

ISO/IEC JTC 1/SC 34 N 1192

DATE: 2009-03-30

ISO/IEC JTC 1/SC 34
Document Description and Processing Languages
Secretariat: [Japan \(JISC\)](#)

DOC. TYPE	Meeting Report						
TITLE	Meeting Notes on FCD 19756, Information technology -- Topic Maps -- Constraint language (TMCL)						
SOURCE	WG 3						
PROJECT	JTC 1.34.19756						
STATUS	Notes from the SC 34/WG 3 Meeting held in Prague, Czech Republic, 2009-03-25/27. This document is circulated to the SC 34 members for information.						
ACTION ID	FYI						
DUE DATE							
DISTRIBUTION	P, O and L Members of ISO/IEC JTC 1/SC 34 ; ISO/IEC JTC 1 Secretariat; ISO/IEC ITTF						
ACCESS LEVEL	Open						
ISSUE NO.	72						
FILE	<table border="1"><tr><td>NAME</td><td>1192.pdf</td></tr><tr><td>SIZE (KB)</td><td></td></tr><tr><td>PAGES</td><td>52</td></tr></table>	NAME	1192.pdf	SIZE (KB)		PAGES	52
NAME	1192.pdf						
SIZE (KB)							
PAGES	52						

Secretariat ISO/IEC JTC 1/SC 34 - IPSJ/ITSCJ (Information Processing Society of Japan/Information Technology Standards Commission of Japan)* Room 308-3, Kikai-Shinko-Kaikan Bldg., 3-5-8, Shiba-Koen, Minato-ku, Tokyo 105-0011 Japan *Standard Organization Accredited by JISC
Telephone: +81-3-3431-2808; Facsimile: +81-3-3431-6493; E-mail: kimura@itscj.ipsj.or.jp



TMCL - Open issues

Prague, March 25-27, 2008



Very brief intro

- TMCL has no syntax of its own
 - instead, it's just a collection of PSIs
 - can be expressed in any Topic Maps syntax
- However, it's designed to be nicely expressible in CTM
 - mainly through the use of templates
- Constraints on the use of the PSIs will be expressed through a TMCL schema
- The semantics of the PSIs are expressed through TMQL queries

MAX_INT

- How to conveniently express unlimited cardinality in the CTM syntax?
- A typical constraint will look like
 - place has-occurrence(article-about, 0, MAX_INT) .
- This becomes
 - constraint-x isa tmcl:topicoccurrence-constraint
 - tmcl:card-min: 0;
 - tmcl:card-max: MAX_INT .
- Problems
 - MAX_INT is not a valid literal according to CTM
 - XML Schema integers do not have a concept of infinity
 - CTM does not support optional parameters to templates (would require “if”)

MAX_INT - solutions

- #1: Use “inf”
 - this is a valid literal for xsd:double
 - support for it has to be added to CTM
 - downside is that we cannot require max-card & min-card to be xsd:integer
 - best to allow both xsd:integer *and* xsd:double?
 - place has-occurrence(article-about, 0, inf) .
- #2: Use cardinality topics
 - allow users to define their own
 - predefine 0-1, 0-*, 1-*
 - constraint now becomes:
 - zero-any isa tmcl:cardinality-topic; tmcl:card-min: 0 .
 - place has-occurrence(article-about, zero-any) .
 - downside is
 - an extra topic type and an extra association type
 - slightly more complex TMQL queries



MAX_INT - solutions, 2

- #3: CTM template overloading
 - template x with 2 params is a different template from template x with 3
 - the templates would therefore be defined twice (with and without card-max)
 - Constraint would look like this:
 - place
 - has-occurrence(dc:description, 0, 1);
 - has-occurrence(article-about, 0) .
- #4: Define more templates with different names
 - place
 - has-occurrence(dc:description, 0, 1);
 - has-occurrence-nomax(article-about, 0) .

Solution #3 in detail

```
def has-occurrence($tt, $ot, $min, $max)
  constraint-x isa tmcl:topicoccurrence-constraint
    tmcl:card-min: $min;
    tmcl:card-max: $max .
end
def has-occurrence($tt, $ot, $min)
  constraint-x isa tmcl:topicoccurrence-constraint
    tmcl:card-min: $min .
end
# could define a third without $min as well
place
  has-occurrence(dc:description, 0, 1); # invokes first template
  has-occurrence(article-about, 0) .    # invokes second template
```

