M. Linking Guidelines to Electronic Health Record Design for Improved Chronic Disease Management. *AMIA Annu Symp Proc.* 2003;2003:66–70.
167. Marcos M, Maldonado JA, Martínez-Salvador B, Boscá D, Robles M. Interoperability of clinical decision-support systems and electronic health records using archetypes: A case study in clinical trial eligibility. *J Biomed Inform.* 2013 Agosto;46(4):676–89.
168. González-Ferrer A, Peleg M, Verhees B, Verlinden J-M, Marcos C. Data Integration for Clinical Decision Support Based on openEHR Archetypes and HL7 Virtual Medical Record. In: *Proceedings of the 2012 International Conference on Process Support and Knowledge Representation in Health Care [Internet].* Berlin, Heidelberg: Springer-Verlag; 2013 [cited 2014 Jul 15]. p. 71–84. Available from: http://dx.doi.org/10.1007/978-3-642-36438-9_5
169. Garcia D, Moro CMC, Cicogna PE, Carvalho DR. Method to integrate clinical guidelines into the electronic health record (EHR) by applying the archetypes approach. *Stud Health Technol Inform.* 2013;192:871–5.
170. Institute of Medicine (US). Standardization to Enhance Data Sharing. 2013 Mar 29 [cited 2015 Nov 11]; Available from: <http://www.ncbi.nlm.nih.gov/books/NBK137818/>
171. Pawlik M, Augsten N. RTED: A Robust Algorithm for the Tree Edit Distance. *Proc VLDB Endow.* 2011 Diciembre;5(4):334–45.
172. Pawlik M, Augsten N. A Memory-Efficient Tree Edit Distance Algorithm. In: Decker H, Lhotská L, Link S, Spies M, Wagner RR, editors. *Database and Expert Systems Applications [Internet].* Springer International Publishing; 2014 [cited 2015 Nov 10]. p. 196–210. Available from: http://link.springer.com/chapter/10.1007/978-3-319-10073-9_16
173. Cunningham J, Ainsworth J. Simulating realistic enough patient records. *Stud Health Technol Inform.* 2015;210:35–9.
174. Kate Hamilton, Lauren Wood. Schematron in the Context of the Clinical Document Architecture (CDA). In: *Proceedings of Balisage: The Markup Conference 2012 [Internet].* Montréal, Canada; 2012 [cited 2015 Nov 11]. Available from: <http://www.balisage.net/Proceedings/vol8/html/Wood01/BalisageVol8-Wood01.html>
175. Boone KW. Validating the Content of a CDA™ Document. *CDA TM Book.* 2011;
176. Sáez C, Rodrigues PP, Gama J, Robles M, García-Gómez JM. Probabilistic change detection and visualization methods for the assessment of temporal stability in biomedical data quality. *Data Min Knowl Discov.* 2014 Sep 2;29(4):950–75.
177. Sáez C, Robles M, García-Gómez JM. Stability metrics for multi-source biomedical data based on simplicial projections from probability distribution distances. *Stat Methods Med Res.* 2014 Aug 4;