Constraining XML Topic Maps with XTche

Giovani Rubert Librelotto[1], José Carlos Ramalho[1], and
Pedro Rangel Henriques[1]

Universidade do Minho, Departamento de Informática
4710-057, Braga, Portugal
{grl, jcr, prh}@di.uminho.pt

Abstract. This paper presents a process for specifying constraints on topic maps with a constraint language. This language allows to express contextual conditions on classes of Topic Maps. With XTche, a topic map designer defines a set of restrictions that enables to verify if a particular topic map is semantically valid. As the manual checking of large topic maps (frequent in real cases) is impossible, it is mandatory to provide an automatic validator.

The constraining process presented in this paper is composed by a language and a processor. The language is based on XML Schema syntax. The processor is developed in XSLT language. The XTche processor takes a XTche specification and it generates a particular XSLT stylesheet. This stylesheet can validate a specific topic map (or a set of them) according to the constraints in the XTche specification.

In this paper we will show, in abstract terms and with concrete examples, how to specify Topic Maps schemas and constraints with XTche.

1 Introduction

Topic Maps are a standard for organizing and representing knowledge about a specific domain. They allow us to specify subjects and relationships between subjects. Steve Pepper [9] defines subject as the term used for the real world thing that the topic itself stands in for. A topic, in its most generic sense, can be anything whatsoever - a person/object, an entity/organization, a concept - regardless of whether it actually exists or is a mental abstraction [12].

Besides the simplicity and powerfulness of the topic/association-based model, there are two Topic Maps features that are important in the process of understanding and reasoning about a domain: the hierarchical structure that is represented in a map (defined by the relations is-a or contains); and the complementary topic network (made up of other links that connect topics that are not included in each other but just interact).

The facts above explain the importance of Topic Maps to describe knowledge in general; in particular their application to define ontologies is one of the growing up fields. So Topic Maps are nowadays widely used within XML environments: in archives, for cataloging purposes; or in web browsers, for conceptual navigation.

To build reliable systems, like those referred, it is crucial to be sure about the validation of the underlying semantic network.

Like in other fields, as formal language and document processing, it is wise to validate the syntax and semantics of a topic map before its use. This is precisely the focus of this paper: we propose XTche, a language to define Topic Maps Schema and Constraints. The validation process of a topic map based on a XTche specification will also be under the scope of the paper.

In section 2 there is an overview about the basic concepts in the area of this work: Semantic Web, Ontology, and Topic Maps; it creates the context and motivation for our concern with the precise semantics of Topic Maps. Section 3 describes XTche; before the introduction of XTche specific semantic constructors, we distinguish schema and contextual constraints. Then the automatic analysis of a XTche specification (in order to generate a concrete validator) is discussed. Section 4 compares our proposal with related work and exemplifies the use of our constraint language. A synthesis of the paper and hints on future work are presented in the last part, in the section 5.

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 [6]:

– 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 [13]. 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 [8] are a good solution to organize concepts, and the relationships between those concepts, because they follow a standard notation – ISO/IEC 13250 [3] – 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 [13], 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 XTche – A Language for Topic Maps Schema and Constraints

This section presents a language to define constraints on Topic Maps, called XTche. This language allows the topic map semantic validation. Before describing the language and its processor (a validator generator), we give the motivation behind its development, and discuss what a constraint is in this context.

As shown in section 2, when developing real topic maps, it is highly convenient to use a system to validate them; this is, to verify the correctness of an actual instance against the formal specification of the respective family of topic maps (according to the intention of its creator).

Adopting XTM format [10], the syntactic validation of a topic map is assured by any XML parser because XTM structure is defined by a DTD [11]. 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.

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) [7]. XTche language meets 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; it is no more necessary a separate syntactic description. A 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.

A Topic Map Schema defines all topic types, scopes, subject indicators, occurrence types, association types, association roles, and association players. So, it is possible to infer a topic map skeleton (written in XTM) from the schema; the user or an application (like Oveia [LSRH04] , a Metamorphosis [LRH03] module) must only fill it in (with data extracted from the information resources) to obtain the topic map instances. This functionality (skeleton derivation and syntactic validation) will not be more developed in this paper, as it is devoted to the semantic aspects.

