Integration of Heterogeneous GML Sources

Gunnar Misund
Associate Professor
Head of Environmental Computing
-
Harald Vålerhaugen
M. Sc. Student
-
Østfold University College
Faculty of Computer Science
Halden, Norway

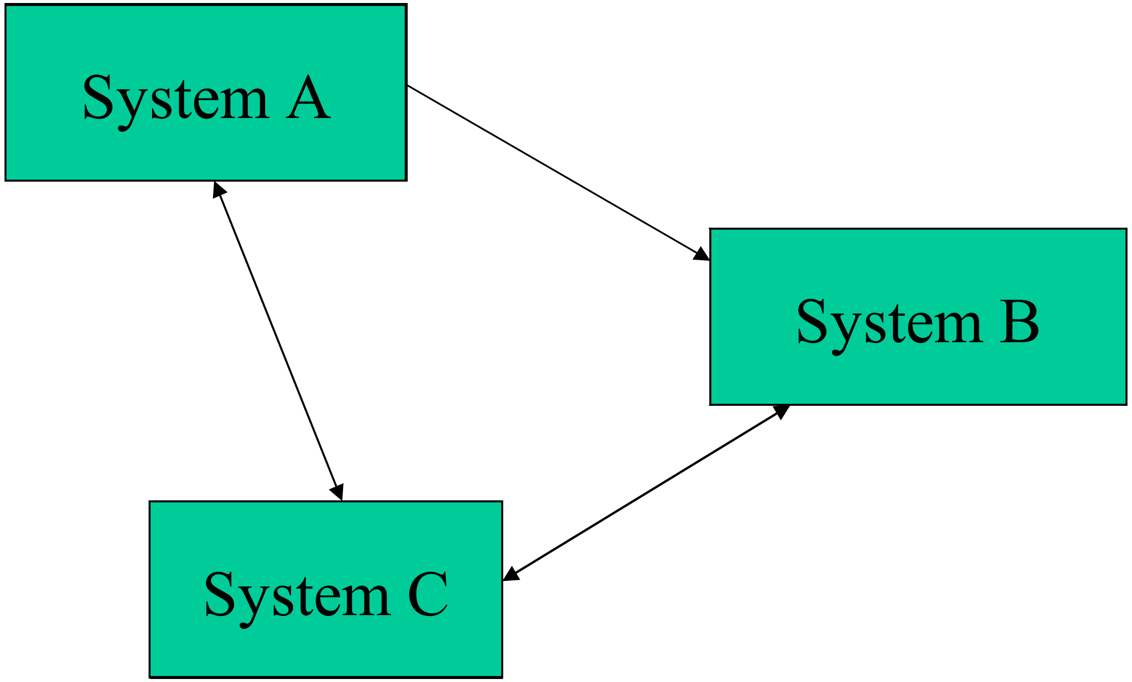
Outline

- Introduction
  - GML: Enabling or barring interoperation?
- Cascading GML Analysis
  - Schema Analysis
  - Structural Analysis
  - Cascading Process
- Lazy Integration
  - Project OneMap
  - Integrating Schemas
- Generic GML Browser
- Final Remarks

