Model based Validation of XML Requirements, Applied on Healthcare IT Standards

Abderrazek Boufahja[1] and Eric Poiseau[2]
[1]*Gazelle Team, Kereval / IHE-Europe, Rennes, France*
[2]*Gazelle Team, Inria Rennes / IHE-Europe, Rennes, France*

Keywords: Model-based, Validation, Requirements, XML, Healthcare Standards, Schematron, Interoperability, Quality.

Abstract: Nonconformity of healthcare implementations to the medical standards has become a real source of troubles and loss of interoperability between systems. Healthcare documents frequently contain inconsistent requirements related to the standards they must conform to. Few standards and methodologies exist to deal with complex requirements, and often they are only dedicated to some specific kinds of healthcare standards, like CDA, HL7 and DICOM. The complexity of standards and their constant evolution have made difficult the implementation of robust check methods and tools for healthcare documents. In this paper, we propose a novel model-based validation methodology, which allows enumerating and validating requirements related to healthcare documents that have XML based structure. Model-based methodology defined and specified in this paper allows checking any kind of requirements even for healthcare documents with complex standards' cascading. Experimentations of model based checking demonstrate that this method was highly effective in detecting inconsistencies, and orienting implementers of healthcare technologies.

1 INTRODUCTION

Numerous regional and national healthcare initiatives are requiring eHealth applications to be tested for conformance to profiled standards. Meaningful Use (Melissa Markey, 2012) in the USA, Elga (Georg Duftschmid, 2009) in Austria or ASIP (ASIP, 2012a) in France illustrate the desire of healthcare organizations to build an infrastructure to share PHR. Most of those initiatives are profiling the HL7 CDA (HL7, 2005a) standards and/or the IHE XDS (IHE, 2012a) profiles specifications. CDA specifies how to structure and code health records while XDS provides the sharing mechanism of these records.

Testing the conformance of healthcare solutions becomes an important issue in order to achieve the interoperability of the different components contributing to the sharing of such documents. Since both XDS messages and CDA documents have an XML structure, we have developed a methodology that allows the conformance checking of XML documents in the context of hybrid healthcare standards. This methodology is based on UML modeling and MDA approach to describe the specifications. The main idea of this methodology is to inject constraints into UML models instead of executing xpath rules into XML documents. The result of this methodology is a report of validation generated automatically from the models describing the healthcare standards.

Following a review of existing solutions for validating XML documents in the field of healthcare we will present our methodology and its evaluation based on its use on real life project like epSOS (Thorp, 2010) (epSOS, 2013).

2 STATE OF THE ART

Numerous tools provide validation of healthcare standards based XML technology. However, most of them are based on Schematron standard and MDHT project's methodology. This section provides an analysis of these methods.

2.1 Schematron

2.1.1 Presentation

Schematron is an ISO/IEC standard that offers the possibility to create complex rules, like conditional rules, iterations over nodes, nodes content comparison, etc. This set of schematron rules can be used to

Boufahja A. and Poiseau E..
Model based Validation of XML Requirements, Applied on Healthcare IT Standards.
DOI: 10.5220/0004776902950302
In *Proceedings of the International Conference on Health Informatics* (HEALTHINF-2014), pages 295-302
ISBN: 978-989-758-010-9

check the conformance of an XML document. Many groups are using schematron rules for the validation of XML documents (Lee, 2000). There are open source tools that allow validating an XML document regarding a schematron, and they are widely used and maintained by a large community, offering some trust in the performance and the quality of the standard. A schematron document is basically composed of multiple xpath assertions, which represent the rules that the tested XML document shall verify (Clark and DeRose, 1999).