3.1 Schema Constraints and Contextual Constraints

XTche is designed to allow users to constrain any aspect of a topic map; for instance: topic names and scopes, association members, topics allowed as topic type, roles and players allowed in an association, instances of a topic (enumeration), association in which topics must participate, occurrences cardinality, etc.

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.

An extensive list of Topic Maps constraints, classified as *schema* or *contextual constraints*, is presented in [5]. Although all the concerns in this list are constraints, there is actually a slight difference in the way of dealing with the two subsets. So, the wish to have XTche expressing both – contextual constraints and schema constraints – has a direct influence in the design of the language and its processing. We will care about that in the following subsections.

3.2 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 3.5).

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 3.2 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 3.3 and Section 3.4. 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 3.2) 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. However, 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 Section 3.2.

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.

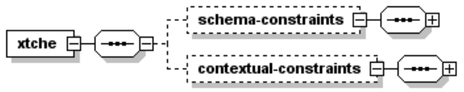

Fig. 1. XTche specification inicial structure

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 in the introduction to this section. 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.

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.

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.

About this subject, it is possible to make an analogy between an XTche specification and an XML document: an XTche instance should be a well-formed[1] XML Schema but it also needs to be valid according to XTche schema. So, its correctness is assured by *XTche-SpecVP* that performs separately those two verifications.

Figure 2 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.

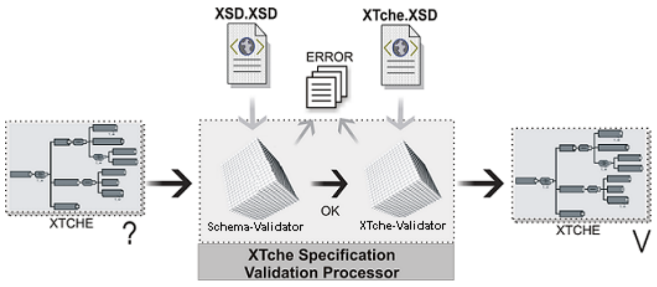

Fig. 2. XTche-Specification Validation Processor

Rules verified by *XTche-SpecVP* in the second phase are described in [5].

3.3 Schema constraint specification

The schema constraint specification follows closely XTM schema. 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 `country` must have occurrence of type `map` in the scope `geography`, we should write the code below:

```
<xs:element name="country">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="map">
            <xs:complexType>
               <xs:sequence>
```

[1] The concept of being well-formed was introduced as a requirement of XML, to deal with the situation where a schema (DTD, XML Schema, or RelaxNG) is not available.

```
                  <xs:element name="geography">
                     <xs:complexType>
                        <xs:attribute ref="xtche:occurrenceScope"/>
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>
               <xs:attribute ref="xtche:occurrenceType"/>
            </xs:complexType>
         </xs:element>
       </xs:sequence>
       <xs:attribute ref="xtche:topicType"/>
    </xs:complexType>
 </xs:element>
```

Figure 3 is the respective diagrammatic view.

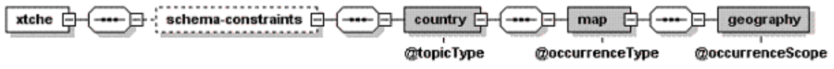

Fig. 3. An XTche specification

To compare the XTche specification in Figure 3 and XTM structure, Figure 4 exhibits a part of that schema, where the path to occurrence scope is in contrast.