Interoperability

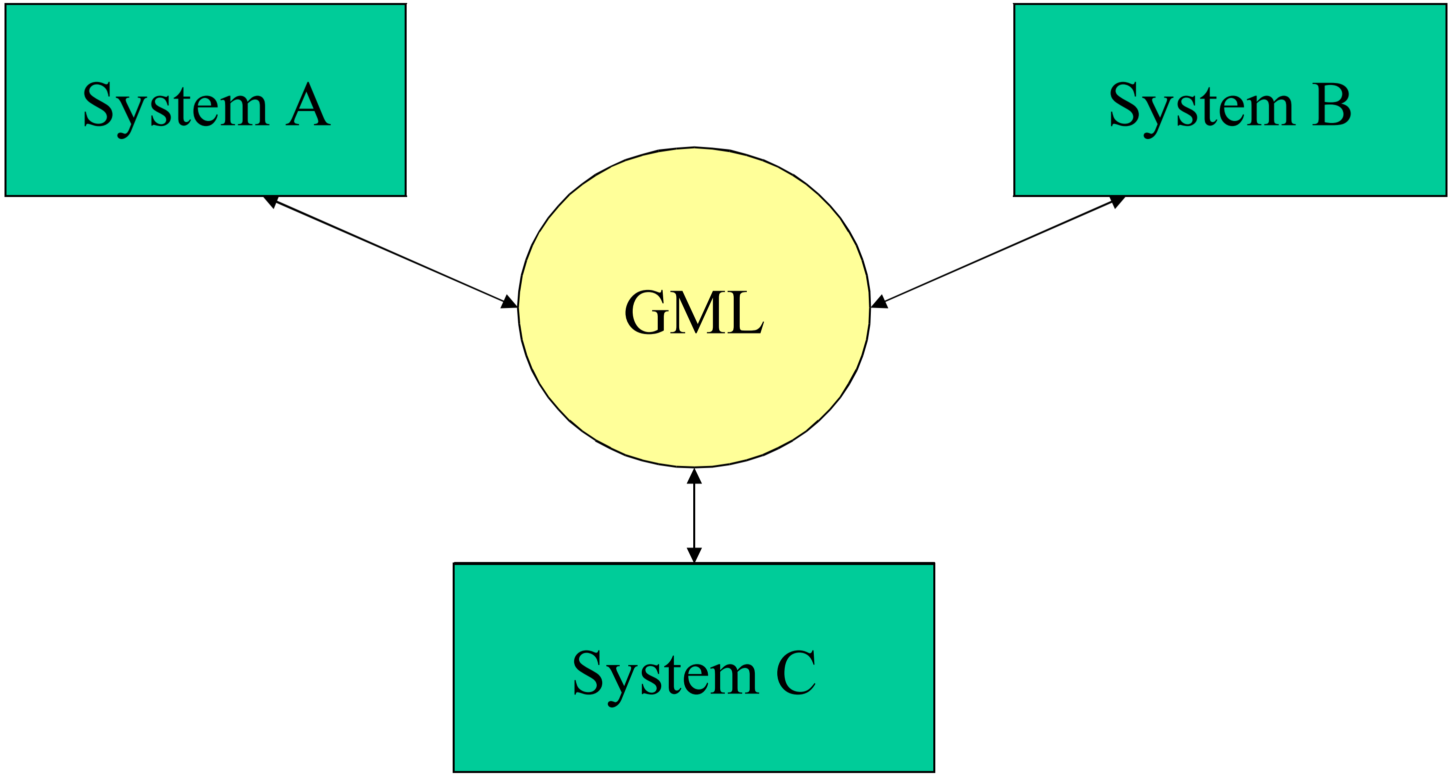
- GML is developed for storage, as well as sharing an interchange of of geographic information.
  - Over Internet
  - Between systems
- Based on schemas from version 2
  - Utilizing namespaces
  - GML 2.x a small specification
  - GML 3.x complex specification

