TM/XML

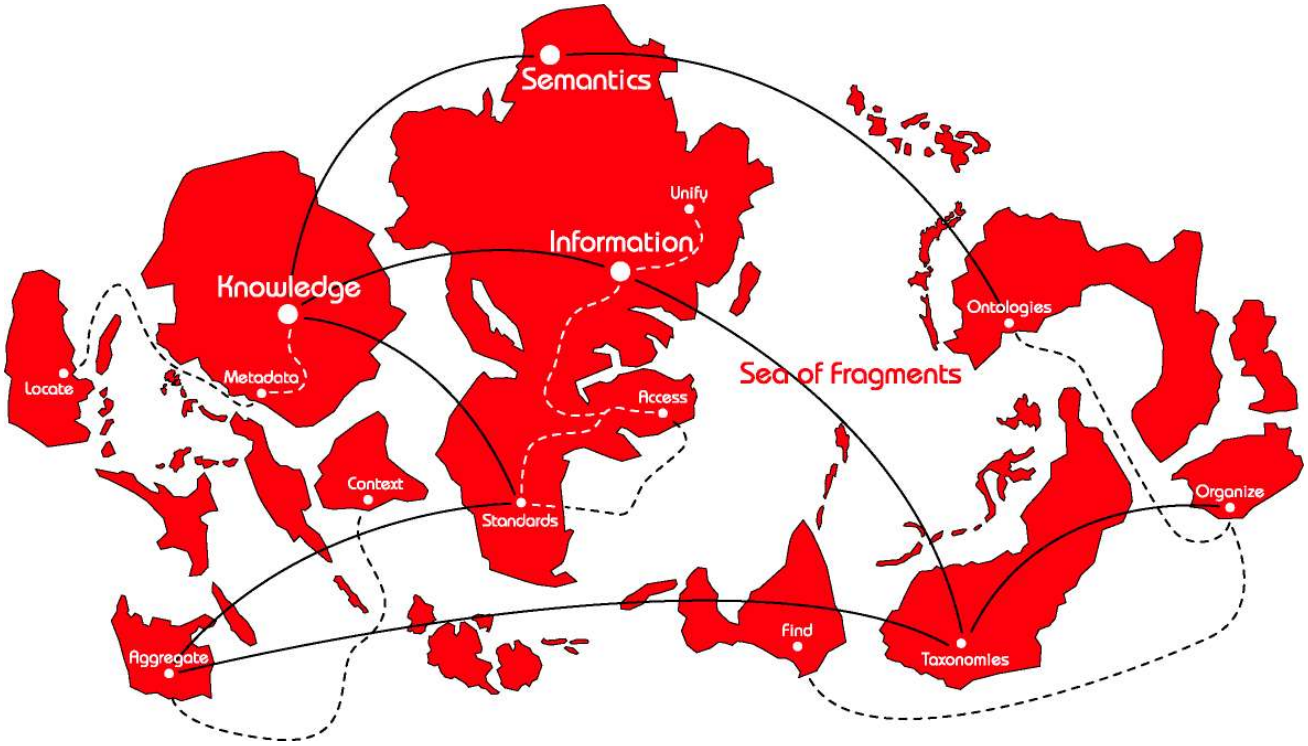
A human-friendly XML syntax for Topic Maps

TMRA '05

Lars Marius Garshol
Ontopia
<larsga@ontopia.net>

Dmitry Bogachev
Omega Consulting
<dmitryv@cogeco.ca>

2005-10-05



Overview

- **Introduction**
  - why another XML syntax for Topic Maps?
- **The syntax**
  - how it works
  - examples
  - comparison with XTM
- **TMViews**
  - using TM/XML with fragments
- **Conclusion**
  - status
  - further work