2.1.2 Strengths and Weaknesses

Schematrons are efficient when used for few rules, and they are simple to use as they are an interpreted language. However, schematrons have multiple weaknesses. First the processing of the schematrons is too long. When hundreds of rules are to be verified, the validation takes too much time. The second problem is the use of xpath as language of constraint. Xpath is known to be a difficult language and hard to debug (Clark and DeRose, 1999). Also, the code defined by hundreds of xpath rules is not easy to read and to understand. As a consequence, it is not easy to maintain a set of schematron rules. Another weakness of schematrons, there is no real coupling between requirements and rules. Moreover, there is no automatic generation of documentation. We cannot know for an existing schematron if a requirement is checked or not. Furthermore, there is no unit testing generation for the written rules. Finally, there is no support for changeable value sets. In healthcare standards, the list of technical keywords grows frequently. There are no techniques on the schematrons that allows using a dynamic validation regarding to a dynamic list of a standard value sets.

2.2 MDHT Project

2.2.1 Presentation

MDHT is an open source tool developed and maintained by Open Health Tools (Sondra Renly, 2012), a not-for-profit Trade Association, incorporated in the United State of America. MDHT is dedicated to document and to validate medical IT standards that are based on HL7 CDA standard (HL7, 2005a). The aim of the project is to offer a tool that allows the description of the requirements related to healthcare specifications based on HL7 CDA standard, in a simple UML model. As output, the tool provides a documentation of the requirements, the java source code for the manipulation (read and write) of CDA documents, and a validator of CDA documents related to the standard described in the UML model. MDHT provides an eclipse plugin for the edition of the underlying UML model. A large community uses MDHT as a reference of development and for requirements validation and documentation (example: CCD, IHE, HITSP, etc).

2.2.2 Strengths and Weaknesses

MDHT is an excellent tool for CDA document validation with a large user base. However the tool is designed for CDA and can not be easily extended to the validation of XML documents which are not CDA documents, like XDS messages or XDW documents (IHE, 2012b), as the profile of UML stereotypes that describe the UML models is based on the structure of CDA documents.

3 MODEL BASED VALIDATION OF HEALTHCARE DOCUMENTS

3.1 Objectives

Our goal is to create a method that allows describing formally all requirements under an XML healthcare standard. This method should be generic and support inheritance between standards. The performance of this method shall be comparable to the performance of schematrons, and even better. This method shall provide a documentation and a coverage of the implemented requirements. The maintainability of the tools and models shall be better than schematrons maintainability. The method described here provide also generated unit testing for each implemented requirement, which is a huge advantage comparing to schematrons. The outcome of the proposed methodology is to provide a UML model and a set of methods used to convert it into documentation, validators and test procedures (Hans-Erik Erikson, 2004).

Standards Inheritance: We can point as example two standards that share the same basic parent standards, and extend them with more rules and requirements. These standards are the Swiss CDA Laboratory documents (eHealth Suisse, 2013), and the French CDA Laboratory documents (ASIP, 2012b). These two kinds of documents are based on the standards below, described on the figure 1. The two standards have a common base, and differ only on the top of the pyramid of standards. The method described here takes care of this kind of inheritance, and allows having shared models for shared standards.

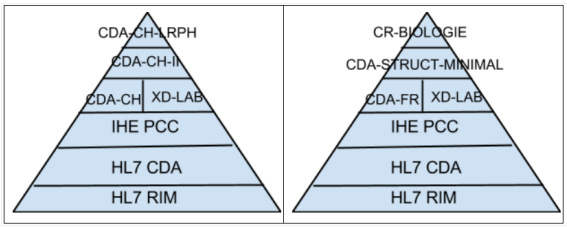

Figure 1: Inheritance between healthcare standards.

