Journal of Biomedical Informatics 55 (2015) 143–152

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Automatic generation of computable implementation guides from clinical information models

Diego Boscá [a,*], José Alberto Maldonado [a,b], David Moner [a], Montserrat Robles [a]

[a] Instituto Universitario de Aplicaciones de las Tecnologías de la Información y Comunicaciones Avanzadas (ITACA), Universitat Politècnica de València, Valencia, Spain
[b] VeraTech for Health SL, Valencia, Spain

ARTICLE INFO

Article history:
Received 13 November 2014
Revised 3 February 2015
Accepted 7 April 2015
Available online 21 April 2015

Keywords:
Archetype
Natural Rule Language
Implementation guide
Data validation
Clinical information model

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 oriented to human-readability, 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.

1. Introduction

Capturing requirements in the clinical domain is a difficult task [1]. 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 [2]. Typically clinicians rely on non-formal approaches (such as spreadsheet or word processor 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 [3] 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 [4], which are being used to help providers to meet the applicable Meaningful Use objectives [5]. However, the interpretation of the constraints in an implementation guide may differ from person to person [6], therefore limiting semantic interoperability.

Another type of resource for the specification of clinical information models are archetypes. Archetypes are a key part of the dual model approach on which the EN ISO 13606 norm [7] and the openEHR specification [8] are based. The dual model approach is a recent paradigm for the specification of EHR Architectures

* Corresponding author at: Instituto ITACA, Universitat Politècnica de València, Camino de vera s/n, Edificio 8G, Spain. Tel.: +34 963870000x75277.
E-mail address: diebosto@upv.es (D. Boscá).

http://dx.doi.org/10.1016/j.jbi.2015.04.002
1532-0464/

(EHRA). It distinguishes two models: the Reference Model (RM) and Archetype Model. In a broad sense, a RM 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 RM 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 RM 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 RM 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) [9] 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 a 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 Fig. 1, 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) [10]. We also include additional reference materials in the implementation guide, such as a mindmaps, value sets and bibliographic references.

Our solution will improve the current implementation guides generation process in two different ways: Firstly, our implementation guides are based on a formal natural language that allows the direct application to end EHR systems and data, and secondly, implementation guides are generated automatically from the organization's information models, in our case archetypes.

We will exemplify our approach by generating implementation guides from an EN ISO 13606 archetype, openEHR archetypes from the Clinical Knowledge Manager (CKM) [11] and HL7 CDA archetypes from the Genetic Testing Report.

We will evaluate the correctness of our methodology from three different perspectives. First, we demonstrate formally that the generated data instances are valid with respect to the underlying information model. Second, we test if generated rules can correctly validate data instances. Third, we evaluate the quality of implementation guides and their usefulness for the development of EHR systems.

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

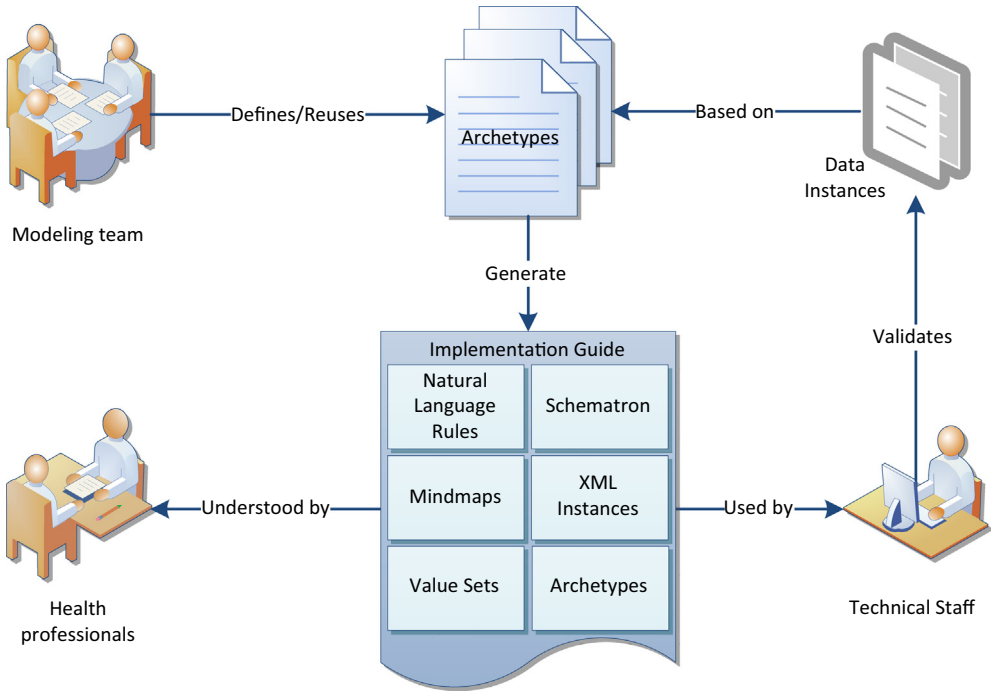


Fig. 1. Proposed architecture for the generation of implementation guides from archetypes.