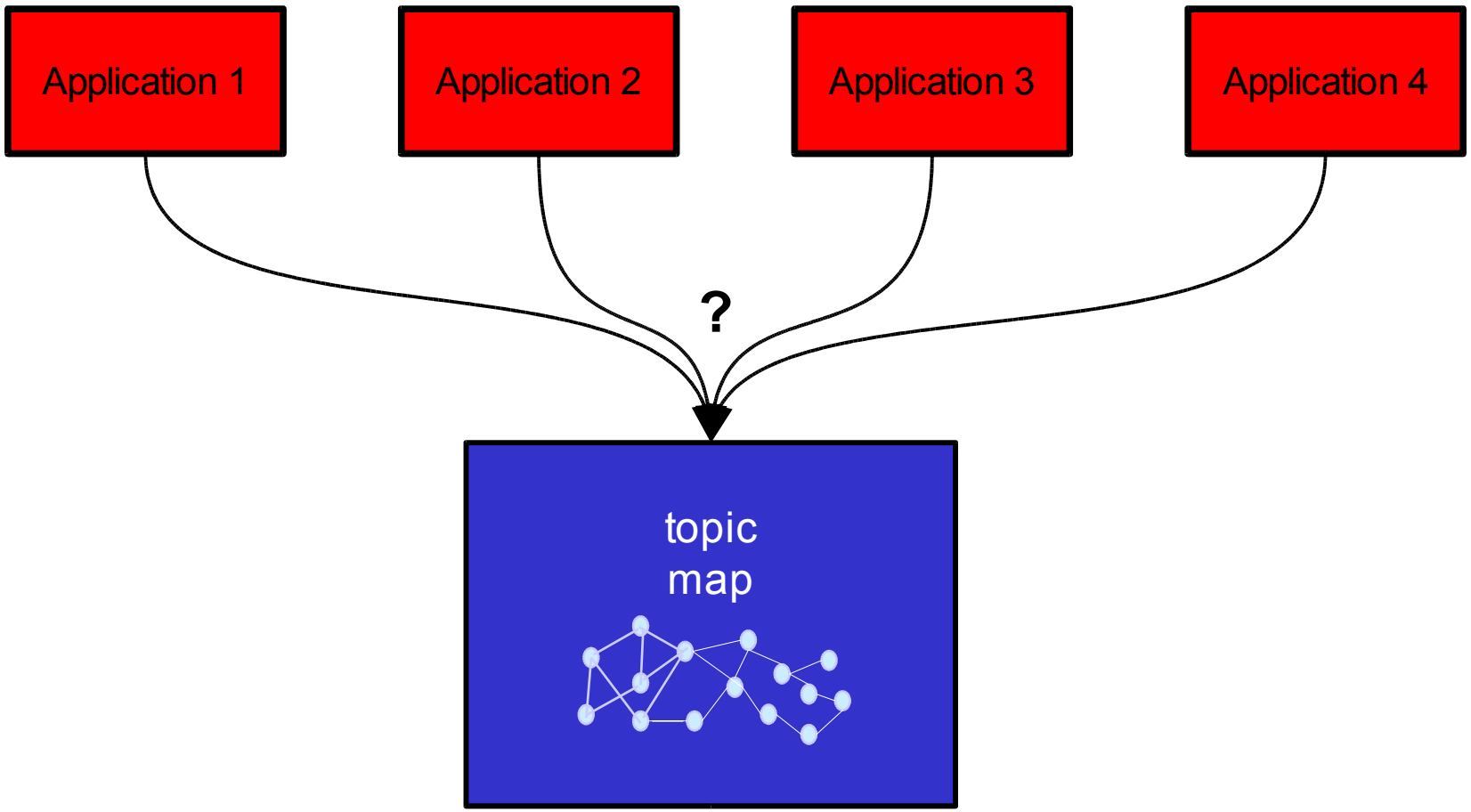
Introduction

Why TM/XML?

