eHealth2014 – Health Informatics Meets eHealth
A. Hörbst et al. (Eds.)

doi:10.3233/978-1-61499-397-1-101

Validating EHR Documents: Automatic Schematron Generation using Archetypes

Klaus PFEIFFER[a,1], Georg DUFTSCHMID[a] and Christoph RINNER[a]
[a]*Section for Medical Information Management and Imaging, Center for Medical Statistics, Informatics, and Intelligent Systems,Medical University of Vienna*

Abstract. The goal of this study was to examine whether Schematron schemas can be generated from archetypes. The openEHR Java reference API was used to transform an archetype into an object model, which was then extended with context elements. The model was processed and the constraints were transformed into corresponding Schematron assertions. A prototype of the generator for the reference model HL7 v3 CDA R2 was developed and successfully tested. Preconditions for its reusability with other reference models were set. Our results indicate that an automated generation of Schematron schemas is possible with some limitations.

Keywords. EHRs, Archetypes, Validation, Schematron, Health Information Management.

1. Introduction

Semantic interoperability is seen as an essential factor to improve the quality and safety of patient care [1]. In order to achieve semantic interoperability current Electronic Health Record (EHR) standards [2–4] are based on standardized logical information models. The instances of these models form the contents of EHRs exchanged in shared EHR systems like the Austrian ELGA [5]. The EHR in its generic form saves data about a patient and his/her state of health in a machine-readable form. Since the EHR documents are exchanged between different health care providers and the EHRs originate from different systems, it has to be assured that the exchanged EHR documents conform to the specified standards, guidelines or minimum requirements. The participating systems, which can use completely different software, agree on a particular "building plan" of the exchanged EHR documents. This mutual agreement is very important in further attempts to advance from the capability of systems to exchange information in a human readable format, called functional interoperability, to the capability of systems to exchange information in a machine readable format, so the receiving system can further process the data, called the semantic interoperability [6]. In order to enable an error-free and subsequent automatic processing of an EHR document in the receiving institution, it must be validated that the document actually satisfies all prescriptions of the before mentioned agreement.

The task of validating EHR documents has been analyzed in different projects. In [7] an approach for the semantic validation based on W3C XML schemas was

[1] Klaus PFEIFFER: kpfeiffer@gmx.at

presented. The authors identified several limitations of this technology. Amongst others, when deriving the XML schema from the before-mentioned "building plan" of an EHR document, the schema elements (and thus also the documents to be validated) have to be renamed to avoid violations of a particular XML schema specific constraint. Even though this renaming procedure can be typically automated, in some cases a manual parameterization is required. Further, XML schema does not allow a distinction of errors and warnings in the validation of EHR documents.

Besides XML schema, the validation of EHR documents is frequently also based on Schematron [8] that is also used in the forthcoming Austrian ELGA. Being an open standard and allowing its rules to be processed on any system with an XSLT processor available, Schematron suits the needs of the heterogeneous landscape of EHR systems well.

A clinical archetype is a best practice representation based on mutual agreement of a particular data structure within an EHR document. It is specified in the archetype definition language (ADL) and after being parsed is represented as an instance of the archetype object model (AOM). Visual archetype editors to create archetypes independently of the underlying reference model exist [9]. By defining constraints on EHR reference models, archetypes allow the requirements of structures and semantics of EHR documents to be modelled.

In this article we present an approach for the automatic generation of Schematron schemas from archetypes. The proof of concept implementation was tested with archetypes based on the ELGA implementation guides for the *discharge summarization notes (physician)* and *discharge summarization notes (nursing)* [10] and corresponding CDA documents. Also artificial archetypes representing the various types of constraints and corresponding documents were used for testing.

2. Methods

In Figure 1 an overview of the validation process is depicted. As input for the automatic generation of Schematron schemas from archetypes, ADL files (multiple

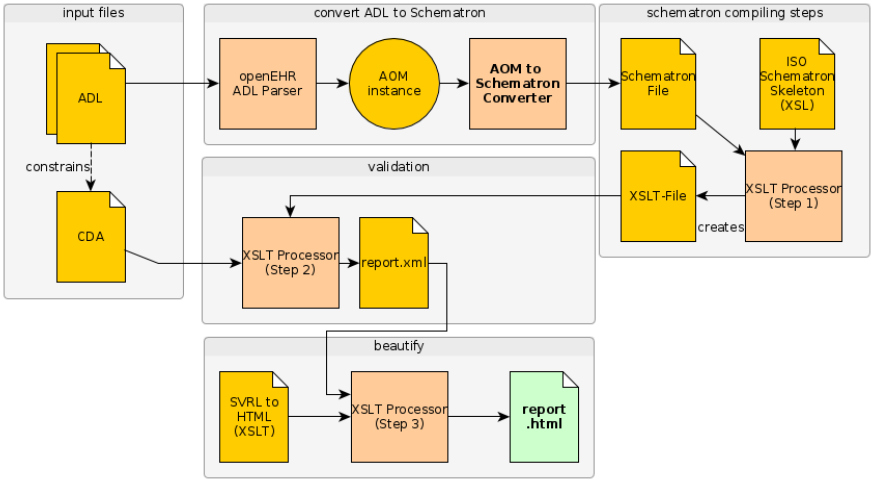

Figure 1. Overview of the different validation steps.