3.2 Principle

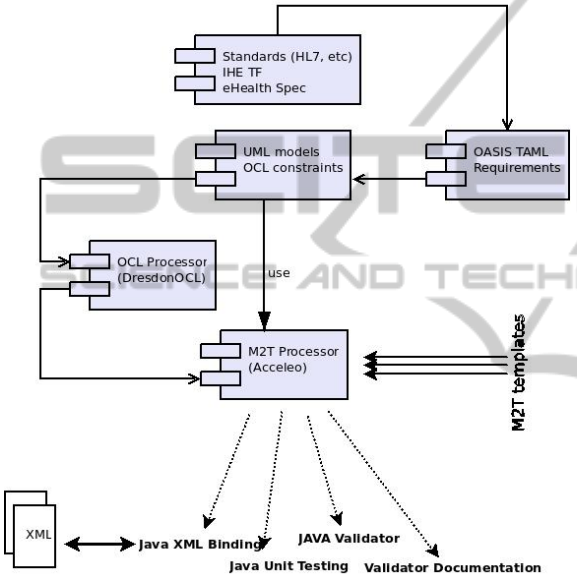

Figure 2: Principle of model based validation of XML healthcare documents.

The principle of the method that we propose for the validation of XML documents based on UML description, is the following (figure 2):

1 - From medical standards like HL7, DICOM (Hongli Lin, 2010) and IHE standards, we extract all requirements, and we insert them into a specific UML model, which has a specific structure, that we will describe later. The UML model contains constraints written in OCL (Object Constraint Language) (OMG, 2012). The purpose of this language is to describe the relationship between elements of the UML model, which can not be simply described by diagrammatic notation. Each OCL constraint represents a requirement on a medical standard. OCL is a powerful language that permits many variants of constraints, like loops, search constraints, conditional constraints, etc. By our experience on more than 50 validators of IHE documents, OCL can generate the description of any kind of rules related to the model. The created UML model contains also the structure of the XML document. This structure allows linking the OCL constraint to its corresponding XML element.

2 - The OCL constraints are then processed to a programming language code like JAVA. In the industry, there are many processors of OCL. The most popular one is DresdenOCL, which is a library developed and maintained by students and scientists of the Software Technology Group at Dresden University of Technology (Birgit Demuth, 2009) (Birgit Demuth and Zschale, 2004).

3 - The UML models and the processed OCL constraints are then used by a UML model to text generator (M2T), to generate a specific validator based on the UML contents (OMG, 2008). There are many projects that present themselves as a UML model to text tool, the most popular one is Acceleo (OMG, 2008). Acceleo is not a classic generator of code from UML models, like EMF generator. To generate code you have to provide some M2T templates, which describe the generated text from the model. The generated text can be of any kind: html, java, or C++ for example. The idea was to generate java code that allows transforming the XML document to be validated from XML to JAVA instances, and then to validate these java instances by using the code generated by the OCL processor. The M2T method allows also to generate a documentation of the UML model, and unit tests for constraints written on OCL. Each module is described by its M2T templates.

3.3 Healthcare Requirements Management

Requirements on healthcare documents are generally written on human language, not a formal one. A major problem of the maintainability of schematrons was the fact that once the schematron written, we do not know which requirements are described on it, and which ones are not. We have no information about the coverage of rules written on schematrons, according to requirements from healthcare specifications. Many tools offer the possibility to list requirements, like TestLink for example. The one that we chose was an OASIS standard: taml (OASIS, 2011). This standard is a common structure for defining requirements. We used this standard and restricted its structure to better conform to the context of requirements on healthcare documents. This standard allows specifying the list of predicates, and for each predicate you can specify a list of tags which describe the predicate. We restricted the tags to: 'section' and 'page', which describe the section on the document and the page that refer to the requirement. Each predicate is defined by a unique identifier

(/taml:testAssertion/@id), and each list of requirements is defined by a unique identifier: /testAssertionSet/common/normativeSource/target/@idscheme.
We restricted the OASIS taml standard by imposing that the common element of the taml:testAssertionSet shall be present, and this common element shall contain information about the healthcare document that we are processing, like the document name, version, source name, and a URI to the original document. These properties are included in the element taml:common/taml:refSourceItem. We forced the taml descriptor to have a unique identifier by adding the element taml:common:/taml:target. This is an example of taml assertion token from the taml document of CDA PADV specification:

```
<testAssertion id="CONF-5">
<predicate>The Pharmaceutical Advice section
SHALL contain code.</predicate>
    <prescription level="mandatory"/>
    <tag tname="Section">6.3</tag>
    <tag tname="Page">19</tag>
</testAssertion>
```

Each requirement is identified by a unique couple (target@idscheme , testAssertion@id). So we defined a stereotype applied on UML constraint elements, which describes the relationship between constraints and requirements (figure 3). It contains two attributes: IDs and targetIDScheme, where IDs represents the list of ID on the taml document, and targetIDScheme is the identifier of the taml document.