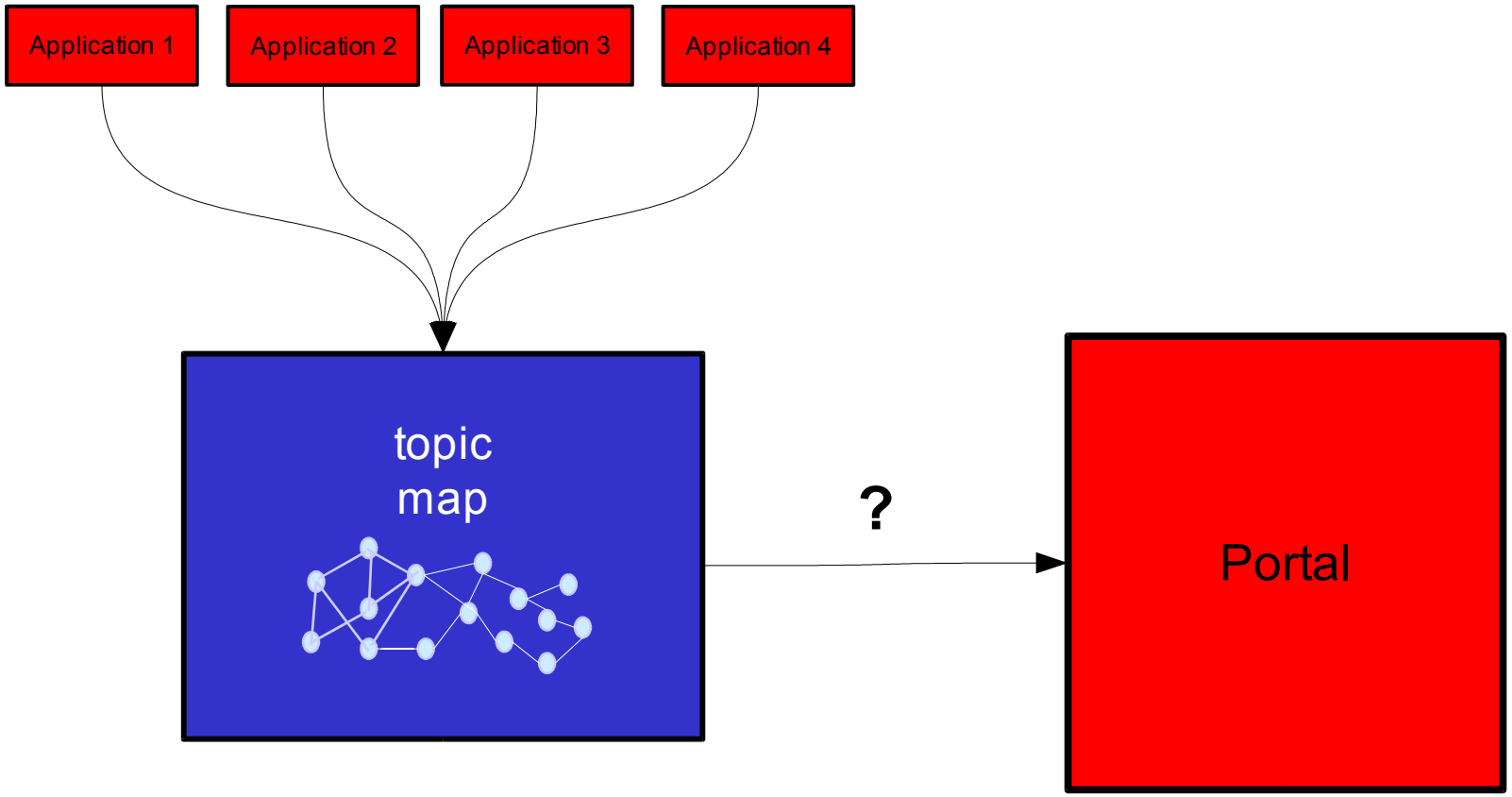
Why TM/XML? What's wrong with XTM?

- **As an interchange syntax XTM is fine**
  - it's not perfect, but few things are
  - it is *not* our goal that TM/XML should compete with XTM
  - we have other uses in mind for TM/XML
- **Basically, TM/XML is intended for use when integrating with non-Topic Maps systems**
- **Goals**
  - minimize learning curve for integrators
  - remove need for Topic Maps software in clients
  - produce as natural XML as possible