Constraint Language (OCL) [12], an OMG [13] standard for the definition of rules over UML models [14]. There are also languages for defining Horn-like rules for the Ontology Web Language (OWL) [15], such as Semantic Web Rule Language (SWRL) [16] or RuleML [17]. The widespread use of rules, formal or not, has caused the creation of proposals, like the W3C Rule Interchange Format (RIF) [18], 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 [10] and Attempto Controlled English [19]. 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 [20]. 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) [19] 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 [21], 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 [22] 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 [23] as well as from archetypes [24]. Advanced features of archetype methodology, such as reuse of internal or external types can be also reproduced with Schematron.

Another successful approach for the automatic generation of systems and reference artifacts is Model-Driven Development (MDD). Model-Driven software development tries to improve correctness and productivity in software creation by producing code from abstract, human-elaborated modeling diagrams. Model-driven architecture (MDA) [25,26] is the OMG proposal to support Model-driven engineering. In MDA business processes and applications are specified using platform-independent models (PIM), which define system functionality. Standard mapping techniques transform PIMs into platform-specific models (PSM) and into final implementations. In MDA there is also a layer to bridge the gap between domain experts and information technologists called Computation Independent Model (CIM). CIM requirements should be traceable to the PIM and PSM constructs that implement them.

MDA development has been successfully applied to healthcare domain in several use cases [27–31]. This is a good solution to the problem of continuously evolving clinical knowledge. However, if not done right, the system should be continuously regenerated as the clinical knowledge evolves. Dual model approach [2] tries to solve this problem by leaving the specification of clinical knowledge out of the information model. This approach proposes a simple information model to give support to the clinical information models. Even if clinical knowledge changes, the information model does not need to change. Furthermore, in a dual model architecture clinicians are in charge of defining the requirements, contents, and structure of the clinical information models. Using dual model architecture allows the reuse of quality publicly available archetypes available in different international repositories, which guarantees the quality of source models in our methodology.

When compared with Model-Driven development, dual model approach is similar in the way that both put models as key parts of systems. In fact, archetypes can be considered as MDA CIMs [32] and have been used already as such in several projects [33,34]. Archetypes have been used in MDA projects for the transformation between information models [35], but not for the generation of derived reference materials such as instances, rules, or implementation guides.

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 [36]. 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.

The quality of implementation guides and reference materials highly depends on the quality of source information models. Several metrics and requirements have been proposed to deal with clinical information model quality [37,38]. If clinical information models follow this metrics and requirements, we can ensure the quality of end systems [39]. Clinical information models quality assessment is an important topic, as demonstrated by the development of ISO/DTS 18864 norm of quality metrics for Detailed Clinical Models [40].

3. Material and methods

3.1. LinkEHR platform

LinkEHR® [41] is a software tool for the integration and normalization of health data [42]. 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.

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 classes, section classes, 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 hided 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.

3.2.1. Generation of NRL rules

In the archetype methodology, archetype entities are created by constraining a reference model type [2], 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 Fig. 2. In Fig. 3 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 a node identifier is found in data.

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 Fig. 4), attributes existence and cardinality (shown in Fig. 5), and on data values (shown in Fig. 6). 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 Fig. 5 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.

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.

3.2.2. 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, only entities (classes and attributes) of the RM which are actually constrained need to appear in the archetype definition. It is supposed that the constraints defined in the underlying RM 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 RM 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 RM. This process is fully described in [42]. 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 Fig. 7.