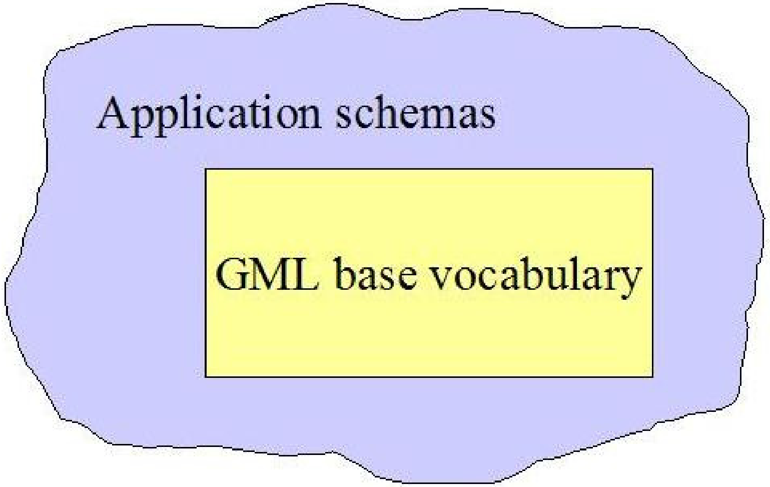
Classical view, proprietary formats



”Speaking” in GML

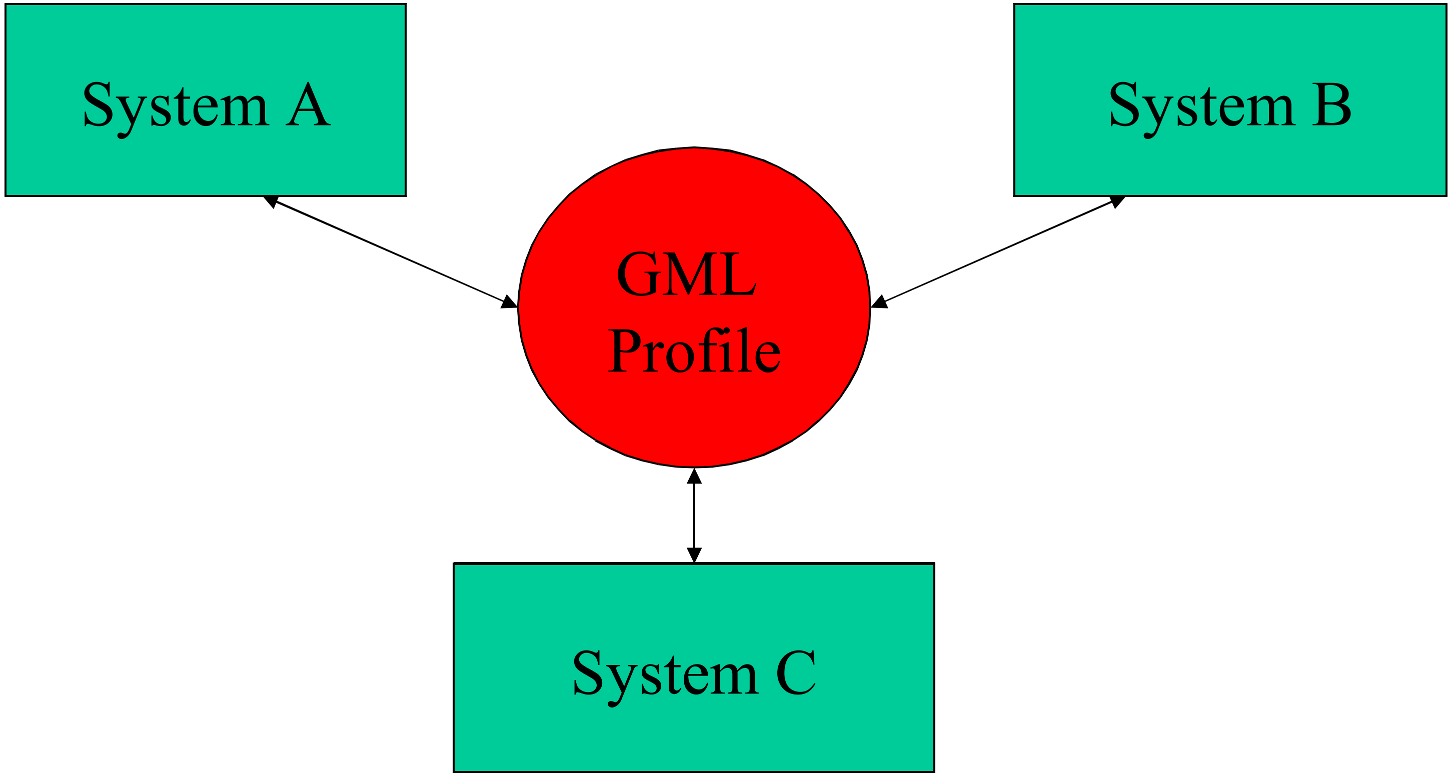
Ideal world

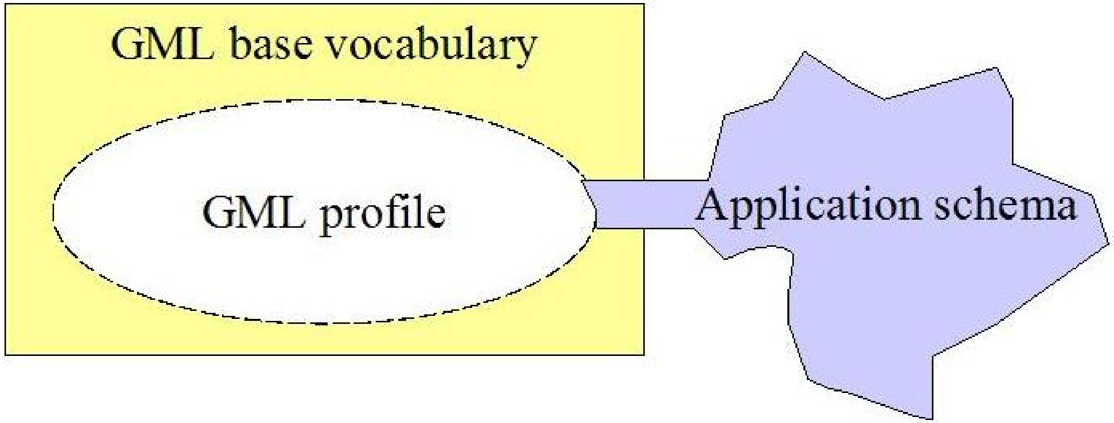



Application schemas
GML base vocabulary

”Speaking” in GML (2)