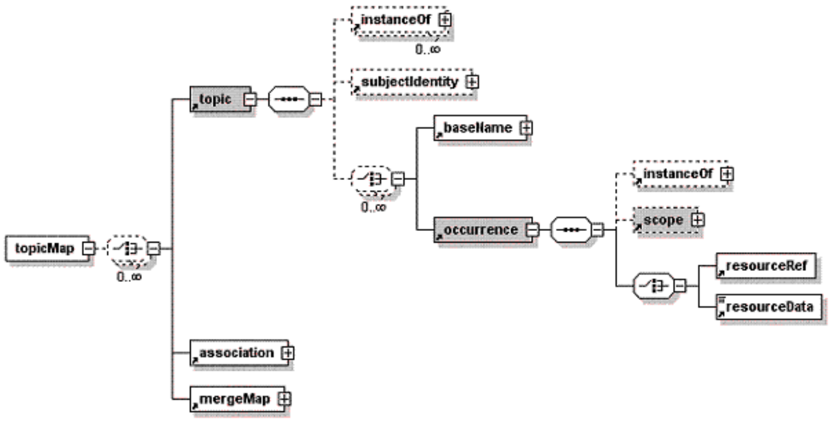

Fig. 4. XTM schema

As shown in Figure 3 one schema constraint is a sequence of concrete topics (`country`, `map`, and `geography`) each one qualified by an associated XTche attribute. A similar description in XTM (Figure 4) uses generic element names (`topic`, `occurrence`, and `scope`) 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 contains more semantic information.

```
<topic id="xxx">
    <instanceOf>
        <topicRef xlink:href="#country"/>
    </instanceOf>
    <occurrence>
        <instanceOf>
            <topicRef xlink:href="#map"/>
        </instanceOf>
        <scope>
            <topicRef xlink:href="#geography"/>
        </scope>
    </occurrence>
</topic>
```

A more sophisticated XTche example inspired in the *E-Commerce Application*, subsection 6.1 of [7], is described in [5].

3.4 Contextual constraint specification

Contextual constraints appear in the XTche specification as subelements of <`contextual-constraints`>, the second subelement of <`xtche`>, as explained in subsection 3.2 (see the skeleton shown). They do not have more subelements; they only have attributes.

For instance, to create a topic `person` and say that *it can be used for scoping occurrences and nothing else*, we have to add a <`person`> subelement with an `@occurrenceScope-Exclusive` attribute, as shown in Figure 5.

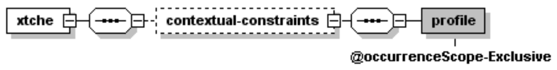

Fig. 5. A contextual constraint specification example

Such a 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 used as a `topicRef` element at the end of //occurrence/scope path, as shown in Figure 6.

3.5 XTche Processor and TM-Validator

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 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.

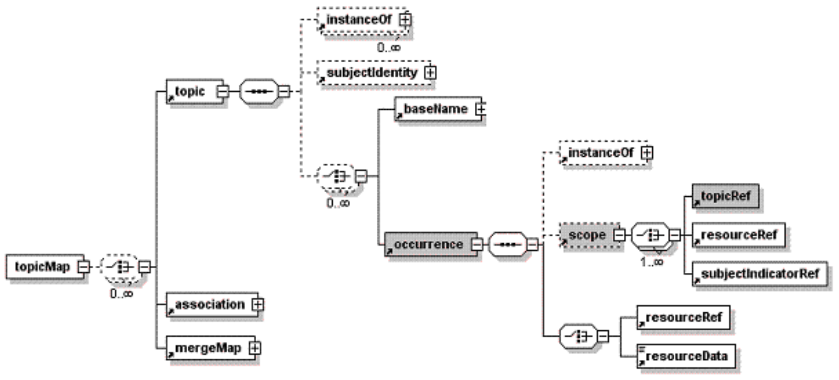

Fig. 6. XTM schema

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 7. It takes a valid topic map schema and constraint specification (an XML instance, written according to the XTche schema), verified by the XTche-SpecVP introduced in Section 3.2, and generates an XSL stylesheet (the TM-Validator) that will process an input topic map and will generate an ok/error messages (an ok message states that the topic map is valid according to the XTche specification).

Both XSL stylesheets (the generator and the validator) are interpreted by a standard XSL processor like Saxon[2], what in our opinion is one of the benefits of the proposal.