3.2.3. 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

```
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"}
                }
            }
        }
    }
```

Fig. 2. 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'
```

Fig. 3. 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
```

Fig. 4. Sample rule for occurrences constraint using the readable variables described in Fig. 3.

each entity with an assertion for each one of the tests (namely tests for occurrences, cardinality, existence, and values). In Fig. 8 we show the equivalent Schematron rule to the NRL rule described in Fig. 4. As it can be observed, Schematron rules are by far less understandable than NRL rules.

```
-- 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
```

Fig. 5. 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'
```

Fig. 6. Sample rules for checking different kinds of data value constraints.

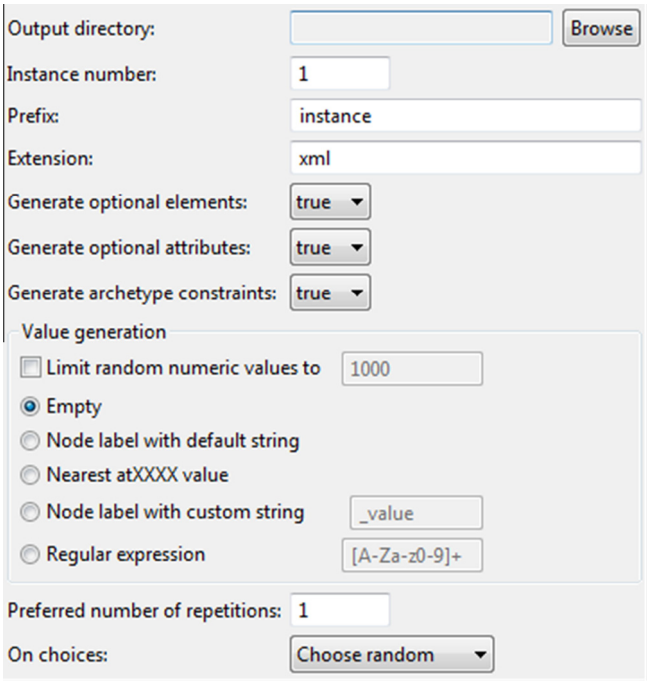


Fig. 7. Options for XML instance generation in LinkEHR.

```
<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'])&gt;=0 and
count(EN13606:parts[archetype_id='at0003'])&lt;=1"/>
  <assert test="count(EN13606:parts[archetype_id='at0004'])&gt;=0 and
count(EN13606:parts[archetype_id='at0004'])&lt;=1"/>
  <assert test="EN13606:items[@xsi:type='CLUSTER']"/>
</rule>
```

Fig. 8. Schematron rule for blood pressure measurement occurrences.

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 RM. 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 RM.

3.2.4. 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 and openEHR archetypes.

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 [43] 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.

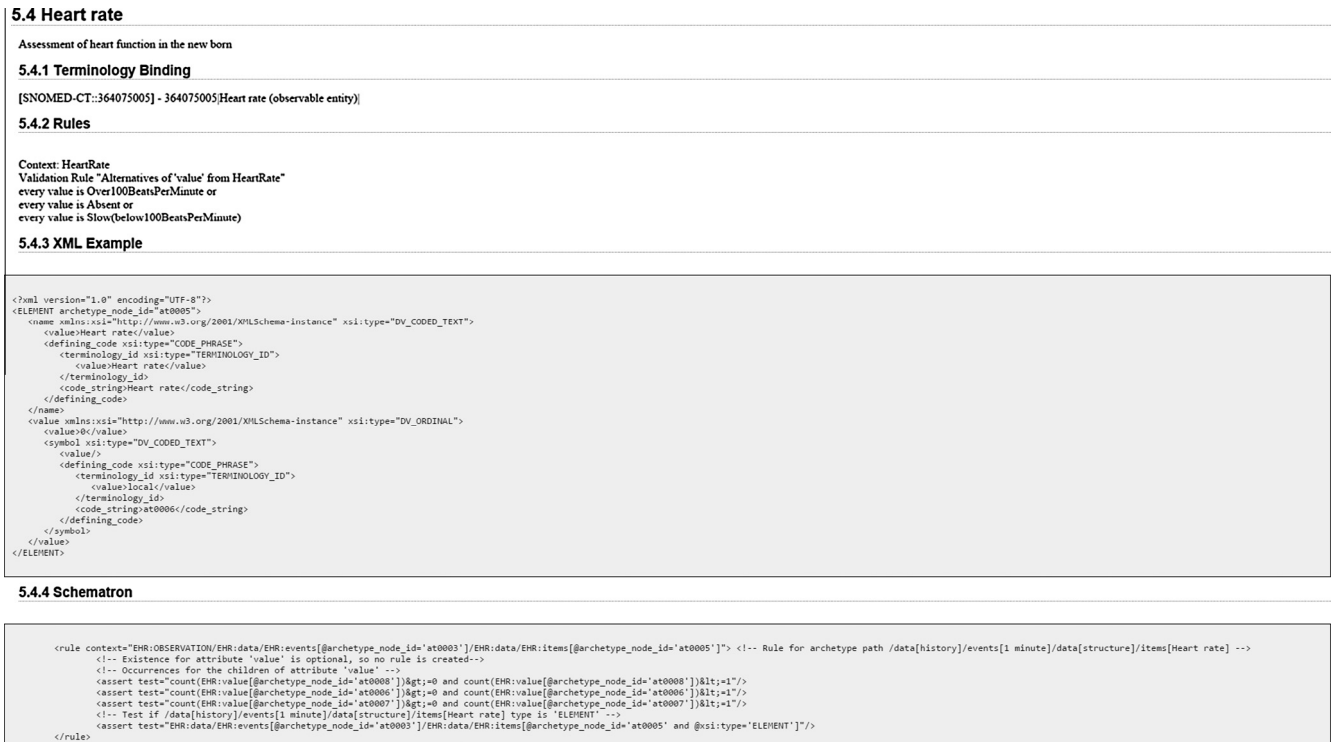
All the developed modules have been included in the LinkEHR platform in order to provide different export formats for archetypes. Each module uses both a set of configuration parameters and documentation about the reference model being to control the generation process and output appearance of the corresponding 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 [44] with improved terminology bindings. In the second example, we generated an implementation guide from an HL7 CDA archetype [45]. 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. Fig. 9 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.

In the second example we employed an archetype [45] created from the Genetic Testing Report (GTR) HL7 implementation guide [46]. The HL7 CDA archetype contains all the data constraints defined by the GTR implementation guide. Fig. 10 shows and excerpt of the original implementation guide, whereas Fig. 11 shows the same excerpt represented in the automatically generated guide.

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 Fig. 11 we have followed the first approach.



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)

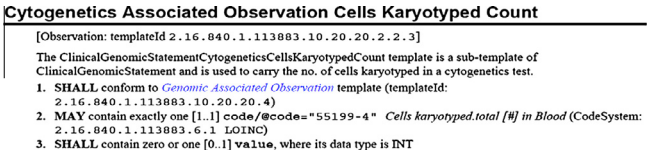
5.4.3 XML Example