MAX_INT - solutions, 3

- **#5: Extend xsd:integer** Chosen solution!
 - add “*” as a literal for infinity in CTM
 - need to extend both lexical space and value space
 - alternatively: define ctm:integer...
 - place has-occurrence(article-about, 0, *) .
 - actually, looks like it could work (refer Graham email to sc34wg3)
 - may want to do template overloading anyway
 - TMQL has to be extended so that “*”^^ctm:integer is recognized as being bigger than any xsd:integer



Solution #5 in detail

```
def has-occurrence($tt, $ot, $min, $max)
constraint-x isa tmcl:topicoccurrence-constraint
  tmcl:card-min: $min;
  tmcl:card-max: $max .
end
place
  has-occurrence(dc:description, 0, 1);
  has-occurrence(article-about, 0, *) .
# same as has-occurrence(article-about, 0, "*"^^ctm:integer)
```



Schema topic

- Should there be a topic representing the schema?
 - Benefits
 - provides convenient point to attach metadata
 - makes it possible to track which constraints belong to which schema
 - makes it easy to check if a topic map contains a schema or not
 - Downside
 - no easy way to create topic and connecting associations using templates
 - Solutions:
 - #1: just drop it
 - #2: provide it, and redo templates
 - **#3: provide it, but do not require use of connecting associations**
 - **template overloading would help**
 - #4: make support for global wildcards (not perfect, though)
- Chosen solution!**



Schema topic: the problem

```
%include http://www.isotopicmaps.org/tmcl/schema.ctm  
place has-occurrence(dc:description, 0, 1, schema-topic) .
```

```
schema isa tmcl:schema .  
def has-occurrence($topicType, ...)  
  ?c isa tmcl:occurrence-type-constraint .  
  belongs-to-schema(schema, ?c)  
end
```

```
belongs-to-schema(schema, $c) from instance-of($c, tmcl:constraint)
```



Naming of PSIs

- topicType
- associationtype
- roletype
- occurrenceType
- nameType
- abstract-topictype-constraint
- subjectidentifier-constraint
- nametypescope-constraint
- topicoccurrence-constraint
- ...

- topic-type
- association-type
- role-type
- occurrence-type
- name-type
- abstract-topic-type-constraint
- subject-identifier-constraint
- name-type-scope-constraint
- topic-occurrence-constraint
- ...

**We choose
this!**

Store TMQL in the schema?

- In the current draft all TMCL constructs carry TMQL queries defining their meaning
 - that is, the TMQL queries from the standard are repeated in every topic map which uses TMCL
- Validation is defined in terms of running these queries
 - that is, if someone uses the standard PSIs, but changes the queries, the meaning of the constraints also changes
- User-defined constraints are defined the same way
- Solutions
 - #1: leave as is (repeat queries in every schema topic map)
 - **#2: only user-defined constraints carry TMQL**
 - **define a topic type for user-defined constraints**
 - **queries for user-defined constraints are in the topic map**
 - **queries for built-in constraint types are *not* in the topic map**
 - **built-in constraint types are defined in the spec using TMQL queries**

Chosen solution!



Validation of TMCL schemas

- The TMCL standard must say that
 - TMCL schemas must be valid (against the TMCL schema for TMCL schemas, given in the standard) before they can be used for validation
 - do we need to use loose validation here?
 - (yes, probably, maybe define in the schema itself)

tmcl:topic-type-constraint

- If this topic is included, topic types which are not instances of tmcl:topic-type cause errors
- If left out, such topic types are accepted
- Similar types are provided for associations, occurrences, etc
- Is this a good way to support a choice between strict/loose validation?
- **Instead we decide to:**
 - **define loose validation as only verifying some subset of defined constraints**
 - **define strict validation as all defined constraints**
 - **explicitly allow users a more fine-grained choice**
 - **for conformance, say validators are required to support strict validation**
 - this is because this the hardest to do (if they do this, they do everything)
 - what their default behaviour is and how to turn things on and off is left to implementors



If it's not explicitly allowed...

- ...then what?
 - Given
 - place isa topic-type;
 - has-occurrence(article-about, 0, *);
 - has-occurrence(tmdm:subject, 0, *) . # allows occurrences of any type
 - what if a place has a dc:description?
 - **#1: Strict vs loose**
 - **strict TMCL says only what's explicitly allowed through constraints is OK**
 - **therefore, under strict validation this would be illegal**
 - **however, if the user turns off that extra check, it would be OK**
- Chosen solution!**



How this would be validated (in strict mode)