Real world



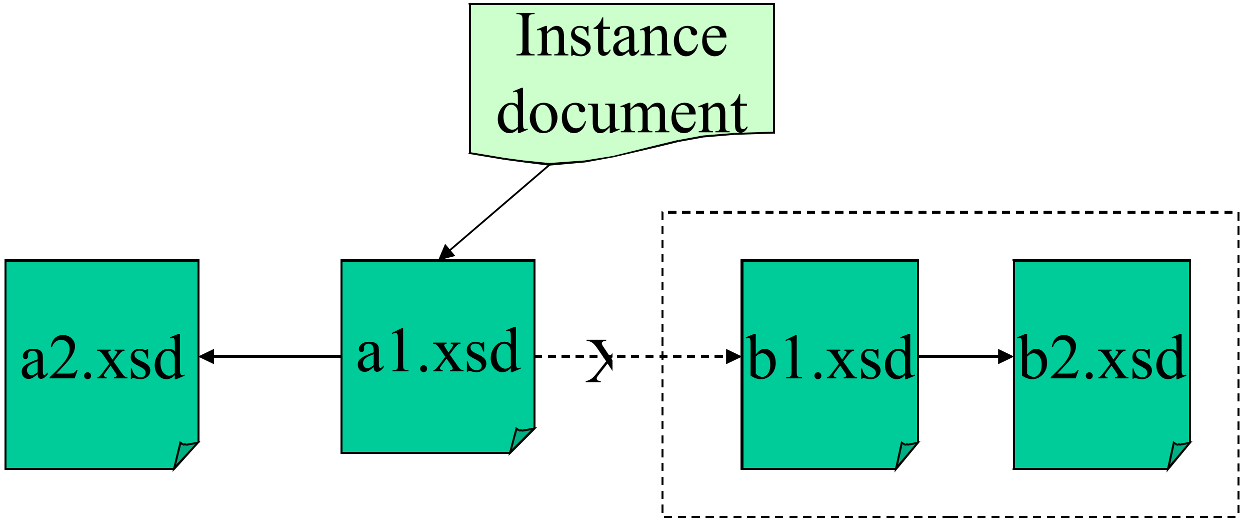

GML base vocabulary
GML profile
Application schema

Why?

- GML is complex, especially GML 3
- Loose restrictions
  - eternal nesting of *FeatureCollections*
  - *non-homogeneous features in same layer*
- The use-cases vary
- Too easy to design poor application schemas
- Lack of (open) software for utilization of arbitrary GML

Everyday GML problems

- Unreliable networks
- Documents not entirely in accordance with schemas
- Invalid schemaLocation-attribute (files, WFS)



Cascading GML Analysis

- Background
- Methods
  - Schema parsing
  - Structural analyzis
  - Manual mapping
  - The bundle

Why cascading method?

- More reliable
- Inconsistency within schemas
- Partially or completely unavailable schemas