```
<?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']"> <!-- Rule for archetype path /data[history]/events[1 minute]/data[structure]/items[Heart rate] -->
        <!-- Existence for attribute 'value' is optional, so no rule is created-->
        <!-- Occurrences for the children of attribute 'value' -->
        <assert test="count(EHR:value[@archetype_node_id='at0008'])&gt;=0 and count(EHR:value[@archetype_node_id='at0008'])&lt;=1"/>
        <assert test="count(EHR:value[@archetype_node_id='at0006'])&gt;=0 and count(EHR:value[@archetype_node_id='at0006'])&lt;=1"/>
        <assert test="count(EHR:value[@archetype_node_id='at0007'])&gt;=0 and count(EHR:value[@archetype_node_id='at0007'])&lt;=1"/>
        <!-- Test if /data[history]/events[1 minute]/data[structure]/items[Heart rate] type is 'ELEMENT' -->
        <assert test="EHR:data/EHR:events[@archetype_node_id='at0003']/EHR:data/EHR:items[@archetype_node_id='at0005' and @xsi:type='ELEMENT']"/>
    </rule>
```

Fig. 9. Excerpt of an automatically generated implementation guide from an Apgar score openEHR archetype.



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

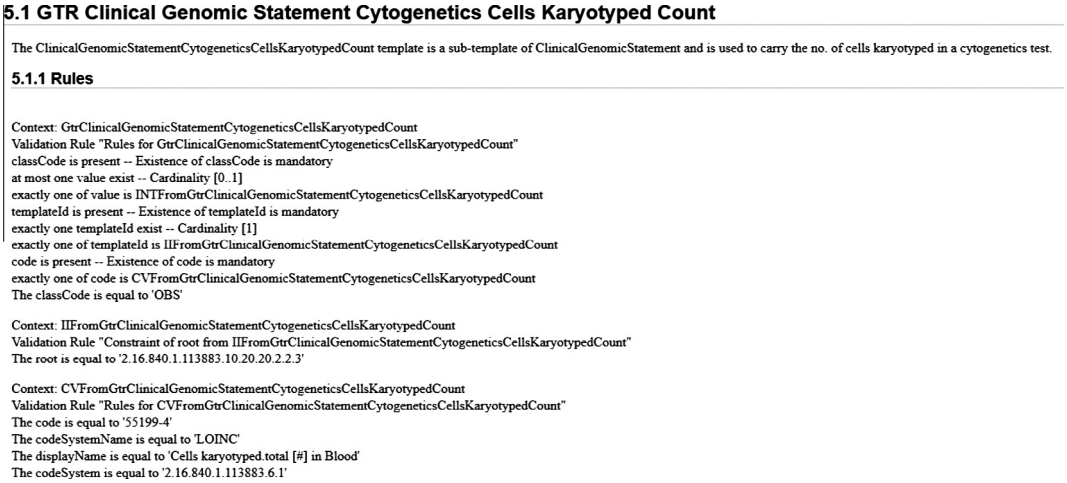
Fig. 10. Cells Karyotyped Count from the original Genetic Testing Report Implementation Guide.

4.1. Evaluation

We evaluate our methodology from three different perspectives: That generated instances are correct instances of a given model, that generated rules can correctly validate data instances, and that generated implementation guides have good quality and are useful for the development of EHR systems.

4.1.1. Generated instances are correct instances of a given model

The instance generation process is as follows: A constant archetype with the same structure as the source archetype is generated. This constant archetype defines a fixed single value for all atomic attributes. A constant archetype can be seen as a singleton archetype (i.e. an archetype that only defines one instance). This specialization relationship is formalized by means of a subsumption relation [47]. We say that an archetype A specializes a class B if B subsumes A. Based on this, a validation process exists for this subsumption relationship. We generated 10 constant archetypes for the each one of the Spanish Ministry of Health EHR project (Historia Clínica Digital del Sistema Nacional de Salud, HCDSNS) [48]. All the constant archetypes were validated with their corresponding archetype. All constant archetypes were found to be subsumptions of the source archetypes and thus are correct. For a deep



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'

Fig. 11. Executable rules from Cells Karyotyped Count automatically generated from the HL7 CDA Genetic Testing Report archetype.

discussion of the subsumption relationship and the existing implementation in LinkEHR we refer the reader to [42].

4.1.2. Generated rules can correctly validate data instances

We tested the Schematron rules source as we can execute them directly and they are equivalent to NLR rules [23]. To validate the rules we generated a set of random instances from one of the COMPOSITION archetypes of the Spanish Ministry of Health EHR project. These instances are demonstrated to be correct from the above validation process. These instances were modified to introduce common errors found in implementations [49] and the kind of errors detected by the meaningful use interoperability evaluation [6]. Introduced errors include incorrect data, incorrect structure, and terminology misuse. Data heterogeneity is also pointed as problematic in the meaningful use interoperability evaluation. This kind of problem is harder to validate as heterogeneous data does not imply that data is incorrect.

Incorrect data errors include wrong occurrences of elements, wrong existence of attributes, wrong values (values out of range, values outside a list of valid values, and values that do not follow a pattern), missing values, wrong element types, and wrong attributes for a given type. All these errors were detected by the generated rules.