- place isa topic-type;
- has-occurrence(article-about, 1, *); # query1
- has-occurrence(tmdm:subject, 0, *) . # query2
- query1: returns topics of type place which have <1 or >* article-about
- query2: returns topics of type place which have <0 or >* occurrences
- query3: returns topics of type place which have occurrences for which there are no occurrence constraints



tmcl:exclusive-instance

- Specifies that two topic types are disjoint
- Alternatives:
 - #1: leave as is
 - #2: change name to tmcl:disjoint-instances
 - **#3: change to disallow by default, and instead provide tmcl:overlapping-instances**
Chosen solution!
 - #4: provide both tmcl:disjoint-instance *and* tmcl:overlapping-instances
- Note: write formal TMQL so that this constraint can be turned off by saying that tmcl:topic-type overlaps with itself



tmcl:exclusive-instance - part 2

- How many topic types can be joined by such a constraint?
 - **the CTM template supports only 2**
 - **however, any number (greater than 1) is allowed**

Chosen solution!



Unique occurrences

- TMCL provides a unique occurrence constraint
- This, however, only makes occurrence values unique within a topic type
 - note: scope is ignored
- Usecases:
 - unique email address on person
 - unique social security number on person
 - *may* be used to implement merging rules (not specified in standard)
- Is this right?
 - observation: if you want it to be global, use tmdm:subject as the topic type
 - **therefore: this is OK, we leave it as is**

Chosen solution!



Unique occurrences - part 2

- Why can only occurrences be declared unique?
- What about names?
 - **add unique-name-constraint**
- What about associations?
 - use case: a person should only have a single father
 - use case: a city should only be contained in a single province
 - **this is already supported via role type constraints**
 - **double check this**



Symmetric association types

- TMCL has no explicit support for these (but they can be expressed)
 - friend-of(friend: a, friend: b)
 - borders-with(neighbour: country-a, neighbour: country-b)
- They can be detected by analysing constraints, but this is awkward
- Should there be a special constraint for this?
 - **NO!**
- If you use the binary template, it would not work as expected
 - so don't do that
 - however, should there be a special template? **YES!**

Documentation

- TMCL says nothing about how to document schemas; should it?
- Should we define our own PSIs? Or recommend Dublin Core?
- How much should be defined?
 - **tmcl:version -> string**
 - **tmcl:description -> string**
 - **tmcl:comment -> string**
 - **tmcl:see-also -> URL reference**
 - **the schema for TMCL schemas allows, but does not mandate these**
 - **TMCL recommends Dublin Core for the rest, but does not include it in the schema**
 - **TMCL will map these PSIs to Dublin Core where possible**
- Support for schema evolution, a la OWL?
 - **we don't define anything**



Regular expression syntax

- Which syntax should we use?
 - **Perl-style regular expressions (as defined in some suitable spec)**
 - **must make sure that these support Unicode strings**

Regular expressions

- Currently, all constraints on names, occurrences, and identities support regular expressions on the string values
- This provides additional power, but in most schemas this comes out as
- place
 - `has-name(tmdm:name-type, 1, 1, “.*”);`
 - `has-occurrence(dc:description, 1, 1, “.*”);`
 - `has-occurrence(article-about, 0, *, “.*”) .`
- Problem: we have to repeat the blank regular expression everywhere
- Solutions:
 - #1: leave as is
 - #2: template overloading
 - #3: templates don't support regular expressions
 - `has-occurrence(dc:description, 1, 1)`
 - #4: extend CTM with the concept of a default value for parameters
 - **#5: make regular expression constraints independent of topic type**
 - `has-occurrence(dc:description, 1, 1)`
 - new constraint type for this must be added to the spec

Chosen solution!



How to solve it with template overloading

```
def template($a)
  template($a, default-value)
end
def template($a, $b)
  #whatever
end
```

Regular expressions - scope

- Should these be per topic type or globally for the name/occurrence type?
 - observation: it's very closely tied to datatypes
 - use case: URIs on subject identifiers
 - use case: phone numbers
 - use case: social security numbers
 - observation: tmdm:subject does allow you to specify it globally
 - observation: proper modelling is usually globally
 - observation: modelling this globally solves the template problem
- **WE MAKE THESE CONSTRAINTS GLOBAL**

Syntax for regular expressions

- Regular expressions currently represented as strings in CTM
 - “.”
 - “\d+”
- Proposal #1: string syntax without escaping
 - /.*/
 - \d+/
- Proposal #2: string syntax without escaping
 - ‘.’
 - ‘\d+’
- Proposal #3: change interpretation of backslash in CTM
 - “illegal escape sequences” are treated literally
 - that is, \d means just \d
 - however, \n means newline
- Proposal #4-#5: as #2-#3, but with regular expression datatype
- Observation: nothing will make this pretty
 - **LEAVE AS IS** Chosen solution!