files in case of archetypes nested via slots) and a CDA document that should be validated are used. The ADL files are the starting point for the step “convert ADL to Schematron”. In this step the ADL file is parsed and the resulting AOM is used to automatically generate the Schematron schemas. The next step “Schematron compiling steps” consists of compiling the Schematron schema with the ISO Schematron Skeleton, which is itself written in XSLT, to an XSLT file. In the step “validation” we validate the input document using the created XSLT file. An XML file containing the validation report with elements described by the Schematron Validation Report Language is generated. Step “beautify” is optional and converts the output from the Schematron validation into the easier to read HTML format.

The AOM archetype nodes form a tree like structure and hold the data of the constraints along with the information of the context for the constraints. After the computation of the context elements, the tree elements contain all necessary information for the Schematron schema generation. In the following the key steps of converting the AOM instance to Schematron rules are described in more detail. As a simple example we use a constraint on attribute *string_attr1*, which prescribes the text “something” for this attribute. Using ADL this constraint is represented as follows:

```
templateId matches {
        II matches {
                root matches {"1.2.3"}
                assigningAuthorityName matches {"ELGA"}
        }
}
string_attr1 matches {"something"}
```

Using the openEHR ADL parser the ADL constraint is converted into an instance of the AOM. For a detailed description of how the various constraints are mapped to the corresponding AOM classes see “chapter 6.2 Class Descriptions” in [11]. The resulting AOM instance can easily be processed with a program language like Java. In the AOM instance, *string_attr1* holds instances of class *CString* and the prescribed value is put into a single-item list as can be seen in the following example.

```
org.openehr.am.archetype.constraintmodel.primitive.CString@36d26755[
    pattern=<null>,list=[something],assumedValue=<null>,
    defaultValue=<null>]
```

In the third step the AOM instance is converted into a Schematron schema (see following example). Hereby, an assertion about the presence or absence of the previously defined prescription is made. In our example it is tested whether attribute *string_attr1* actually holds value "something". Using the attribute *role*, the classification of the constraint is made. In our case, *role* is set to "error". Additionally a generic error message is added, which is later used in the generated report if this assertion is triggered.

```
<rule context="//*[hl7:templateId[@assigningAuthorityName='ELGA' and @root='1.2.3']]">
    <assert role="error" test="((lower-case(@string_attr1)=lower-case('something')) or
    (lower-case(hl7:string_attr1)=lower-case('something')))">Der Wert von string_attr1 MUSS
    'something' sein.</assert>
    <assert role="error" test="not(@string_attr1 and hl7:string_attr1)">string_attr1 darf
    entweder nur als XML-Element oder nur als XML-Attribut vorkommen.</assert>
</rule>
```

This Schematron schema can now be used to validate the XML file displayed in the following example.

```
<ClinicalDocument xmlns="urn:hl7-org:v3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance">
        <templateId assigningAuthorityName="ELGA" root="1.2.3" />
        <string_attr1>anything</string_attr1>
</ClinicalDocument>
```

The output of this validation is the report shown in Figure 2.

Schematron schemas can group various assertions into rules. The CDA attribute *templateId* is used to define the context of the rules applied to a specific archetype node.

Constraints on dates and times, as specified in the HL7 CDA, are converted to an XPath-compatible format. This task is achieved by using XPath string and date functions to convert the data in the EHR document from the ISO 8601 compliant representation without separators (e.g., 20131013) to a representation with separators (e.g., 2013-10-13).

The generation of the assertions is then accomplished by traversing the tree elements and transforming the information of every tree element into a Schematron assertion. For a detailed explanation of the meaning of the elements used in Schematron schemas see [12].

3. Results

In order to test the presented method, we chose previously created archetypes representing the ELGA implementation guides for the *discharge summarization notes (physician)* and *discharge summarization notes (nursing)* [10] and corresponding CDA documents. Furthermore we used a set of artificial test archetypes that include all types of constraints handled by our application and corresponding documents. Using object oriented software design principles and current XML technologies, a prototype was implemented. Oracle Java 1.6 was selected as the programming language. Apache

Figure 2. Example of a validation report (HTML output).