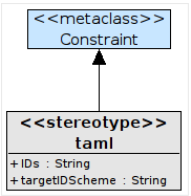

Figure 3: Relationship between rules and requirements.

This correlation between constraints and requirements allows calculating the coverage of the validator according to the list of requirements, and then to identify requirements that are not implemented in the UML models.

3.4 UML Models' Specification

3.4.1 Constraints' Specification

Each requirement extracted from the specifications of the XML document is translated into a UML constraint that contains an OpaqueExpression element with the attributes:

- language: 'OCL'
- body: the OCL constraint

The OCL constraint shall always have the result equals to true when applied to an UML instance specification. On healthcare standards, especially on HL7 ones, there are three kinds of rules: requirements, warning, and notes, which are specified by the keyword SHALL, SHOULD and MAY (Bradner, 1997). We specified a stereotype applied on UML constraint elements, which allows to specify if the constraint is an error, a warning, or a note. This stereotype is named ConstraintType (figure 4).

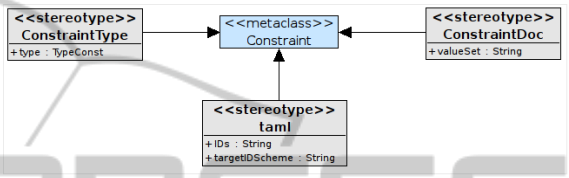

Figure 4: Stereotypes related to Constraint element.

We defined also a stereotype to document a constraint when it is related to a valueSet: a dynamic list of values, that can be provided by a CTS or a SVS provider (IHE, 2010) (HL7, 2005b) (Heymans S, 2011). Each created constraint element shall be related to a UML class element.

3.4.2 Classes' Specification

There are two kinds of UML classes in this methodology: classes used to describe the content of the XML document, and classes used to apply a list of rules on a kind of XML element.

1- Classes used to Describe the Content of the XML Document: These classes, called in our specification StructureClass (SC), contain attributes with the same structure described in the schema of the XML document. The profile used to describe the relation between classes and the schema elements is 'Ecore' profile (Dave Steinberg, 2008). From an XSD we can generate the UML model containing the description of the content of the XML document. The principal stereotypes used from this profile to describe the XML structure are: EPackage, Eclass, EEnum, EAttribute, and EReference. The generation of code binded to XML is based on these stereotypes. On this kind of classes, basic constraints can be included. If we are sure that a rule shall be applicable to any restriction of the standard, and it is not related to some specific context, we can add it directly on the class of description of the element.

2- Classes used to Apply Rules on a kind of XML Element: When we are on a special specification of a standard, or on an affinity domain which restrict the original standard, like for example epSOS

CDA standard, we know that rules applied by these standards are not absolute, and we can not attach these rules directly to the class of description of the element. We defined the notion of 'package of constraints'. Each package of constraints contains a list of classes of constraints, and each class of constraints contains a list of constraints, has a generalization to the parent UML StructureClass or to another class of constraints, and has a stereotype that defines the kind of the class of constraints.

We defined three kinds of stereotypes that describe the kind of a class of constraints (figure 5):

- TemplateSpec (TS): described by a (path, id). The list of constraints on this class is applied only when the value of the path on the XML instance has the same value as the id
- ConstraintsSpec (CS): the list of constraints is applied automatically to any instance of the element described by the parent class
- AdvancedTemplate (AT): defined by an OCL rule. The list of constraints on this kind of classes is applied only when the specified rule is verified on an instance of the parent class.