Errors related to terminology misuse provide interesting examples. While missing or incorrect codes or code system are easily detected, discrepancies between codes and displayed names (e.g. the display name for RxNorm code 2670 is 'codeine', but the display name shows 'penicillin') are harder to detect. Our current solution does not support this. However, as archetypes allow the binding of terminology subsets to given values paths this could be tested if a terminology server is available.

Data heterogeneity is caused mostly for the loose constraints of the clinical information models. One advantage of using a formal definition to represent the clinical information models is that this heterogeneity can be reduced as desired (e.g. if the administration of a medication is 'every day' each implementer can choose to use either '24 hours' or '1 day'. This heterogeneity can be easily avoided by fixing the clinical information model to either hours or days, which would then be checked by the corresponding rules). Archetypes work as formal interfaces for data exchange to solve data heterogeneity problems.

4.1.3. Generated implementation guides have good quality and are useful for the development of EHR systems

Generated implementation guides were evaluated by clinical advisors of the Madrid region shared EHR project. They were considered as a useful resource and were provided as materials that have allowed the involvement of clinicians in the project. From a technical point of view the processable additional materials have speed up the development of some modules, concretely the validation of incoming data to the EHR repository and in the load test of such repository. Regarding the compliance of Detailed Clinical Models (DCM) quality metrics [40] by the generated implementation guides themselves, the metrics that can be measured are mostly the same as the ones that can be measured in archetypes. Our current generation process passes (complies with) metadata related metrics (such as DCM version, purpose, description, and authors), structure and value related metrics (valid atomic values, ranges, cardinalities, identification of data elements, etc.), and terminology related metrics (terminology binding, use of standard units, use of international standard terminology, translations, etc.). In addition to all the aforementioned metrics our generation process also complies with the 'multiple outputs' metric, as our methodology is able to generate multiple different artifacts with the same meaning.

Generated implementation guides are useful for the development of EHR systems as they share the structure with HL7 implementation guides. This ensures that the IT technical staff that currently uses implementation guides will be able to use them without further training. Each part of the generated implementation guides can be reused directly on end systems for testing and validation purposes.

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.

Generated implementation guides have the same structure and contents as hand-made implementation guides have. However, as generated implementation guides are based on a formal language, any section of it can be extracted for its direct use on information systems. Another critical point is the time and effort needed for the generation of the implementation guides. Hand-made implementation guides can potentially be really big (e.g. CCDA implementation guide [4] contains 500 pages) and its maintenance and correctness can be a difficult and time consuming task. Our methodology alleviates much of this creation, maintenance and validation tasks thus saving time and money.

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 [45,50]. This approach also solves common HL7 CDA problems [49] 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 objective.

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", and "MAY". 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 Fig. 10, both the "Observation" type and the templateId are not explicitly defined with a "SHALL" rule).

