Implementing TMCL: XTche – a topic map schema and constraint specification language

Giovani Rubert Librelotto
UNIFRA – Centro Universitário Franciscano
Rua dos Andradas, 1614, Santa Maria, RS – Brazil
giovani@unifra.br

and

Renato Preigschadt de Azevedo
UNIFRA – Centro Universitário Franciscano
Rua dos Andradas, 1614, Santa Maria, RS – Brazil
rpa.renato@gmail.com

and

Rogério Corrêa Turchetti
UNIFRA – Centro Universitário Franciscano
Rua dos Andradas, 1614, Santa Maria, RS – Brazil
turchetti@unifra.br

and

José Carlos Ramalho
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
jcr@di.uminho.pt

and

Pedro Rangel Henriques
Universidade do Minho, Departamento de Informática
Braga, Portugal, 4710-057
prh@di.uminho.pt

Abstract

In this paper we present a Topic Maps Validation System – XTche constraint language and its processor. We started with our strong motivation to check a topic map for syntactic and semantic correctness – as a notation to describe an ontology that supports a sophisticated computer system where its validation is crucial!
Then we assume XTM and TMCL as starting points and we used our background in compilers and XML validation to come up with our proposal. XTche complies with all requirements stated for TMCL but it is an XML Schema oriented language. This idea brings two benefits: on one hand it allows for the syntactic specification of Topic Maps (not only the constraints), eliminating the need for two separated specifications (schema and constraints); and on the other hand it enables the use of an XML Schema editor (like XMLSpy) to provide a graphical interface and the basic syntactic checker.
With XTche, a topic map designer defines a set of restrictions that guarantee that a particular topic map is semantically valid.

Keywords: Topic Maps, Ontology, Semantic Validation, TMCL, XML.

1 Introduction

According to Topic Map Data Model [9], Topic Maps *are abstract structures that can encode knowledge and connect this encoded knowledge to relevant information resources*. Topic Maps allow a domain knowledge representation in semantic networks, composed of topics and associations.
Nowadays, almost all topic maps are built by hand. This kind of edition is time consuming and has important financial costs. There are several tools for topic map edition but they have some limitations like the lack of a topic map semantic validator.
In order to cope with a broad range of scenarios, a topic is a very wide concept. On one hand, this makes Topic Maps a convenient model for knowledge representation; but on the other hand, this can also put in risk the topic map consistency. A set of semantic constraints must be imposed to the topic map in order to grant its consistency.
The Topic Maps standard does not provide language constructors to specify the semantics. So it is not possible to derive from the standard mechanisms to validate a topic maps against the contextual rules. Therefore it is necessary to improve the ISO 13250 standard adding a support for constraints definition enabling the creation of a processor for topic map automatic validation.
The main contribution of this paper is a constraint language for topic maps called XTche and its processor. XTche language [11] is TMCL-based (*Topic Map Constraint Language*) [15]. This language allows to complement the description of the semantic network structure (composed of topic and associations) with schema, contextual, and existence constraints, thus defining the semantics of topic maps that should be preserved.
In order to present this automatic system to validate topic maps, Section 2 presents the state-of-art about Semantic Web, Ontology, and Topic Maps. Section 3 introduces TMCL (*Topic Map Constraint Language*). Section 4 presents XTche language and its processing is described in section 5. Section 6 shows the validation process of each XTche specification. A comparison between topic maps schema languages – namely XTche, OSL, and AsTMa! – is found in section 7. The conclusion is found in section 8.

2 Semantic Web, Ontology, and Topic Maps

Semantic Web is concerned with the arrangement on the Web of information in such way that its meaning can be understood by computers as easily as by people; that is, the web pages contain not only the concrete information to be shown, but also metadata that allows for its semantic interpretation. Such an organization of information offers new perspectives for the Web [13]:

- Greater efficiency and precision in the search for and comprehension of information by users, humans or machines;
- Automatic treatment of information,
- Transfer of simple tasks like search, selection, updating, and transaction from the user to the system.