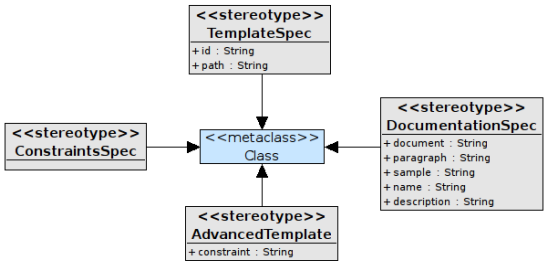

Figure 5: Stereotypes of classes of constraints.

The UML classes that are described by these stereotypes are created manually in the UML model, in order to provide restrictions on the StructureClass (SC) classes, using UML constraints that describe requirements of the specifications. However, the StructureClass(SC) classes are created automatically from the XSD schema that describes the structure of the XML documents to be validated (R. Bhuvaneswari, 2012). This schema is generally provided by the standard specification.

The mixing of these kinds of classes of constraints can lead us to illogic situations, by generalization from one kind to another. So we defined some rules of inheritance between TemplateSpec (TS), ConstraintsSpec (CS), AdvancedTemplate (AT) and StructureClass (SC).

When a class of constraints generalizes another class of constraints, parent class rules are added to the child class rules, and the two lists of rules are executed only if we can execute the two lists in the same time. If we allow a TemplateSpec to inherit from another TemplateSpec, the rules of the child are executed only if the element tested on the XML document verifies the two paths of the parent and the child classes. If one of these paths is not verified, the rules are not executed, and here we have a problem because no error is reported to the designer of the XML document, showing that the XML element is missing a path, one or the other. For this reason, the table below specifies this kind of relationship between classes of constraints (table 1).

Table 1: Inheritence between classes of constraints.

–	TS	CS	AT	SC
TS	x	x	x	v
CS	v	v	v	v
AT	x	x	x	v
SC	x	x	x	v

We described these relationships by a formal grammar $G = (N, \sigma, Y, P)$, where:

- Alphabet: $\sigma = \{TS, CS, AT, SC\}$
- Nonterminal symbols: $N = \{Y\}$
- Start symbol: Y
- production rules P:

$Y \rightarrow TS.SC$
$Y \rightarrow AT.SC$
$Y \rightarrow Y.SC$
$Y \rightarrow CS.(SC|Y)$

This can be described by the diagram below (figure 6).

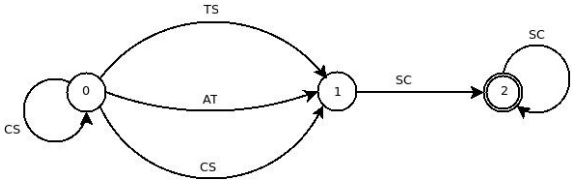

Figure 6: State diagram of classes of constraints' relationship.

The language defined by G is:

$$L(G) = \{CS^n(TS|AT|CS)SC^m; n, m \in \mathbb{N}, n \geq 0, m \geq 1\} \quad (1)$$

We also defined a stereotype to document classes of constraints, named DocumentationSpec. It allows specifying information about the standard which is the origin of constraints, and this provides a better documentation of the class of constraints when generating the documentation of the model of constraints.

3.5 M2T Generation's Specification

3.5.1 XML Binding

As explained on the principle of this method, as output we need to generate a code that allows transforming XML elements to object instances. In the implementation of this methodology, we chose JAXB as the API to bind XML to objects (McLaughlin, 2002). Oracle provides a tool, named xjc (McLaughlin, 2001), which allows to generate from an XML schema, java classes containing a full description of the XSD elements, based on JAXB annotations. The same functionality of generation of code containing a binding with XSD elements is done by a M2T template. Creating our own generator of java code gives us the possibility to add further methods and attributes, that are not generated by xjc. For example, we have introduced the ability to validate xpath constraints as a method of validation on the generated java code.