Maven [13], Apache Subversion [14], GIT [15], Checkstyle [16], JUnit, Lombok [17, 18] and the integrated development environment Eclipse were used to improve the quality of code and facilitate the reuse of the developed source code.

The Java reference implementation [19] provided by the openEHR foundation proved as a good basis for the generator. It could completely cover the first step of the automatic generation – the parsing of the ADL file to the archetype object model.

For the generation of the various assertions a set of constraint handlers were implemented for every type of constraint as depicted in Table 1.

The generator traverses the tree elements and chooses the corresponding constraint handler for the constraint specified in the archetype. The handler then generates the objects representing the tests by checking on the properties of the constraint class. If a property (e.g. the pattern property in the class CString describing the allowed regular expression this string value may hold) has a value, the test object – that already holds the information about the context, the XPath test expression and the report message – is generated and added to the list of test objects. From these objects the assertions and rules are derived and an assignment to a pattern, considering the templateId, is made.

The source code was made public at GitHub, which is a web-based hosting service for software development projects and is also used by the openEHR foundation. It can easily be extended with multiple repositories and projects and provides a comfortable interface to contribute to the project. It is a free service for open source projects. The repository can found under the URL https://github.com/klaus7/a2s

In Figure 3 the use of our tool is demonstrated for a scenario where two institutions exchange EHR documents. Institution A uses software A to manage its EHRs, whereas institution B uses software B with a different internal representation of the health data. Both institutions agree on a specific archetype definition to exchange their data. In this example institution A exports the document and validates the output with the archetype. In case of a negative validation, the document would have to be reviewed and corrected. In case of a positive validation it is sent to institution B. Not blindly trusting incoming data, the latter also validates the received document. Upon negative validation, the incoming data is rejected and the sending institution is notified. In case of a positive validation the EHR is imported into software B and can now be further processed.

4. Discussion

The benefits of our approach can be summarized as follows: Schematron is a widely known schema language in the domain of medical information processing, the modeling of archetypes is supported by various tools, and archetypes are a mature standards-based technology. Further, most problems reported in [7] when applying XML Schema for validating archetype-based EHR documents, such as the necessary renaming procedure and lacking distinction of errors and warnings (compare section 1), could be solved by using Schematron.

However, our approach also has some limitations. First of all, it obviously presumes that EHR document types are specified by means of archetypes. Even though archetypes play a significant role in current EHR research and have amongst others been adopted by EuroRec [20], the number of existing quality-assured archetypes is still rather small. Therefore, at the moment our approach requires that expertise for the development of archetypes is present or obtained.

Table 1. Overview of the constraint handlers implemented in the prototype of the generator.

Handler Class	Handles ...
CAttributeHandler	"required" and "not allowed" constraints.
CComplexObjectHandler	assertions for the occurrences of elements.
CMultipleAttributeHandler	assertions for the cardinality of elements.
CBooleanHandler	boolean type constraints.
CDateTimeHandler	constraints on dates and times.
CDurationHandler	constraints on time durations.
CNumberHandler	constraints on integers and real numbers.
CStringHandler	constraints on string values.

Due to the potential different representations of dates and times in different reference models and the XPath standard, a transformation has to be made and configured before the generation.

We do not yet handle invariants in our prototype, which allow dependencies to be defined between archetype nodes. We did not consider this a high priority as these constraints have been very rarely used in existing archetypes up to now.

Existing HL7 CDA implementation guides explicitly refer to the XML representation of CDA documents by distinguishing whether a constraint refers to an XML element or an XML attribute of the document. Archetypes, in contrast, only express constraints on object models but remain independent of the actual representation format of the EHR contents. Therefore they do not provide the language constructs to express whether a prescription must be manifested in an XML element or an XML attribute. Even though a disjunctive combination of element- and attribute-assertions can be generated for each archetype constraint, a document containing a prescribed value in an element would be positively validated when the value should actually be held by an attribute.

The current prototype operates with archetypes constraining the HL7 CDA. For the generator to work with other reference models, the following parts would have to be adapted: namespaces and document root of the generated file, the strategy for retrieving the context of the rules applied to a specific archetype node (equivalent of the templateId), and optionally the date and time format. We estimate that it should be possible to implement these changes within a few days of work.