Organization, *standardization* and *automatic treatment of information* are the key elements that allowed the transition from the *first Web generation*, which is first of all a vast collection of anarchic information, to the *Semantic Web*, which aims at treating decentralized, sharable, and exploitable knowledge.
The Semantic Web requires the cooperation of various disciplines: Ontologies, Artificial Intelligence, Agents, Formal Logic, Languages, Graph Theory and Topology, etc. Our working area is Ontologies for the Web, more exactly, ontologies represented by Topic Maps to be handled by web applications and browsers.
An ontology is a way of describing a shared common understanding, about the kind of objects and relationships which are being talked about, so that communication can happen between people and application systems [19]. In other words, it is the terminology of a domain (it defines the universe of discourse). As a real example consider the thesaurus used to search in a set of similar, but independent, websites.
Ontologies can be used to:

- Create a structured core vocabulary, to be used and validated by a set of actors in a community;
- Define and use logical relationships and rules between the concepts, allowing an efficient use of intelligent agents;
- Develop, maintain, and publish knowledge (that changes rapidly) about an organization (the whole or a part), easily providing different views.

Topic Maps [17] are a good solution to organize concepts, and the relationships between those concepts, because they follow a standard notation – ISO/IEC 13250 [2] – for interchangeable knowledge representation. Topic Maps are

composed of topics and associations giving rise to structured semantic network that gathers information concerned with certain domain. This hierarchical topic network can represent an ontology.
A topic map is an organized set of topics (formal representation of subjects), with:

- several names for each topic (or subject of the index);
- pointers (occurrences) between topics and external documents (information resources) that are indexed;
- semantic relationships, whether they are hierarchical or not, between topics via associations;

It also has the capability of supporting multi-classification (a topic can belong to more than one class), and offers a filtering mechanism based on the concept of *scope* that is associated with names, occurrences, and associations.
According to [19], Topic Maps are very well suited to represent ontologies. Ontologies play a key role in many real-world knowledge representation applications, and namely the development of Semantic Web. The ability of Topic Maps to link resources anywhere, and to organize these resources according to a single ontology, will make Topic Maps a key component of the new generation of Web-aware knowledge management solutions.
On one hand, this section helps to understand our interest on Topic Maps in the actually important area of Semantic Web; on the other hand, the concepts so far introduced pointed out the indubitable need for mechanisms to guarantee the semantic correctness of Topic Maps.

3 TMCL – Topic Maps Constraint Language

The language to define topic map constraints is called as *Topic Map Constraint Language* (TMCL). This language is currently on its way for standardization (ISO 19756 [14]). The objective of TMCL is to allow formal specification of rules for topic map documents [15]. TMCL has a similar purpose as schema languages for relational databases or XML applications. The constraint language is required to formalize application specific rules. Currently there are many different proposed constraint languages.
The requirements for TMCL include capacity to express constraints over the ontology of a topic map. An ontology in topic maps can be described through classes of topics, association types, role types, and occurrence types. Besides, the relation between association types and specific role types, occurrence types and topic types, role types in association types and topic classes, implicit rules and cardinality, etc, are important to the definition of a ontology for a topic map and they are covered by TMCL requirements.

4 XTche Language

The objective of XTche language is to formally specify constraints for topic maps. Adopting XTM format, the syntactic validation of a topic map is assured by any XML parser because XTM structure is defined by a DTD. However, it is well known that structural validity does not mean the complete correctness – semantics should also be guaranteed.
Using XML Schema instead of DTD improves the validation process because some semantic requirements (domain, occurrence number, etc.) can be added to the structural specification. Still XML parsers will deal with that task.
However other semantic requirements remain unspecified. So, a specification language that allows us to define the schema and constraints of a family of Topic Maps is necessary.

4.1 An XML Schema-based language

Like XTM , XTche specifications can be too verbose; that way it is necessary to define constraints in a graphical way with the support of a visual tool. To overcome this problem, XTche syntax follows the XML Schema syntax; so, any XTche constraint specification can be written in a diagrammatic style with a common XML Schema editor.
It is up to the designer to decide how to edit the constraints and schemas: either in a XML Schema visual editor (that outputs the respective textual description), or in an XML text file according to XTche schema. The XTche specification (in textual format) is taken as input by *XTche Processor* that analyzes and checks it, and generates a Topic Maps validator (*TM-Validator*) as output (more details in the Section 5).
XTche is an XML Schema-based language. All XTche specifications are XML Schema instances; but, obviously, not all XML Schema instances are XTche specifications.
Section 4.1.1 describes the skeleton for all the XTche specifications. That skeleton is a generic, but incomplete, XML Schema that must be fulfilled with particular constraints for each case, as detailed in Section 4.2.1 and Section 4.3. To write those constraints, the basic schema language is extended by a set of domain specific attributes that are defined in a separated file (also presented in Section 4.1.1) imported by the skeleton.
Like any other schema, before processing an XTche specification (in order to generate a *TM-Validator*), its correctness should be checked. An usual XML Schema-based parser is not enough to do that desired validation; we had to extend