3.5.2 Validation Code

The second kind of output from UML models is the classes of validation. This generated code is the combination of the OCL code transformed and processed by the OCL processor, and the generated code from a M2T template that links rules between themselves. The result of the execution of the generated code is a list of verifications. This list contains errors, warnings, notes and reports about processing of rules. For each package of classes of constraints, we generate a class of validation. This technique allows the reusability of the generated code, and simplifies the imbrication of standards. For example, in the laboratory domain described in figure 1, for each block in the two pyramids, we define a package of constraints. So for each of HL7 CDA, IHE PCC and XD-LAB, we define a unique package of validation. Then we define specific packages for CDA-FR, CDA-CH, and LAB-FR, LAB-CH. The validator is the combination of the common packages and the specific packages. The M2T templates generate code based on the visitor pattern: for each element on the object structure generated to describe the elements of the XML, we pass an instance of the package of validation, to a method generated from the template of generation of structured classes (figure 7).

This method verifies the rules of the package on the current instances, and calls the same method on its attributes with the same package's instance.

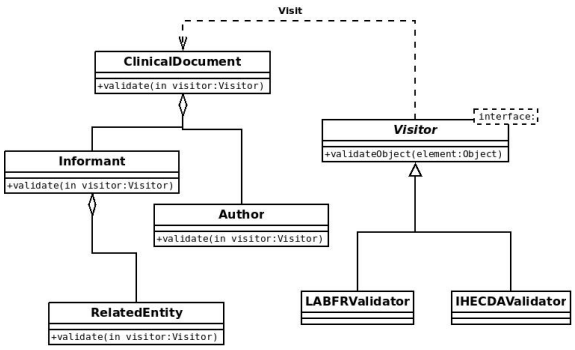

Figure 7: State principle of the visitor pattern applied on XML elements validation.

3.5.3 Unit Testing Generation

The generation of unit tests is managed by a M2T template to facilitate the process of unit testing. This feature does not exist on the schematrons' process. The generated code provides for each constraint, two tests: one OK and one KO. The principle of each test is to validate a whole document, and to verify if the result of the constraint is what it is supposed to be. The specification of the XML documents to be verified can not be done automatically; it is the role of the tester to provide them.

3.5.4 Templates Coverage

As we define a structure of templates and advanced templates, the specified model of constraints provides an overview of the complexity of the XML document. This feature is very useful in CDA documents and in XDS metadatas analysis, as the specifications of these kinds of documents are based on templates structure. For example, in CDA documents, sections and entries have an attribute named templateId which is a unique identifier of the kind of the element, and it is referenced by the TemplateSpec stereotype (HL7, 2005a).

3.5.5 Documentation Generation

The defined stereotypes to document constraints and classes of constraints are used for the generation of the documentation. The generation is performed using a M2T template, with an html output. There are two kinds of documentation that can be generated:

- the documentation of the structure of the XML document: this documentation is a description of the elements of the XML document. For each element we can document the cardinality, the type, the name, and the parent.

- the documentation of classes of constraints: this documentation contains the relationship between con-

straints and classes, documentation of the kind of constraints and the kind of classes of constraints, and finally a documentation of the link between constraints and taml assertions.

4 IMPLEMENTATION

4.1 Implementation Details

We have implemented and used this methodology for the development of validators used by the Gazelle Test Bed plateform (Eric Poiseau, 2010). The UML editor used for the implementation is Topcased: it offers a rich graphical editor, and provides an efficient search tool for an easy retrieval of constraint or the attributes of a stereotype. The OCL processor used is DresdenOCL and the M2T generator used is Acceleo, as they are open source tools. The generated code for the validation and testing is in JAVA and the generated documentation is in HTML. Acceleo templates are packaged as an eclipse plugin, so a developer who want to use the tool to create a validator for a specific standard only needs to upload the plugin and to write the UML models, and then to automatically generate the java code of validation, the unit test code and the documentation. So there are no further acceleo templates to write.