Letting a template add the datatype

- `def template($a)`
- `topic occtype: $a^^bar:baz .`
- `topic bocctype: $a^^baz:quux .`
- `template($a^^foo:bax) . # whoops`
- `end`
- `template("foo")`
- `template("foo2"^^xsd:string) # would need to account for this case`

- This is currently a syntax error in CTM, because only strings can have ^^ after them
- Should we extend CTM to allow this?
 - use case: easily adding datatype information
 - con: most datatypes that are actually used are already supposed
- **NO, NOT NOW**



Datatype for regular expressions

- The current spec has not defined one
- The question is, should we define it?
 - convenient expression of the syntactic constraints on regular expressions
- Definition would consist of
 - URI for datatype
 - definition of lexical space with reference to some regexp spec
 - whatever else is needed to satisfy XSD-2
- Usage would then require
 - “.*”^{^^}tmcl:regexp
 - which would basically force us to make a special syntax for this in CTM
 - *unless* templates are able to add datatypes
- **NO**



Scope constraints

- Current approach: at least one language in scope:
 - `displayname has-scope(language, 1, INF)`. This doesn't seem to add any benefit.
- Proposed alternative
 - `displayname`
 - `has-theme(language)`;
 - `has-theme(iso-code)`;
 - `has-scope(2, INF)`.
- Benefit of alternative
 - says 2 scoping topics, don't care how many of each type
- Can already be expressed
 - `displayname`
 - `has-scope(language, 0, *)`;
 - `has-scope(iso-code, 0, *)`;
 - `has-scope(tdm:subject, 2, *)`.

NO



Scope constraints, part 2

- Spec currently defines separately
 - nametypescope-constraint
 - occurrencetypescope-constraint
 - associationtypescope-constraint
- There is no reason to do this, so replace with
 - scope-constraint
- As in fact used on previous slide...



Scope constraints, part 3

- TMCL currently defines a topic type called scopetype
 - this is the type of all topic types whose instances can be used as scopes
 - appears to duplicate what scope constraints do
- Pro & con
 - con: not clear what the difference between scopetype and tmcl:topicitype is
 - pro: allows statement of intent to be made about topic types
 - pro: allows looser form of scope validation than has-scope constraints
 - use case: language scoping throughout map supported easily
- **WE KEEP THIS**
 - **requires us to define the TMQL for validation based on scopetype**



TMCL bug

- topicType isa topicType .
- Conflict with TMRM
 - isa is not reflexive in TMRM
- Should probably be fixed in TMRM

Need to discuss with Robert,
but probably a TMRM change



Do you have to implement TMQL?

- We need two levels of conformance
 - basic: validation based on TMCL-defined constraints only
 - full: basic plus user-defined constraints
- So the answer is
 - for basic: no
 - for full: yes
- If a basic validator gets a schema with user-defined constraints, the user-defined constraints are ignored



Formal definition of TMCL

- Basic validation:
 - A set of TMQL queries for individual constraints, defined in the spec
 - If a query returns a non-empty result set, the values in it represent violations
 - we will aim for the values to be as close to violating topic/occurrence etc as possible, but do not guarantee a complete match
 - Validation consists of running all the queries
 - if all queries return empty result sets, the topic map is valid
- Full validation
 - basic, plus running each of the user-defined queries in the same way



User-defined queries

- Sometimes requiring user-defined queries to return non-empty result sets is going to be inefficient
 - this may also be awkward in formulating queries
- We should therefore define two types of user-defined constraints:
 - one which acts like the basic constraints
 - and one which acts in the opposite way (ie: empty result set = violation)
 - consider ways to use the result set to provide user-friendly error messages

Overlap of constraints - what happens?

- When an item matches two constraints, then what?
 - place
- `has-occurrence(dc:description, 0, 1);`
- `has-occurrence(tmcl:occurrence-type, 1, *) .`
- What happens is:
 - query1: checks cardinality of dc:description occurrences
 - query2: checks cardinality of tmcl:occurrence-type (ie: all) occurrences
 - query3: checks that no occurrences exist which are not explicitly allowed@
 - must handle subtyping



Interaction with subtyping

- Does subtyping actually play nice with the formal definition of the language?
 - yes, because overlapping works nicely
- $B \text{ ako } A$.
 - instances of B must follow all constraints on A, plus all constraints on B