it with one more layer (to take care of the above referred domain specific attributes) as will be explained in subsection 6.
Those two XML Schemas (the first one incomplete) are all that is necessary to learn the general structure of the new XTche language to define the schema of Topic Maps. To use it, the topic map designer shall also know how to write the constraints he wants to be satisfied by each particular topic map instance. However, before explaining both the schema and contextual constraints, let us just talk about the XTche validation that will guarantee that a particular specification is a well-formed XML-Schema and a valid XTche description.

4.1.1 XTche Skeleton

An XTche specification has a schema where the `<xtche>` element is the root. This element is composed of a sequence, where two elements are allowed: `<schema-constraints>` – that specifies the schema constraints – and `<contextual-constraints>` – that specifies the contextual constraints. Both subelements are optional; it means a specification can only have one kind of constraints. These subelements are composed of a sequence, where each subelement represents a particular constraint.

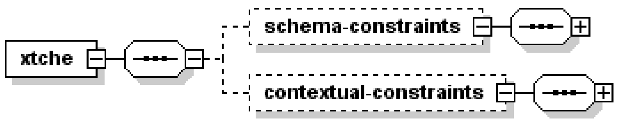

Figure 1: XTche specification inicial structure

The diagram of Figure 1 represents the code presented below, the generic skeleton above referred (that must be completed in each case). It begins with root specification, where the namespace `xtche` must be declared with the value `http://www.di.uminho.pt/~gepl/xtche`.
After that, it is necessary to import the schema that specifies the XTche attributes, as discussed above. This schema is available in `http://www.di.uminho.pt/~gepl/xtche/xtche-schema.xsd`. Finally, a sequence of two non-required elements (contextual-constraints and schema) allows the definition of all the constraints necessary to validate the particular topic maps under definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xtche="http://www.di.uminho.pt/~gepl/xtche" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import namespace="http://www.di.uminho.pt/~gepl/xtche"
    schemaLocation="http://www.di.uminho.pt/~gepl/xtche/xtche-schema.xsd"/>
    <xs:element name="xtche">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="schema-constraints" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <!-- schema constraint 1 -->
                            <!-- schema constraint 2 -->
                            ...
                            <!-- schema constraint N -->
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="contextual-constraints" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <!-- contextual constraint 1 -->
                            <!-- contextual constraint 2 -->
                            ...
                            <!-- contextual constraint N -->
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

The specific XML Schema for XTche attributes (imported by the skeleton above) is found in `http://www.di.uminho.pt/~gepl/xtche/xtche-schema.xsd`. That schema defines all the attributes required to qualify the elements in an XTche specification.

4.2 Schema and Contextual Constraints

A list of requirements for the new language was recently established by the ISO Working Group – the *ISO JTC1 SC34 Project for a Topic Map Constraint Language* (TMCL) [15]. XTche language meets almost all the requirements in

that list; for that purpose, XTche has a set of constructors to describe constraints in Topic Maps, as will be detailed in the next subsections. But the novelty of the proposal is that the language also permits the definition of the topic map structure in an XML Schema style. An XTche specification merges the schema (defining the structure and the basic semantics) with constraints (describing the contextual semantics) for all the topic maps in that family.
These constraints can be divided in two parts: *schema constraints* and *contextual constraints*. The first subset defines the Topic Maps Schema (i.e., the structure of topics, associations, and occurrences); the second one is applied over particular conditions in a topic map.