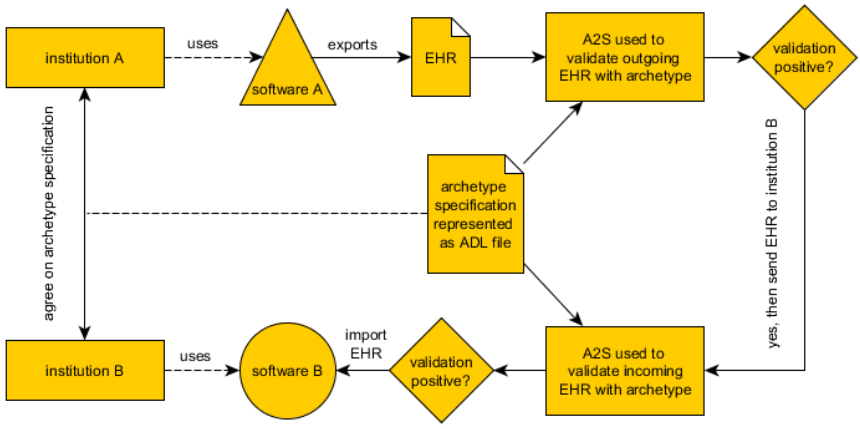

Figure 3. Validation scenario with two institutions exchanging EHR documents.

References

[1] V. N. Stroetmann, D. Kalra, P. Lewalle, A. Rector, J. M. Rodrigues, K. A. Stroetmann, G. Surjan, B. Ustun, M. Virtanen, and P. E. Zanstra, "Semantic interoperability for better health and safer healthcare. Deployment and research roadmap for Europe." European Community, 2009.

[2] R. H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. M. Behlen, P. V. Biron, and A. Shabo Shvo, "HL7 Clinical Document Architecture, Release 2," *Journal of the American Medical Informatics Association*, vol. 13, no. 1, pp. 30–39, 2006.

[3] International Organization for Standardization, "ISO 13606. Health informatics - Electronic health record communication", 2008.

[4] T. Beale, S. Heard, D. Kalra, and D. Lloyd, "The openEHR reference model: EHR information model," 2008. [Online]. Available: http://www.openehr.org/programs/specification/releases/1.0.2.

[5] S. Herbek, H. A. Eisl, M. Hurch, A. Schator, G. Rauchegger, A. Kollmann, T. Philippi, P. Dragon, and E. Seitz, "The Electronic Health Record in Austria: a strong network between health care and patients," *European Surgery*, vol. 44, no. 3, pp. 155–163, 2012.

[6] International Organization for Standardization, "ISO/TR 20514:2005 – Electronic health record, definition, scope, and context", 2005.

[7] C. Rinner, S. Janzek-Hawlat, S. Sibinovic, and G. Duftschmid, "Semantic validation of standard-based electronic health record documents with W3C XML schema," *Methods of Information in Medicine*, vol. 49, no. 3, p. 271, 2010.

[8] K. Boone, "Validating the Content of a CDA™ Document," in *The CDA TM book*, Springer London, 2011, pp. 275 – 281.

[9] 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.

[10] ELGA GmbH, Harmonisierungsarbeit für medizinische Dokumente - ELGA CDA-Implementierungsleitfäden [Online]. Available: http://elga.gv.at/index.php?id=28 [Accessed: 21-Jan-2014].

[11] Beale T. The openEHR Archetype Model: Archetype Object Model. 20-Nov-2008 [Online]. Available: http://www.openehr.org/releases/1.0.2/architecture/am/aom.pdf [Accessed: 16-Mar-2014].

[12] Jelliffe R. ISO Schematron Specification - free official version. 2006 [Online]. Available: http://www.schematron.com/spec.html. [Accessed: 09-Mar-2013]

[13] The Apache Software Foundation, "Apache Maven." [Online]. Available: http://maven.apache.org/. [Accessed: 09-Mar-2013].

[14] The Apache Software Foundation, "Apache Subversion." [Online]. Available: http://subversion.apache.org/. [Accessed: 06-Jan-2013].

[15] "GIT." [Online]. Available: http://git-scm.com/.

[16] D. Schneider and K. Lars, "Checkstyle (eclipse-cs)." [Online]. Available: http://eclipse-cs.sourceforge.net/. [Accessed: 09-Mar-2013].

[17] Jappe van der Hel, Philipp Eichhorn, Reinier Zwitserloot, Robbert Jan Grootjans, Roel Spilker, and Sander Koning, "Project Lombok," 2009. [Online]. Available: http://projectlombok.org/. [Accessed: 21-May-2013].

[18] M. Kimberlin, "Reducing Boilerplate Code with Project Lombok." Jan-2010.

[19] R. Chen and G. Klein, "The openEHR Java reference implementation project," Studies in health technology and informatics, vol. 129, no. Pt 1, pp. 58–62, 2007.

[20] "EuroRec - EHR Archetypes," EHR Archetypes, 12-Jan-2014. [Online]. Available: http://www.eurorec.org/services/archetypes/index.cfm. [Accessed: 19-Jan-2014].