Regarding data validation, usually XML instances are validated against a schema (XML Schema, DTD, or RelaxNG) or Schematron. XML Schema and DTD are widely used in healthcare for the validation of clinical documents [51]. Archetype XML instances cannot be validated with older versions of XML Schema due to the problem of Unique Particle Attribution [52]. Regarding Schematron, our generation process is similar to the one described in [24], but we are able to generate Schematron for any EHR standard and not only for CDA. 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, such as validating only archetype constraints, or archetype and reference model constraints all together.

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 [53–59]. 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 [20], 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.

6. Conclusion

Implementation guides are one of the most common documents for the provision of clinical specifications for a 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. Existing quality metrics can be applied in order to measure both the source clinical information model or the generated implementation guide.

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.

As [6] demonstrates, current use implementation guides do not guarantee that each implementer generates outputs that are completely compatible. The use of archetypes as formal basis for the generation of implementation guides and reference artifacts allows implementers to directly test the systems and technical outputs by using the validated clinical models. The need for the involvement of the clinicians in the definition of clinical documentation practices in their organization is recognized by clinicians themselves [60]. A methodology like the one we proposed is needed to ensure that all clinicians can be involved in this process regardless of their IT expertise level.

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.

Conflict of interest

The authors declared that there is no conflict of interest.

Acknowledgements

We want to thank Pablo Serrano, planning director of *Hospital Universitario 12 de Octubre* for his comments and insights on the clinical aspects of our generated implementation guides.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jbi.2015.04.002.

References

[1] M. Reddy, W. Pratt, P. Dourish, et al., Sociotechnical requirements analysis for clinical systems, Methods Inf. Med. 42 (2003) 437–444, http://dx.doi.org/10.1267/METH03040437.
[2] T. Beale, Archetypes Constraint-based Domain Models for Futureproof Information Systems, 2000.
[3] R.H. Dolin, L. Alschuler, S. Boyer, et al., HL7 clinical document architecture, release 2, J. Am. Med. Inform. Assoc. – JAMIA 13 (2006) 30–39, http://dx.doi.org/10.1197/jamia.M1888.
[4] HL7 Implementation Guide for CDA® Release 2: IHE Health Story Consolidation, Release 1.1 – US Realm.
[5] Medicare and Medicaid Programs; Electronic Health Record Incentive Program – Stage 2, 2012.
[6] J.D. D'Amore, J.C. Mandel, D.A. Kreda, et al., Are meaningful use stage 2 certified EHRs ready for interoperability? Findings from the SMART C-CDA collaborative, J. Am. Med. Inform. Assoc.
[7] ISO 13606 Health Informatics – Electronic Health Record Communication – Part 1: Reference Model and Part 2: Archetype Interchange Specification.
[8] The openEHR Foundation. <http://www.openehr.org> (accessed 23.10.14).
[9] Archetype Definition Language 1.4 (ADL). <http://www.openehr.org/releases/1.0.2/architecture/am/adl.pdf> (accessed 23.10.14).
[10] Natural Rule Language (NRL) Specification 1.4.0. <http://nrl.sourceforge.net/spec/> (accessed 23.10.14).
[11] S. Garde, R. Chen, H. Leslie, et al., Archetype-based knowledge management for semantic interoperability of electronic health records, Stud. Health Technol. Inform. 150 (2009) 1007–1011.
[12] Object Constraint Language (OCL) Specification. <http://www.omg.org/spec/OCL/2.4/> (accessed 23.10.14).
[13] Object Management Group (OMG). <http://www.omg.org/> (accessed 23.10.14).