4.2.1 Schema constraint specification

The schema constraint specification follows closely XTM schema [18]. Each schema specification is a subelement of `<schema-constraints>`, the first subelement of `<xtche>`, as shown in the skeleton previously presented. It has several elements structured according XTM schema.
For instance: to specify that topics of type `customer` must have one occurrence of type `address` and other occurrence of type `email`, we should write the code below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xtche="http://www.di.uminho.pt/~gepl/xtche" xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace="http://www.di.uminho.pt/~gepl/xtche"
    schemaLocation="http://www.di.uminho.pt/~gepl/xtche/xtche-schema.xsd"/>
    <xs:element name="xtche">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="schema-constraints">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="customer">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="address">
                                            <xs:complexType>
                                                <xs:attribute ref="xtche:occurrenceType"/>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="email">
                                            <xs:complexType>
                                                <xs:attribute ref="xtche:occurrenceType"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute ref="xtche:topicType"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Figure 2 is the respective diagrammatic view.

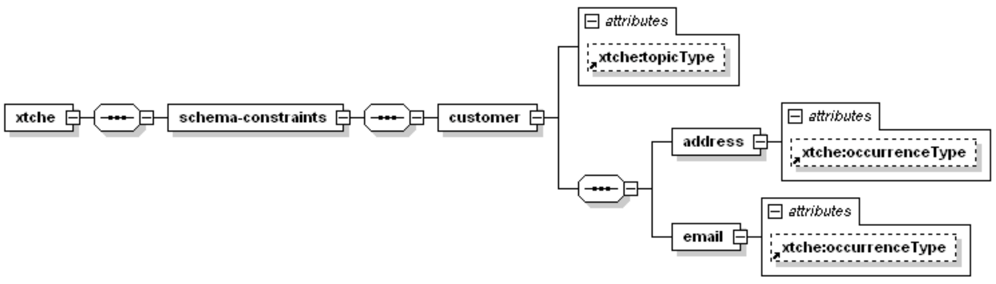

Figure 2: An XTche specification

To compare the XTche specification in Figure 2 and XTM structure, Figure 3 exhibits a part of its schema, where the path to occurrence scope is in contrast.
As shown in Figure 2 one schema constraint is a sequence of concrete topics (`customer`, `address`, and `email`) each one qualified by an associated XTche attribute. A similar description in XTM (Figure 3) uses generic element names (`topic`, `occurrence`, and `instanceOf`) and defines the concrete data via attributes associated to those elements (see code below). This systematic correspondence justifies a previous statement that the XTM code can be inferred from the XTche specification. However, the first one contains more semantic information.

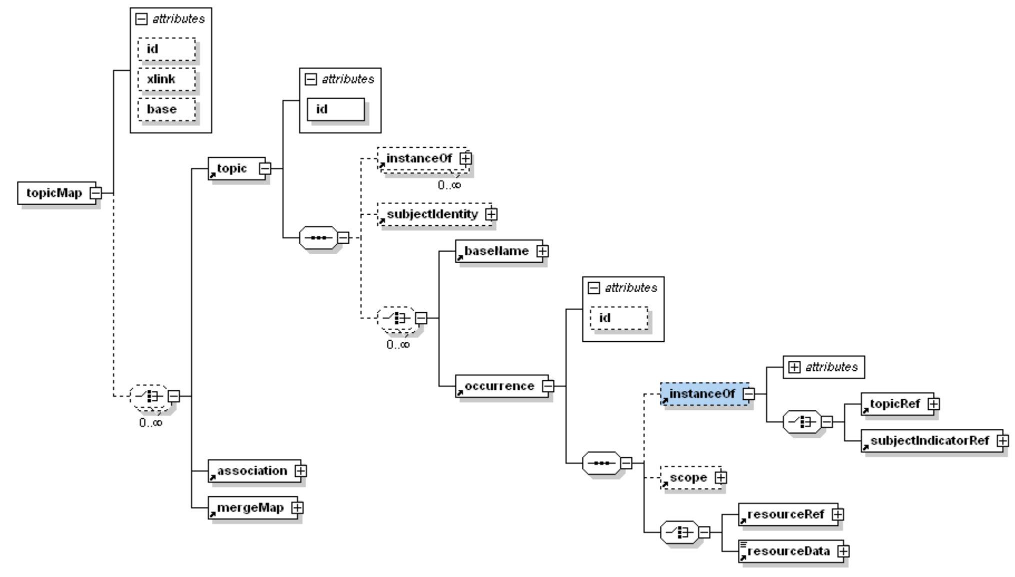

Figure 3: Part of XTM schema

```
    <topic id="xxx">
        <instanceOf>
            <topicRef xlink:href="#customer"/>
        </instanceOf>
        <occurrence>
            <instanceOf>
                <topicRef xlink:href="#address"/>
            </instanceOf>
        </occurrence>
        <occurrence>
            <instanceOf>
                <topicRef xlink:href="#email"/>
            </instanceOf>
        </occurrence>
    </topic>
```