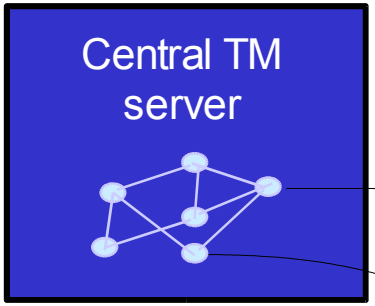
Knowledge hubs

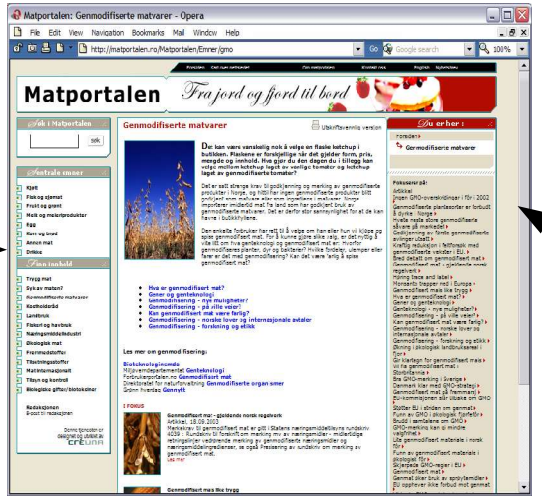


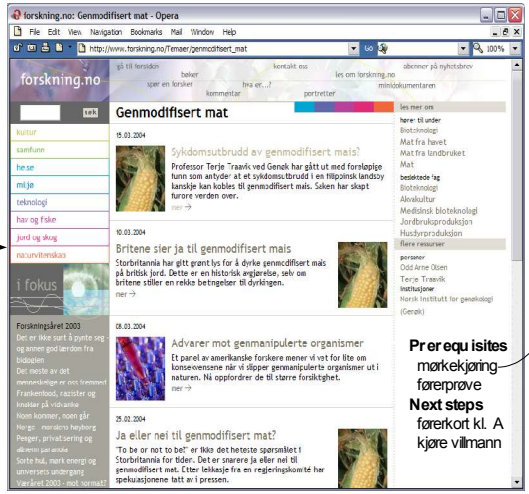
Using the hub in a portal



Connecting portals







The TM/XML syntax

How it works

Examples

Basic idea

- **An XML syntax for Topic Maps is effectively a mapping between two data models**
  - that of XML, and that of Topic Maps
- **XTM represents an object mapping**
  - that is, each TMDM construct has an element of its own in XTM
  - the XML vocabulary is generic, and formulated in terms of the TMDM model
  - this is why XTM is so difficult to process with ordinary XML tools
- **TM/XML represents a semantic mapping**
  - that is, the vocabulary of the topic map is mapped into XML
  - the XML vocabulary reflects the terms from the domain of the topic map
  - the result is relatively natural XML, something a human being might have written to express the same information
  - this is why TM/XML is easy to process with ordinary XML tools

An example topic map in LTM

```
[lmg : person = "Lars Marius Garshol"]

{lmg, homepage, "http://www.garshol.priv.no"}


[ontopia : company = "Ontopia"]

{ontopia, homepage, "http://www.ontopia.net"}


employed-by(lmg : employee, ontopia : employer)
```

Conversion, 1

```
<topicmap ...>
  <person id="lmg">
```

[lmg : person ...

- **We start with the 'lmg' topic**
- **Make an element type name from the topic type**
  - subject identifiers turn into namespace names
  - if none exist, we create names from the item identifiers
- **If the topic has no subject identifiers or locators, create an ID**
  - put this in the “id” attribute

Conversion, 2

```
<topicmap ...>                                  [lmg : person = "Lars Marius Garshol"]
  <person id="lmg">
    <iso:topic-name><tm:value>Lars M. Garshol</tm:value></iso:topic-name>
```

- **We create an element type name from the type of the topic name**
- **In this case it's the default name type defined by ISO 13250**
- **Scope and reifier would have been captured with attributes**
- **The <tm:value> element exists so we can support variants**

Conversion, 3

```
<topicmap ...>
  <person id="lmg">
    <iso:topic-name><tm:value>Lars M. Garshol</tm:value></iso:topic-name>
    <homepage datatype="...#anyURI">http://www.garshol.priv.no</homepage>
```

```
{lmg, homepage,
      "http://www.garshol.priv.no"}
```

- **We create an element type name from the occurrence type**
- **The value of the occurrence becomes the element content**
- **Datatype is captured in an attribute**
- **Scope and reifier as for topic names**