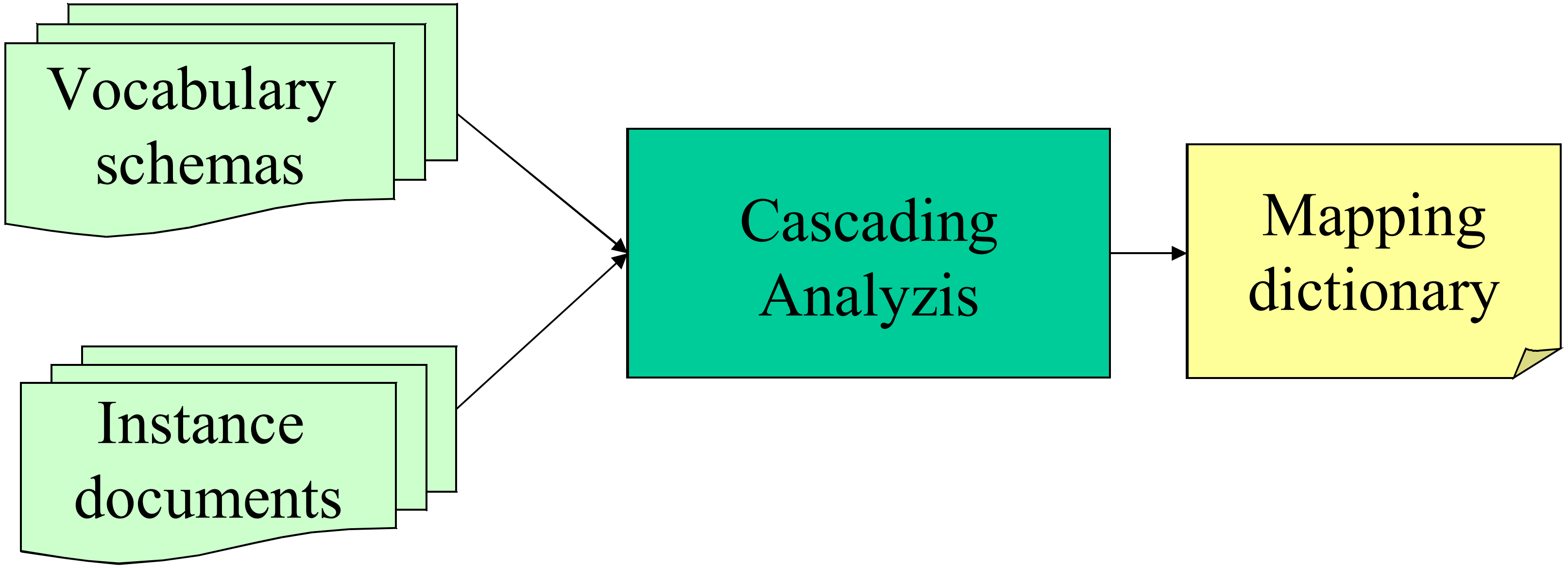
One goal

Find a way to utilize arbitrary GML data in a generic way.

One challenge

Make the method so good, that it becomes an alternative to convert GML into our own GML vocabulary format before utilization.