A more sophisticated XTche example inspired in the *E-Commerce Application*, subsection 6.1 of [15], is described in [12].

4.3 Contextual constraint specification

Contextual constraints appear in the XTche specification as subelements of `<contextual-constraints>`, the second subelement of `<xtche>`, as explained in subsection 4.1.1 (see the skeleton shown). They do not have more subelements; they only have attributes.
For instance, to create a topic `map` and say that *it can be used for typing occurrences and nothing else*, we have to add a `<map>` subelement with an `@occurrenceType-Exclusive` attribute, as shown in Figure 4.

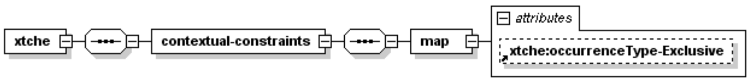

Figure 4: A contextual constraint specification example

The XTche textual code generated by edition of Figure 4 is described below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xtche="http://www.di.uminho.pt/˜gepl/xtche" xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace="http://www.di.uminho.pt/˜gepl/xtche"
    schemaLocation="http://www.di.uminho.pt/˜gepl/xtche/xtche-schema.xsd"/>
    <xs:element name="xtche">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="contextual-constraints">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="map">
                                <xs:complexType>
                                    <xs:attribute ref="xtche:occurrenceType-Exclusive"/>
```