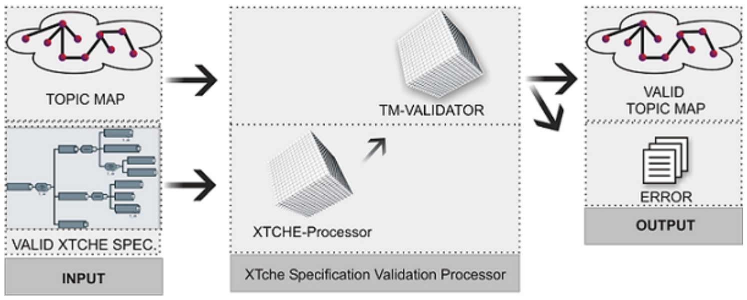

Fig. 7. XTche Architecture

During the development of this generator we found some problems that had a strong impact in the final algorithm. The most important was the ambiguity in constraint selection; until now, we have just said that an XTche specification

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

is composed of a set of constraints; we did not say that these constraints are disjoint in terms of context; in some cases there is a certain overlap between the contexts of different conditions; this overlap will cause an error when transposed to XSL; XSL processors can only match one context at a time. The solution we have adopted was to run each constraint in a different mode (in XSL each mode corresponds to a different traversal of the document tree).

4 Related Work

AsTMa! [1] is another Topic Maps constraint language, and Robert Barta also proposes a mechanism to validate a topic map document against a given set of rules. This language uses *AsTMa=* [2], the authoring language, and extends it with several new language constructs, and logic operators like NOT, AND and OR, simple logical quantifiers and regular expressions. *AsTMa!* exposes some features of a future TMCL.

In another related work, Eric Freese [4] says that it should be possible to use the DAML+OIL language to provide a constraint and validation mechanism for topic map information. The cited paper discusses how to describe validation and consistency of the information contained in Topic Maps using DAML+OIL and RDF, showing how to extend XTM and how to define PSIs and class hierarchies, as well as assign properties to topics.

Comparing XTche with the other known approaches, some advantages of XTche emerge: XTche has a XML Schema-based language, a well-known format. In addition, XTche allows the use of an XML Schema graphic editor, like XMLSpy. With the diagrammatic view, it is easy to check visually the correctness of the specification. Moreover, XTche gathers in one specification both the structure and the semantic descriptions, and it realizes a fully declarative approach requiring no procedural knowledge for users.

Talking about the constraints covered, XTche and *AsTMa!* have more mechanisms to check the Topic Maps validation than the Eric Freese's proposal.

5 Conclusion

In this paper we introduced 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 (like the applications in the area of Semantic Web or archiving) its validation is crucial!

Then we assumed 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; and on the other hand it enables the use of an XML Schema editor (for instance, XMLSpy) to provide a graphical interface and the basic syntactic checker (the first stage of the XTche-SpecVP).

We succeeded in applying this approach to some case studies - E-Commerce Application and a personal video library management system - virtually representative of all possible cases. 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.

References

1. Robert Barta. AsTMa! Bond University, TR., 2003. http://astma.it.bond.edu.au/constraining.xsp.
2. Robert Barta. AsTMa= Language Definition. Bond University, TR., 2004. http://astma.it.bond.edu.au/astma=-spec-xtm.dbk.
3. 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.
4. 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.
5. Giovani Rubert Librelotto, José 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.
6. Mondeca. Questions and Answers. Mondeca, March, 2004. http://www.mondeca.com/english/faqs.htm.
7. Mary Nishikawa, Graham Moore, and Dmitry Bogachev. Topic Map Constraint Language (TMCL) Requirements and Use Cases. ISO/IEC JTC 1/SC34 N0405rev, 2004. http://www.jtc1sc34.org/repository/0548.htm.
8. J. Park and S. Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*, volume ISBN 0-201-74960-2. Addison Wesley, 2003.
9. Steve Pepper. The TAO of Topic Maps - finding the way in the age of infoglut. Ontopia, 2000. http://www.ontopia.net/topicmaps/materials/tao.html.
10. Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification, August, 2001. http://www.topicmaps.org/xtm/1.0/.
11. 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.
12. H. Holger Rath. White Paper: The Topic Maps Handbook.
13. Ann Wrightson. Topic Maps and Knowledge Representation. Ontopia, February, 2001. http://www.ontopia.net/topicmaps/materials/kr-tm.html.