[14] Unified Modeling Language (UML). <http://www.omg.org/spec/UML/> (accessed 23.10.14).
[15] Web Ontology Language (OWL). <http://www.w3.org/TR/owl-ref/> (accessed 23.10.14).
[16] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. <http://www.w3.org/Submission/SWRL/> (accessed 23.10.14).
[17] Rule Markup Language (RuleML). <http://www.ruleml.org/> (accessed 18.07.14).
[18] World Wide Web (W3C) Rule Interchange Format. <http://www.w3.org/TR/rif-overview/> (accessed 23.10.14).
[19] Project Attempto. <http://attempto.ifi.uzh.ch> (accessed 23.10.14).
[20] A. Farkash, J.T.E. Timm, Z. Waks, A model-driven approach to clinical practice guidelines representation and evaluation using standards, Stud. Health Technol. Inform. 192 (2013) 200–204.
[21] N.E. Fuchs, U. Schwertel, R. Schwitter, Attempto controlled English—not just another logic specification language, in: P. Flener (Ed.), Logic-based Program Synthesis and Transformation, Springer, Berlin, Heidelberg, 1999, pp. 1–20. <http://link.springer.com/chapter/10.1007/3-540-48958-4_1> (accessed 23.10.14).
[22] ISO/IEC 19757-3:2006 Information Technology – Document Schema Definition Language (DSDL) – Part 3: Rule-based Validation – Schematron. <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html> (accessed 23.10.14).
[23] A. Alexandru, NRL to Schematron Generation Tutorial. <http://nrl.sourceforge.net/tutorials/schematron/tutorial.html> (accessed 23.10.14).
[24] K. Pfeiffer, G. Duftschmid, C. Rinner, Validating EHR documents: automatic Schematron generation using archetypes, Stud. Health Technol. Inform. 198 (2014) 101–107.
[25] Model Driven Architecture (MDA) Specifications. <http://www.omg.org/mda/specs.htm> (accessed 12.01.15).
[26] F. Truyen, The fast guide to model driven architecture – the basics of model driven architecture. Enero (2006). <http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf> (accessed 12.01.15).
[27] W.G. Michael van der Zel, Bridging the gap between software developers and healthcare professionals. Model driven application development, Hosp. Inf. Technol. Eur. 3 (2010) 20–22.
[28] J. Davies, J. Gibbons, S. Harris, et al., The CancerGrid experience: metadata-based model-driven engineering for clinical trials, Sci. Comput. Program. 89 (Part B) (2014) 126–143, http://dx.doi.org/10.1016/j.scico.2013.02.010.
[29] W. Raghupathi, A. Umar, Exploring a model-driven architecture (MDA) approach to health care information systems development, Int. J. Med. Inf. 77 (2008) 305–314, http://dx.doi.org/10.1016/j.ijmedinf.2007.04.009.
[30] V. Jones, A. Rensink, E. Brinksma, Modelling mobile health systems: an application of augmented MDA for the extended healthcare enterprise, in: 2005 Ninth IEEE International EDOC Enterprise Computing Conference, 2005, pp. 58–69. http://dx.doi.org/10.1109/EDOC.2005.22.
[31] E. Domínguez, J. Lloret, B. Pérez, et al., Model-driven development based transformation of stereotyped class diagrams to XML schemas in a healthcare context, in: J.-L. Hainaut, E.A. Rundensteiner, M. Kirchberg, et al. (Eds.), Advances in Conceptual Modeling – Foundations and Applications, Springer, Berlin, Heidelberg, 2007, pp. 44–53. <http://link.springer.com/chapter/10.1007/978-3-540-76292-8_6> (accessed 16.01.15).
[32] Marcos Menárguez Tortosa, Modelos de representación de arquetipos en sistemas de información sanitarios. <http://www.tdx.cat/bitstream/handle/10803/117386/TMMT.pdf?sequence=1>.
[33] J. Tepandi et al., Archetypes based development from the perspective of domain engineering research topics, (2012) 686–691.
[34] K. Atalag, H.Y. Yang, E. Tempero, et al., Model driven development of clinical information systems using openEHR, Stud. Health Technol. Inform. 169 (2011) 849–853.
[35] C. Martínez-Costa, M. Menárguez-Tortosa, J.T. Fernández-Breis, An approach for the semantic interoperability of ISO EN 13606 and OpenEHR archetypes, J. Biomed. Inform. 43 (2010) 736–746, http://dx.doi.org/10.1016/j.jbi.2010.05.013.
[36] Model-Driven Health Tools (MDHT). <https://www.projects.openhealthtools.org/sf/projects/mdht/> (accessed 23.10.14).