```
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
</xs:schema>
```

Such restriction can not be made explicitly in XTM; this is why we call that family of constraints *contextual*, to distinguish from those that can be included in XTM (called *schema-constraints*). This way, to validate the above stated restriction, the *TM-Validator* needs to check if the topic profile is only found as a `topicRef` element at the end of `//occurrence/scope` path, as shown in Figure 3.

5 XTche Language Processing

Each sentence in XTche language – listing all the conditions (involving topics and associations) that must be checked – specifies a *specific topic map validation process* (*TM-Validator*), enabling the systematic codification (in XSL) of this verification task. We understood that those circumstances, it was possible to generate automatically this *TM-Validator*. For that purpose, we developed another XSL stylesheet that translates an XTche specification into the *TM-Validator* XSL code.
The *XTche processor* is the *TM-Validator* generator; it behaves precisely like a compiler generator and it is the core of our architecture, as can be seen in Figure 5. It takes a valid *topic map schema and constraint specification* (an XML instance, written according to the XTche language) and generates an XSL stylesheet (the *TM-Validator*) that will process an input topic map to generate a valid topic map or error messages. Actually if no errors are detected, the output is the same topic map given as input, but now it is sure that it complies with all the constraints.

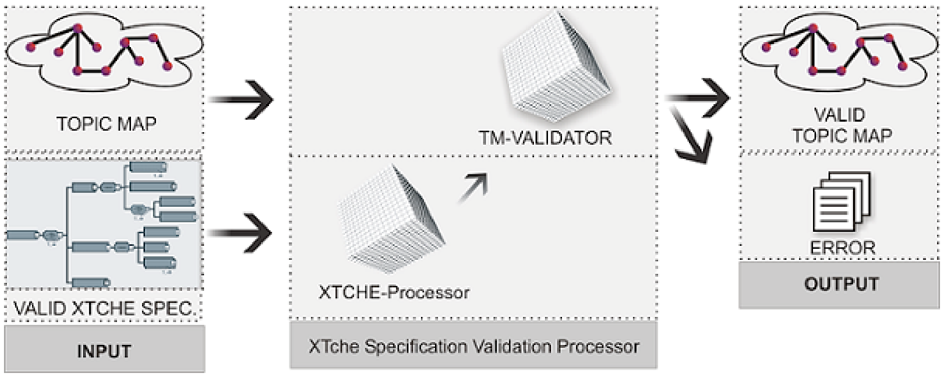

Figure 5: XTche Architecture

Both XSL stylesheets (the generator and the validator) are interpreted by a standard XSL processor like Saxon[1], what in our opinion is one of the benefits of the proposal.

6 XTche-Specification Validation Processor

XTche-Specification Validation Processor (*XTche-SpecVP*) checks the structure of a XTche-specification in agreement with the standard schema for XML Schema language and the specific schema for XTche language, presented in last subsection.
Figure 6 depicts that processor, which behavior is: initially, it verifies if the source XTche specification is a valid XML Schema (any XML parser is able to do this simple task); if no errors are found, the processor executes the second step that consists on the verification of its compliance against the rules defined below. Errors are reported as they occur. The XTche specification is correct if no errors are reported.

7 Related Work

AsTMa! [1] (another Topic Maps constraint language) proposes a mechanism to validate a topic map document against a given set of rules. This language uses *AsTMa=*, the authoring language, and extends it with several new language

[1]http://saxon.sourceforge.net/

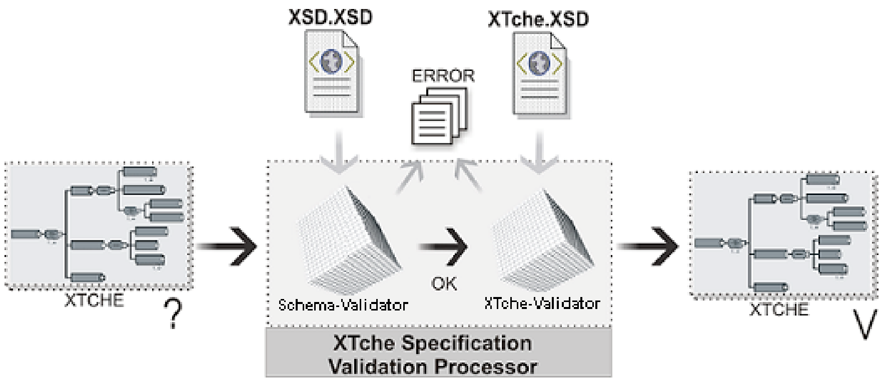

Figure 6: XTche-Specification Validation Processor

constructs, and logic operators like NOT, AND and OR, simple logical quantifiers and regular expressions. *AsTMa!* exposes some features of a future TMCL.
OSL (*Ontopia Schema Language*) [8] allows the definition of a topic map schema, i.e., a set of topic and association class definitions. These class definitions constrain the structure of the instances of the classes, and so control the form information may take in a topic map that uses the schema.
In another work related, Eric Freese [6] shown that it should be possible to use the DAML+OIL [4] language to provide a validation mechanism for topic map information. This paper present how to describe validation and consistency of the information contained in Topic Maps using DAML+OIL, showing examples about PSI (*Published Subject Identifiers*) [16], class hierarchies, and assigning properties to topics.
When a comparison between XTche and the related works is done, some advantages is detected: XTche has a XML Schema-based language, a well-known format. In addition, XTche allows the use of an XML Schema graphical editor, like XMLSpy[2]. With the diagrammatic view, it is easy to check visually the correctness of the specification.
Talking about the constraints covered by these languages, XTche and AsTMa! has more mechanisms to check the validity of Topic Maps than OSL and the Eric Freese proposal.

8 Conclusion

Topic Maps tend to grow quite fast. Most times the designer has some restrictions in mind like: what kind of topics should be used for abstract concepts, what topics may link to resources and what topics can not, ... All these restrictions tend to blur when things get big or if the some member of the team changes. In these situations an automatic system able to validate the restrictions is desirable.