Merging

- How do TMCL schemas merge?
 - typing topics merge as usual
 - constraint topics usually will not merge
 - this could lead to duplicate constraints
 - there might be conflicting constraints
- Formally,
 - duplicate constraints are no problem (see overlap)
 - conflicting constraints is an insoluble problem, and therefore not our problem
 - may be detectable for basic schemas

Reification

- If reifying topics must have as topic type the type of the construct they are reifying
 - `!img date-of-birth: 1973-12-25 ~!img-birthday .`
 - `!img-birthday isa date-of-birth . #` currently, no spec requires this
- then
 - occurrence types and topic types (as in the above) are no longer disjoint
- The meta-schema must have a position on this
 - **WE ALLOW OVERLAP BETWEEN ONTOLOGY TYPES**
 - that is, `occtype` with `topic type`, `name type` with `topic type` etc
 - `occtype` *does not* overlap with `name type`

Reification, constraints on

- Should it be possible to constrain reification?
 - currently it is not
 - that is, to declare that statements of a certain type must be/can be/must not be reified?
 - should it be possible to constrain the type of the reifying topics
- Example of constraint
 - date-of-birth has-reifier(date-of-birth, 1, 1).
 - means that occurrences of type date-of-birth are required to have a reifying topics of type date of birth
 - date-of-birth has-reifier(tmdm:subject, 0, 0).
 - means they cannot be reified
 - date-of-birth has-reifier(date-of-birth, 0, 1).
 - means they can be reified, but need not be
 - numbers greater than 1 are forbidden

Reification, constraints on, part 2

- Another possible approach
 - date-of-birth must-have-reifier(date-of-birth).
 - date-of-birth cannot-have-reifier().
 - date-of-birth can-have-reifier(date-of-birth).
 - these templates each have their separate constraint type
 - alternatively: the templates translate into cardinalities
- **WE GO FOR THE ALTERNATIVE WITH CARDINALITIES**
 - **however, we use the templates on this slide**



Semantics of card-min / card-max in constraints

- has-subject-identifier
 - applies to all subject identifiers, not just the ones that match the regexp
 - we think this was how it was in the draft
 - however, see next slide
- has-occurrence
 - applies to all occurrences of the specified type
- And so on throughout



Multiple subject-identifier constraints

- If we want to require one PSI of a specific form, but also allow others:
- place
 - `has-subject-identifier(0, *, ".*");`
 - `has-subject-identifier(1, 1, "http://psi\\.example\\.org/.*") .`
- This doesn't work,
 - because if you have 2 PSIs, one `example.org`, and one `ontopedia.net`, then
 - first constraint is satisfied,
 - second constraint is violated by `ontopedia.net` PSI
- Could be resolved with two types of `has-subject-identifier` (and for other identity)
 - one where the regexp is used for matching, and one where it's not
- Could also be resolved if the regexp is always used for matching
 - if so, the regexp should be in front

REGEXP IS USED FOR MATCHING



tmdm:subject

- Make explicit?
 - **YES**



Do we want template overloading in CTM?

- Useful for many different purposes
- Changes to CTM:
 - no actual syntax changes
 - change to template semantics:
 - rules for duplicate template definitions changed a little
 - rules for template invocation ditto
- **YES, WE ADD THIS**



Documentation - 2

- Names of association types
 - labels for associations in particular directions
 - employment
 - employed by
 - employs
 - conventions for representing this in topic maps
 - plus, how to deal with in TMCL meta-schema
- **THIS DOES NOT BELONG IN TMCL**

Arc definitions

- Giving identity and names to a traversal through an association type
- That is, creating a topic that represents
 - a starting association role (e.g. employee),
 - an association type (e.g. employed-by), and
 - an ending association role (e.g. employer)
- Use cases:
 - generating domain-specific classes (turning association types into properties)
 - mapping to directed graphs (such as RDF)
- How to do:
 - define 3-4 PSIs necessary to represent this with a topic type and association types
 - name of topic becomes name of the arc

NOT DOING THIS IN TMCL



MIME types

- Xuân will check this and bring it up on the mailing list



Recommendations to the plenary

- Robert Barta resigns as TMQL editor
- Editors of TMCL instructed to prepare draft for FDIS ballot by 2010-06-01
 - intention is that new draft (not for ballot) be prepared **ASAP**
- Disposition of comments on TMQL FCD in N_____
 - LMG will write next week
- Disposition of comments on TMCL FCD in N_____
 - LMG will write next week
- Disposition of comments on Dublin Core PDTR in N_____
 - LMG will write next week



GTM

- Hannes will write feedback on the current level 1 proposal
 - will post by mid-June