[37] D. Kalra, A. Tapuria, T. Austin, et al., Quality requirements for EHR archetypes, Stud. Health Technol. Inform. 180 (2012) 48–52.
[38] S. Ahn, S.M. Huff, Y. Kim, et al., Quality metrics for detailed clinical models, Int. J. Med. Inf. 82 (2013) 408–417, http://dx.doi.org/10.1016/j.ijmedinf.2012.09.006.
[39] F. Boterenbrood, I. Krediet, W. Goossen, Building a high quality medical data architecture for multiple uses in an integrated health care environment, J. Hosp. Adm. (2014) 3, http://dx.doi.org/10.5430/jha.v3n5p55.
[40] ISO 18864 Health Informatics – Quality Metrics for Detailed Clinical Models.
[41] LinkEHR Platform. <http://www.linkehr.com> (accessed 23.10.14).
[42] J.A. Maldonado, D. Moner, D. Boscá, et al., LinkEHR-Ed: a multi-reference model archetype editor based on formal semantics, Int. J. Med. Inf. 78 (2009) 559–570, http://dx.doi.org/10.1016/j.ijmedinf.2009.03.006.
[43] Indizen ITServer. <http://www.itserver.es> (accessed 23.10.14).
[44] J.L. Allones, M. Taboada, D. Martinez, et al., SNOMED CT module-driven clinical archetype management, J. Biomed. Inform. 46 (2013) 388–400, http://dx.doi.org/10.1016/j.jbi.2013.01.003.
[45] D. Bosca, L. Marco, V. Burriel, et al., Genetic testing information standardization in HL7 CDA and ISO13606, Stud. Health Technol. Inform. 192 (2013) 338–342.
[46] CDA Implementation Guide for Genetic Testing Report (GTR) (September 2011 Draft).
[47] G.M. Kuper, J. Siméon, Subsumption for XML types, in: J.V. den Bussche, V. Vianu (Eds.), Database Theory—ICDT 2001, Springer, Berlin, Heidelberg, 2001, pp. 331–345. <http://link.springer.com/chapter/10.1007/3-540-44503-X_21> (accessed 15.01.15).
[48] Recursos de Modelado Clínico (arquetipos). <https://www.msssi.gob.es/profesionales/hcdsns/areaRecursosSem/Rec_mod_clinico_arquetipos.htm> (accessed 16.01.15).
[49] Rene Spronk, Grahame Grieve, Common Issues Found in Implementations of the HL7 Clinical Document Architecture (CDA). <http://www.ringholm.com/docs/03020_en_HL7_CDA_common_issues_error.htm> (accessed 23.10.14).
[50] D. Moner, A. Moreno, J.A. Maldonado, et al., Using archetypes for defining CDA templates, Stud. Health Technol. Inform. 180 (2012) 53–57.
[51] K.W. Boone, Validating the Content of a CDA™ Document. http://dx.doi.org/10.1007/978-0-85729-336-7_20.
[52] Z. Tun, L.J. Bird, A. Goodchild, Validating Electronic Health Records Using Archetypes and XML, 2002.
[53] R. Chen, P. Georgii-Hemming, H. Ahlfeldt, Representing a chemotherapy guideline using openEHR and rules, Stud. Health Technol. Inform. 150 (2009) 653–657.
[54] N. Anani, R. Chen, T.P. Moreira, et al., 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. 14 (2014) 39, http://dx.doi.org/10.1186/1472-6947-14-39.
[55] G.M. Bacelar-Silva, R. Chen, R.J. Cruz-Correia, From clinical guideline to openEHR: converting JNC7 into archetypes and template, in: Anais do XIII Congresso Brasileiro de Informática em Saúde, Curitiba, Brazil, 2012. ISSN: 2178-2857.
[56] S.A. Barretto, J. Warren, A. Goodchild, et al., Linking guidelines to electronic health record design for improved chronic disease management, AMIA Annu. Symp. Proc. 2003 (2003) 66–70.
[57] M. Marcos, J.A. Maldonado, B. Martínez-Salvador, et al., Interoperability of clinical decision-support systems and electronic health records using archetypes: a case study in clinical trial eligibility, J. Biomed. Inform. 46 (2013) 676–689, http://dx.doi.org/10.1016/j.jbi.2013.05.004.
[58] A. González-Ferrer, M. Peleg, B. Verhees, et al. 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, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 71–84. http://dx.doi.org/10.1007/978-3-642-36438-9_5.
[59] D. Garcia, C.M.C. Moro, P.E. Cicogna, et al., Method to integrate clinical guidelines into the electronic health record (EHR) by applying the archetypes approach, Stud. Health Technol. Inform. 192 (2013) 871–875.
[60] T. Kuhn, P. Basch, M. Barr, et al., Clinical documentation in the 21st century: executive summary of a policy position paper from the American College of Physicians. Clinical documentation in the 21st century, Ann. Int. Med. (2015), http://dx.doi.org/10.7326/M14-2128.