Conversion, 4

```
<topicmap ...>
  <person id="lmg">
    <iso:topic-name><tm:value>Lars M. Garshol</tm:value></iso:topic-name>
    <homepage datatype="...#anyURI">http://www.garshol.priv.no</homepage>
    <employed-by topicref="ontopia" role="employee"/>
```

employed-by(lmg : employee, ontopia : employer)

- **Again, element type created from association type**
- **Role captured in an attribute**
- **topicref refers to player of other role**
- **Scope and reifier as before**
- **(N-aries and unaries handled differently; see paper for details)**

Full result

```
<topicmap xmlns:iso="http://psi.topicmaps.com/iso13250/"
    xmlns:tm="http://psi.ontopia.net/xml/tm-xml/">
  <person id="lmg">
    <iso:topic-name><tm:value>Lars M. Garshol</tm:value></iso:topic-name>
    <homepage datatype="...#anyURI" >http://www.garshol.priv.no</homepage>
    <employed-by topicref="ontopia" role="employee"/>
  </person>

  <company id="ontopia">
    <iso:topic-name><tm:value>Ontopia</tm:value></iso:topic-name>
    <homepage datatype="...#anyURI" >http://www.ontopia.net</homepage>
    <employed-by topicref="lmg" role="employer"/>
  </company>
</topicmap>
```

Comparison

- **LTM original**
  - 5 significant lines
  - employed-by(%company% : employer, $E : employee)?
- **XTM**
  - 48 significant lines
  - //xtm:association
    [xtm:member [xtm:roleSpec / xtm:topicRef / @xlink:href = '#employer']
    [xtm:topicRef / @xlink:href = concat('#', $company)]]
    [xtm:instanceOf / xtm:topicRef / @xlink:href = '#employed-by']
- **TM/XML**
  - 14 significant lines
  - //person [employed-by/@topicref = $company]

RELAX-NG schema for TM/XML

```
topicmap = element * { topic+ }
topic =
    element * { id?, identifier*, locator*, topicname*, occurrence*, association* }

identifier = element identifier { xsd:anyURI }
locator = element locator { xsd:anyURI }

topicname = element * { reifier?, scope?, value, variant* }
variant = element variant { scope, reifier?, datatype?, text }
occurrence = element * { reifier?, scope?, datatype?, text }
association = unary | binary | nary
unary = element * { reifier?, scope?, role }
role = attribute role { text }
binary = element * { reifier?, scope?, role, topicref }
nary = element * { reifier?, scope?, assocrole, assocrole, assocrole+ }
assocrole = element * { reifier?, text }
```

TM-Views

What is it?

What is it for?

TM-Views

- **TM/XML was created to be used with fragments**
- **How are fragment boundaries to be determined?**
  - sometimes you want the whole topic
  - sometimes you want the topic plus neighbours
  - sometimes you want the topic plus typing topics
  - sometimes you want just the names
  - ...
- **How to handle fragment updates?**
  - if the information about topic X comes from sources A and B, how can A update X without overwriting B's data?
  - this can be solved with custom programming, but the point of merging is to avoid that
  - TM-Views enables A and B to describe what each is responsible for
  - each can then update its own view of topic X

An example

```
<view id="person_view" name="Person view">
  <topic type="person">
    <identifier type="subjectIdentifier" />
    <basename type="*">
      <except> <basename type="nickname" /> </except>
    </basename>
    <occurrence type="homepage" />
    <association type="employed-by" start_role="employer" role="employee">
      <topic type="company"> <identifier type="*"/>
        <basename type="*"/>
        ...
      </topic>
    </association>
  </topic>
</view>
```

Conclusion

Status

Further work

Status of work

- **The specification is written**
  - basically the paper in the proceedings
- **A prototype has been written**
  - using Jython on top of OKS
  - was used to produce the examples
- **Will be implemented in OKS**
  - as part of web service interface
  - integrated as part of TMRAP

Further work

- **Solving the update problem**
  - how to update the topic map when external information changes?
  - should be possible to just push in a new fragment
  - TM-Views is part of the solution
- **Schema generation**
  - life is much easier for developers if they can get a schema for the TM/XML data they receive/send
  - this can be automatically generated given a schema for the topic map
  - precise details still need to be worked out