The constraining process presented in this paper is composed of a language and a processor. The language is based on XML Schema syntax (we have used the same syntax and concepts in a similar approach to RDFS [3] [7]). The processor is developed in XSLT language. *XTche processor* is very similar to the Schematron [5] or XCSL [10] processors: it's an high-level stylesheet that takes a XTche specification as input and generates a specific XSLT stylesheet. This stylesheet when applied to the Topic Map validates the constraints in the XTche specification.
It means that: on one hand, we were able to describe the constraints required by each problem in a direct, clear and simple way; on the other hand, the Topic Maps semantic validator could process every document successfully, that is keeping silent when the constraints are satisfied, and detecting/reporting errors, whenever the contextual conditions are broken.
XTche language is part of Metamorphosis [11] project – a environment that can extract data from information resources and build a topic map according to a specification, validate it, and generate a conceptual navigation over the topic map knowledge. Metamorphosis – a Topic Maps oriented environment – generates conceptual navigators for heterogenous information resources providing the desired interoperability.

References

[1] Robert Barta. AsTMa! Bond University, TR., 2003. `http://astma.it.bond.edu.au/constraining.xsp`.

[2] Michel Biezunsky, Martin Bryan, and Steve Newcomb. ISO/IEC 13250 - Topic Maps. ISO/IEC JTC 1/SC34, December 1999. `http://www.y12.doe.gov/sgml/sc34/document/0129.pdf`.

[2]Available in http://www.altova.com

[3] Dan Brickley and Ramanathan V. Guha. Resource Description Framework (RDF) Schema specification 1.0. World Wide Web Consortium, March 2000. `http://www.w3.org/TR/2000/CR-rdf-schema-20000327/`.

[4] DARPA. Reference description of the DAML+OIL ontology markup language., March 2001. `http://www.daml.org/2001/03/reference/`.

[5] Leigh Dodds. Schematron: Validating XML Using XSLT. In *XSLT UK Conference*. Keble College, Oxford, England, 2001.

[6] Eric Freese. Using DAML+OIL as a Constraint Language for Topic Maps. In *XML Conference and Exposition 2002*. IDEAlliance, 2002. `http://www.idealliance.org/papers/xml02/dx_xml02/papers/05-03-03/05-03-03.html`.

[7] Lars Marius Garshol. An RDF schema for topic maps. `http://psi.ontopia.net/rdf/`, 2002.

[8] Lars Marius Garshol. The Ontopia Schema Language – Tutorial. `http://www.ontopia.net/omnigator/docs/schema/tutorial.html`, 2004.

[9] Lars Marius Garshol and Graham Moore. Topic Maps – Data Model. In *ISO/IEC JTC 1/SC34*. `http://www.isotopicmaps.org/sam/sam-model/`, January 2005.

[10] Marta H. Jacinto, Giovani R. Librelotto, Jos C. Ramalho, and Pedro R. Henriques. Constraint Specification Languages: comparing XCSL, Schematron and XML-Schemas. In *XML Europe 2002*, Barcelona, Spain, 2002.

[11] Giovani Rubert Librelotto. *XML Topic Maps: da Sintaxe Semntica*. PhD thesis, Departamento de Informtica, Escola de Engenharia, Universidade do Minho, 2005.

[12] Giovani Rubert Librelotto, Jos Carlos Ramalho, and Pedro Rangel Henriques. XTche - A Topic Maps Schema and Constraint Language. In *XML 2004 Conference and Exposition*, Washington D.C., U.S.A, 2004. IDEAlliance.

[13] Mondeca. Questions and Answers. Mondeca, March 2004. `http://www.mondeca.com/english/faqs.htm`.

[14] Graham Moore, Mary Nishikawa, and Dmitry Bogachev. Topic Map Constraint Language. ISO/IEC JTC 1/SC 34, 2004. `http://www.jtc1sc34.org/repository/0549.htm`.

[15] Mary Nishikawa, Graham Moore, and Dmitry Bogachev. Topic Map Constraint Language (TMCL) Requirements and Use Cases. ISO/IEC JTC 1/SC34 N0548, 2004. `http://www.jtc1sc34.org/repository/0548.htm`.

[16] OASIS. OASIS Topic Maps Published Subjects TC. OASIS, 2003. `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tm-pubsubj`.

[17] Jack Park and Sam Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*, volume ISBN 0-201-74960-2. Addison Wesley, 2003.

[18] Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0 - Annex D: XTM 1.0 Document Type Declaration (Normative). TopicMaps.Org Specification, August 2001. `http://www.topicmaps.org/xtm/1.0/#dtd`.

[19] Ann Wrightson. Topic Maps and Knowledge Representation. Ontopia, February 2001. `http://www.ontopia.net/topicmaps/materials/kr-tm.html`.