TMDM mapping to TMRM

Lars Marius Garshol
Ontopia
`<larsga@ontopia.net>`
2005-08-11

11th August 2005

1 Introduction

This document describes a proposed representation of TMDM in TMRM. The intention is that this will be reviewed by SC34, and if approved included in ISO 13250-5 as an annex. The document is couched in a style known to be inappropriate for direct inclusion, but, it is hoped, suitable as a basis for discussion.

This mapping uses version 6.0 of the TMRM (no permanent URL known).

1.1 General principles

The key principle for the representation is that it *must not* lose information. This means that the transformation from TMDM to T+ must be reversible in such a way that an TMDM instance identical to the original will always be produced by the two transformations.

Other general principles of the representation are:

- Proxies representing information items should include all the properties that define the identity of the item (to avoid loss of information).
- Proxies should only contain properties that do not change, such that it is more natural to delete a proxy and create a new one than to modify the existing one.
- Anything that can be reified in TMDM must have a separate proxy representing it.

Issue: How to represent multi-value properties? A single key with a set value, or repeating keys?

Issue: Do we want a semantic mapping or an object mapping? Are we concerned with the size of the representation of TMDM in TMRM? Any particular requirements for the shape?

2 Datatypes

For each datatype in TMDM we define a disjunct value set, together with an interpretation function that produces a value from the lexical representation (string value). This enables us to let the value in T+ be the actual intended value, without loss of information. The actual definitions will be added later.

3 Foundational proxies

[List, and define, foundational proxies for the model.]

4 Constructing a TMRM representation

To construct a TMRM representation of a TMDM instance, start with a map containing only the proxies listed in the previous section.

4.1 Topic map item

For the topic map item, construct a proxy m as follows: $\{\langle itemtype, topicmap\rangle\}$.

For each locator l in the topic map item's [item identifiers] property, add a proxy as follows: $\{\langle itemidentifier, l\rangle, \langle construct, m\rangle\}$.

4.2 Topic items

For each topic item construct a proxy t containing the following properties:

- One $\langle subjectlocator, locator\rangle$ property for each value in the topic's [subject locators] property.
- One $\langle subjectidentifier, locator\rangle$ property for each value in the topic's [subject identifiers] property.
- One $\langle itemidentifier, locator\rangle$ property for each value in the topic's [item identifiers] property.
- One $\langle itemtype, topic\rangle$ property.

Note that this way of constructing the topic proxy means that the value identity is not the same as the specified identity rules in TMDM. This implies that merging functions need to be defined.

The [roles played] property is computed, and so can be ignored. The [topic names] and [occurrences] properties are taken care of below.

If the [reified] property is not null, add a proxy as follows: $\{\langle reifier, t\rangle, \langle reified, t.[reified]\rangle\}$.

Issue: Do we want to be able to represent multiple topic maps in the same model? If so, we should also represent the [parent] property on topic items.

4.3 Topic name items

For each topic name item n construct a proxy containing the following properties:

- $\langle value, n.[value] \rangle$,
- $\langle type, n.[type] \rangle$,
- one $\langle scope, t \rangle$ property for each topic t in the [scope] property of the topic name item,
- $\langle parent, n.[parent] \rangle$,
- $\langle itemtype, topic - name \rangle$

Issue: Should we create a proxy for the scope instead?

For each locator l in the topic name item's [item identifiers] property, add a proxy as follows: $\{(itemidentifier, l), (construct, n)\}$.

4.4 Variant items

For each variant item v construct a proxy containing the following properties:

- $\langle value, v.[value] \rangle$,
- one $\langle scope, t \rangle$ property for each topic t in the [scope] property of the variant item,
- $\langle parent, v.[parent] \rangle$,
- $\langle itemtype, variant \rangle$

For each locator l in the variant item's [item identifiers] property, add a proxy as follows: $\{\langle itemidentifier, l \rangle, \langle construct, v \rangle\}$.

Issue: Deal with the datatype.

4.5 Occurrence items

For each occurrence item o construct a proxy containing the following properties:

- $\langle value, o.[value] \rangle$,
- $\langle type, o.[type] \rangle$,
- one $\langle scope, t \rangle$ property for each topic t in the [scope] property of the occurrence item,

- $\langle parent, o.[parent]\rangle$,
- $\langle itemtype, occurrence\rangle$

For each locator l in the occurrence item's [item identifiers] property, add a proxy as follows: $\{(itemidentifier, l), (construct, o)\}$.

Issue: Deal with the datatype.

4.6 Association items

For each association item a create a proxy with the following properties:

- $\langle type, a.[type]\rangle$,
- one $\langle scope, t\rangle$ property for each topic item t in a.[scope],
- one $\langle r.[type], r.[player]\rangle$ property for each association role item r in a.[roles],
- One $\langle itemtype, association\rangle$ property.

For each locator l in the association item's [item identifiers] property, add a proxy as follows: $\{\langle itemidentifier, l\rangle, \langle construct, a\rangle\}$.

Issue: this loses support for reifying association roles. Creating a proxy for each role is *not* not sufficient to solve this.

Issue: this uses the TMDM representation of type-instance instead of that used by T+.

5 Merging

We define a $\bowtie$ function, corresponding to merging in TMDM. However, in order to do this a number of supporting functions are necessary.

We define a function that given a topic produces the full set of topics which are known to represent the same subject as it. This is deliberately made to include the topic itself.

$$e(x) = \{y | \langle itemtype, topic\rangle \in y \wedge y \cap x \neq \langle itemtype, topic\rangle\} \tag{1}$$

Given this, we can define a function that given a proxy returns the proxy as it looks in the merged map. For topics it produces the maximally merged set of identities, whereas for other proxies it simply replaces all topics occurring in the proxy with the maximally merged ones. This makes the function recursive, but it's a one-step recursion, and so guaranteed to produce a result.

$$m(x) = \begin{cases} \bigcup_{y \in e(x)} y & \langle itemtype, topic \rangle \in x \\ \{p | \exists \langle k, v \rangle \in x : p = \langle m(k), m(v) \rangle\} & \text{otherwise} \end{cases} \tag{2}$$

Now we are finally ready to do the $\bowtie$ function, as follows:

$$x \bowtie y = m(x), \text{if } x = y \tag{3}$$

6 Acknowledgements

This document owes much to an earlier document by Robert Barta and Lars Heuer, and discussions with Lars Heuer during the writing.