4.2 Developers Feedback

Developers who have used the model based tool to create and update UML models were pleased to the fact that OCL was easier to use than xpath on schematrons. The generated unit tests give a kind of confidence on the generated code, and the generated documentation is helpful for debugging when the validation fails. Also, UML models are easier to maintain than schematrons. Developers at IHE-Europe have tested this methodology and adopted it for numerous IHE profiles. At the date of this publication, the number of model-based validators implemented by IHE Europe is close to 80, and it is growing.

4.3 Users Feedback

The model based validation methodology was applied to perform the validation of messages and documents in various standards used in the field of Healthcare. Beside the use within the epSOS project for HL7 CDA and XDS metadatas validations, we have applied the methodology to other specifications in the IHE domain. Today we applied it to many of the XML based profiles, like DSUB, XDS, PDQV3, HPD, and SVS. Users of these validators were satisfied by the robustness of the validation. Moreover, sometimes the implementation of model based validators allowed us to detect inconsistencies between the specifications and the referenced standards.

4.4 Performance Issues

The implementation of this methodology has proved that the model-based validation is easier to maintain and to reuse than schematrons. Furthermore, the model-based validators are faster than schematrons.

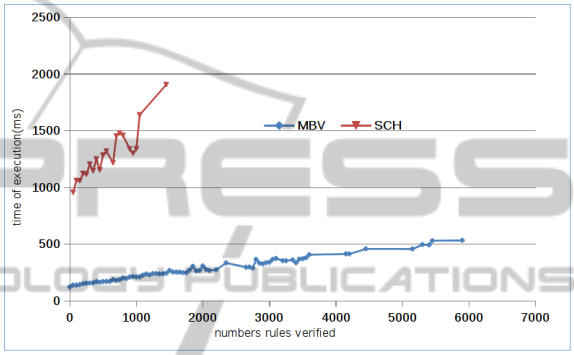

Figure 8: Comparison between execution time (ms) of the validation using model based validation and schematrons.

The figure 8 reports the execution time for the validation of a HL7 CDA documents on the epSOS domain. Red curve represents the execution time of schematrons (SCH) according to the number of rules verified on a document, and the blue curve represents the execution time for the model-based validation (MBV) according to the number of rules verified. 1300 samples CDA documents were processed in order to plot these curves. The samples were provided from 20 national infrastructures of the European member states participating in the epSOS project (McLaughlin, 2002). One can remark that for the same number of rules processed the time of MBV is smaller than for schematrons. In addition, as the number of rules increases the execution time is rising faster for the schematron validation than for the model based one.

5 CONCLUSIONS

In this paper, we designed a methodology of validation of XML documents on healthcare standards based on model based architecture. This methodology has allowed to remain with weaknesses of schematrons technology, especially problems of maintainability, reusability, unit testing, documentation and re-

quirements coverage. It has also simplified the validation of pyramidical standards.

The implementation of this methodology was done using open source tools, especially Topcased as editor, DresdenOCL as OCL processor, and Acceleo as M2T generator. The generated output code was on Java technology. The result of this implementation has covered the needs of developers and users of the generated validators. The use of this methodology to create validators for multiple kinds of healthcare standards in many domains like epSOS and IHE has proved the efficiency of this method: any kind of constraint can be expressed. And also, an important feature was, the model based validators are quicker than schematrons.

Several improvements could be injected into this methodology and its implementation, like a self editor of the UML model, to simplify the creation and the management of classes and constraints. Also, the concept of stereotypes to describe classes and constraints can evolve to a meta-model that describes this set of stereotypes, and a use of GMF can improve the usability and minimize the risk of inconsistencies of UML models. Moreover, the model based validation of XML based healthcare standards could be adapted to other domains that use the XML technology.

REFERENCES

ASIP (April 25, 2012a). *Cadre d'interoprabilité des SIS, Document chapeau*. Agence des systèmes d'information partagés de santé.

ASIP (October 15, 2012b). *Volet Compte Rendu dÉxamens de Biologie Médicale*. Asip Santé, v1.3.0.0 edition.

Birgit Demuth, C. W. (2009). Model and object verification by using dresden ocl. In *Proceedings of the Russian-German Workshop Innovation Information Technologies: Theory and Practice*. Ufa State Aviation Technical University, Ufa, Bashkortostan, Russia.