Generic GML mapping



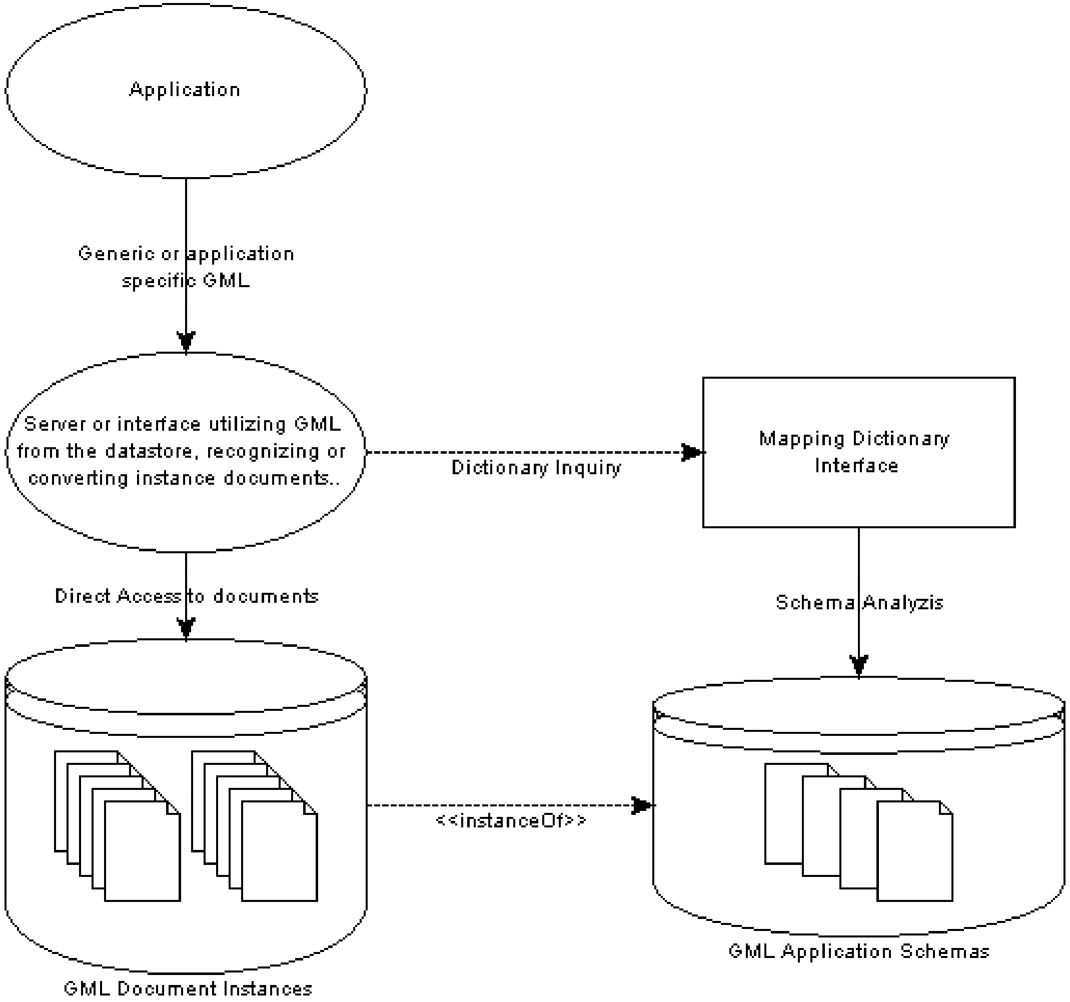
Vocabulary schemas

Instance documents

Cascading Analyzis

Mapping dictionary

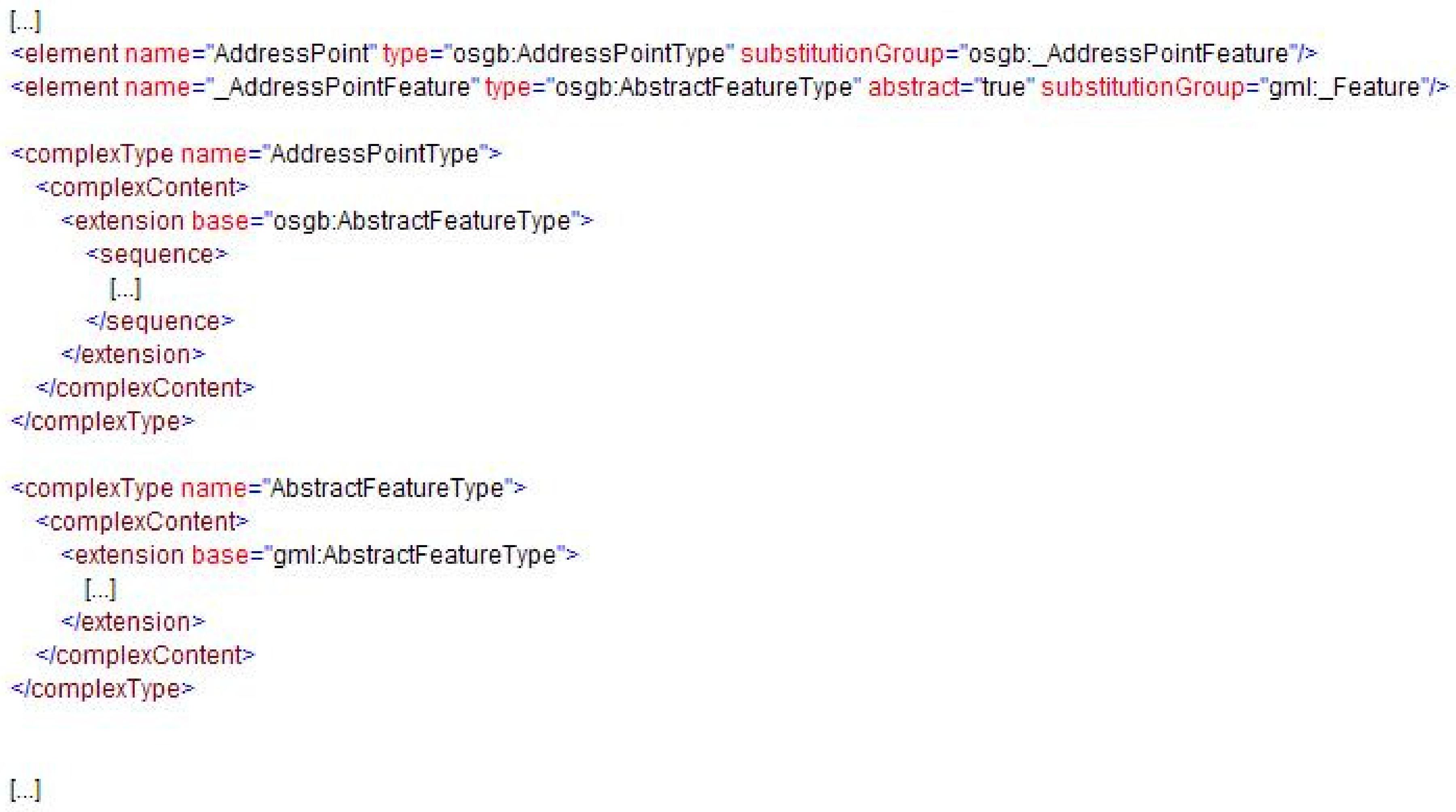
Generic GML utilization


Application
Generic or application specific GML
Server or interface utilizing GML from the datastore, recognizing or converting instance documents..
Dictionary Inquiry
Mapping Dictionary Interface
Direct Access to documents
Schema Analyzis
<<instanceOf>>
GML Document Instances
GML Application Schemas

What data are crucial

- GML application schemas must derive abstract types.
- Non-abstract elements are represented in instace documents, *Complex*- and *SimpleT*ypes are not.
- (In GML) Elements are either instantiations of types declared in the target namespace, in another namespace or ”locally” defined elements.
- They are also either globally defined, or anonymously defined within other elements or type-declarations.

An element ...



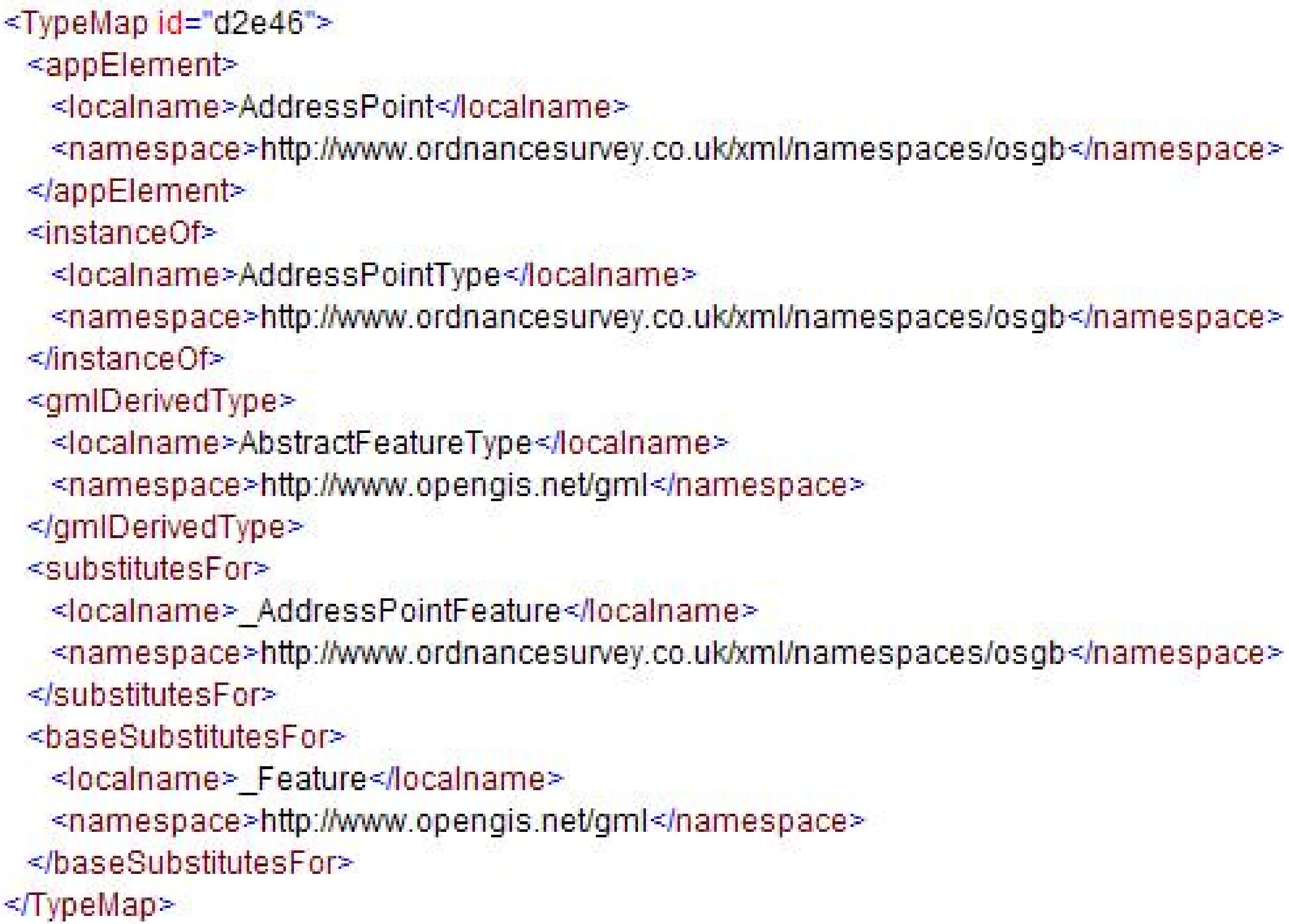
```
[...]
<element name="AddressPoint" type="osgb:AddressPointType" substitutionGroup="osgb:_AddressPointFeature"/>
<element name="_AddressPointFeature" type="osgb:AbstractFeatureType" abstract="true" substitutionGroup="gml:_Feature"/>

<complexType name="AddressPointType">
  <complexContent>
    <extension base="osgb:AbstractFeatureType">
      <sequence>
        [...]
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="AbstractFeatureType">
  <complexContent>
    <extension base="gml:AbstractFeatureType">
      [...]
    </extension>
  </complexContent>
</complexType>


[...]
```