Birgit Demuth, S. L. and Zschale, S. (September 15 - 17, 2004). Structure of the dresden ocl toolkit. In *The 2nd International Fujaba Days : MDA with UML and Rule-based Object Manipulation*. Technical University of Darmstadt, Germany.

Bradner, S. (1997). *RFC 2119 : Keywords for use in RFCs to Indicate Requirement Levels*. W3C recommendation, Harvard University, Boston, Massachusetts.

Clark, J. and DeRose, S. (November, 1999). *XML path language (XPath) version 1.0*. W3C recommendation.

Dave Steinberg, Frank Budinsky, M. P. E. M. (December 16, 2008). *EMF: Eclipse Modeling Framework*. Addison-Wesley Professional.

eHealth Suisse (May 21, 2013). *Format déchange Rapports de laboratoire soumis à déclaration en Suisse*. eHealth Suisse.

epSOS (January 03, 2013). *epSOS Architecture and Design Interoperability Specification*. Smart Open Services for European Patients.

Eric Poiseau, Karima Bourquard, M. O. M. Z. (2010). *Testing Tools Strategy*. Healthcare Interoperability Testing and Conformance Harmonisation (HITCH).

Georg Duftschmid, Wolf gang Dorda, W. G. (2009). The elga initiative: A plan for implementing nationwide electronic health records system in austria. In *Medical University of Vienna*.

Hans-Erik Erikson, Magnus Penker, B. L. D. F. (2004). *UML2 Toolkit*. Wiley Publishing, Inc.

Heymans S, McKennirey M, P. J. (2011). Semantic validation of the use of snomed ct in hl7 clinical documents. In *Journal of Biomedical Semantics*.

HL7 (June, 2005b). *HL7 Version 3 Standard: Common Terminology Services*. Health Level Seven, release 2 edition.

HL7 (May, 2005a). *HL7 Clinical Document Architecture, Release 2.0 (CDA), Normative Edition*. Health Level Seven.

Hongli Lin, Zhencheng Chen, W. W. (2010). Xml schemas representation of dicom data model. In *Bioinformatics and Biomedical Engineering (iCBBE)*.

IHE (August 10, 2010). *IT Infrastructure Technical Framework Supplement, Sharing Value Sets*. Integrating the Healthcare Enterprise.

IHE (August 31, 2012a). *IT Infrastructure Technical Framework*. Integrating the Healthcare Enterprise.

IHE (August 31, 2012b). *IT Infrastructure Technical Framework Supplement, Cross-Enterprise Document Workflow (XDW)*. Integrating the Healthcare Enterprise.

Lee, D. . C. (September, 2000). Comparative analysis of six xml schema languages. In *ACM SIGMOD*.

McLaughlin, B. (2002). *Java And Xml Data Binding*. O'Reilly.

McLaughlin, B. (August, 2001). *Java and XML*. O'Reilly, second edition.

Melissa Markey, M. M. (2012). Practice manager panel: Putting ehrs to meaningful use in the medical office, perspectives from practice managers. HIMSS.

OASIS (November 2011). *Test Assertions Markup Language (TAML)*. Advancing Open Standards For the Information Society (OASIS), v1.0 edition.

OMG (January, 2008). *MOF Model to Text Transformation Language*. Object Management Group, v1.0 edition.

OMG (January, 2012). *OMG Object Constraint Language specification (OCL)*. Object Management Group, v2.3.1 edition.

R. Bhuvaneswari, K. K. (February 2012). Software support for xml schema design patterns and pattern matching of xml schemas. In *International Journal of Scientific and Research Publications, Volume 2, Issue 2*.

Sondra Renly, Rita Altamore, L. N. (November, 2012). A new model for collaboration: Building cda documents in mdht. In *AMIA Annual Symposium Proceeding*.

Thorp, J. (September, 2010). epsos project, an overview of european interoperability initiatives. In *AHIMA*.