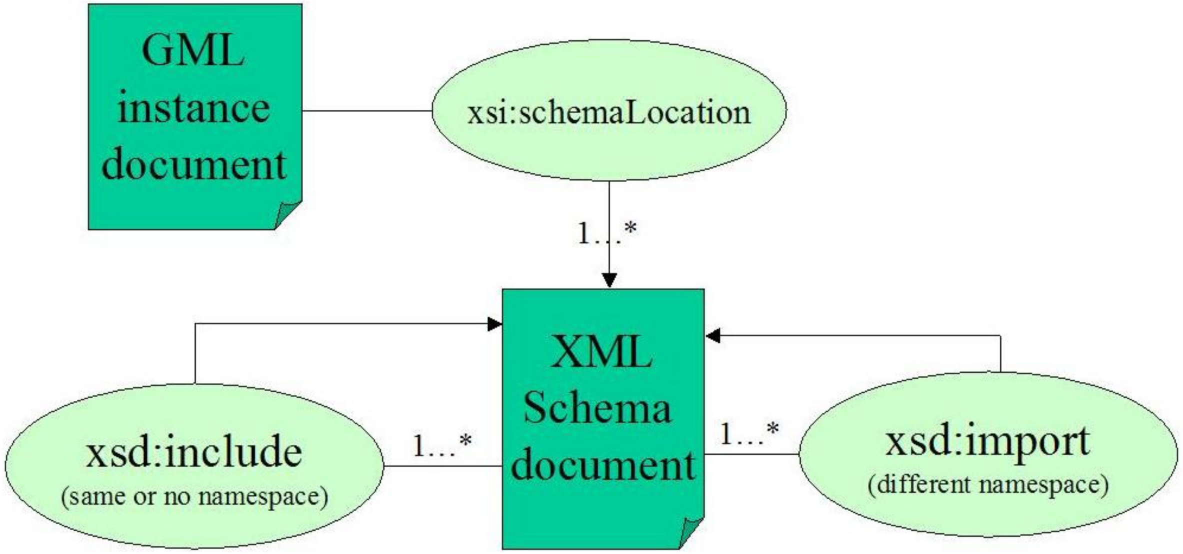
… and a TypeMap



```
<TypeMap id="d2e46">
  <appElement>
    <localname>AddressPoint</localname>
    <namespace>http://www.ordnancesurvey.co.uk/xml/namespaces/osgb</namespace>
  </appElement>
  <instanceOf>
    <localname>AddressPointType</localname>
    <namespace>http://www.ordnancesurvey.co.uk/xml/namespaces/osgb</namespace>
  </instanceOf>
  <gmlDerivedType>
    <localname>AbstractFeatureType</localname>
    <namespace>http://www.opengis.net/gml</namespace>
  </gmlDerivedType>
  <substitutesFor>
    <localname>_AddressPointFeature</localname>
    <namespace>http://www.ordnancesurvey.co.uk/xml/namespaces/osgb</namespace>
  </substitutesFor>
  <baseSubstitutesFor>
    <localname>_Feature</localname>
    <namespace>http://www.opengis.net/gml</namespace>
  </baseSubstitutesFor>
</TypeMap>
```

Schema parsing

- Schemas are primary source for document description
- Straight forward



xsi = ” http://www.w3.org/2001/XMLSchema-instance”

xsd = ”http://www.w3.org/2001/XMLSchema”

Schema parsing (2)

- Stores an XML datastructure with information regarding if, and how the application types are descended from GML types.
- Does not store the structuring rules, nor the restrictions defined in schemas.
  - Thus any editing of the data requires lookup in the schemas.

Schema parsing implementation

- Purely XSLT, using SAXON8 Basic, with XSLT 2.0 ”basic” conformance
- Command line parameters specify schema locations, or alternatively instance documents, or direct WFS GetFeature call (GET)
- Mapping of elements are done as complete as possible, following both *import* and *include* statements in schema files.

Structural and relational analyzis

- A backup solution!
- Reverse engineering
- Parses instance documents, with the purpose of resolving all un-mapped elements.
- Relations between elements
  - Must possibly rely on data being constructed using ”best practice” guidelines
- How much information can we actually gain from structural analyzis, and how reliable is it?
- What about GML3?

S&R analyzis, implementation

- Tree-structure
  - Resolving is based on *neighbour*-, *children*-, and *parentnodes*
- Method
  - Parsing large files, should be done using SAX
  - DOM should be handled with care, but offers the tree-model view
  - SAX, and a proprietary tree-model mapping, is a compromise of the two!

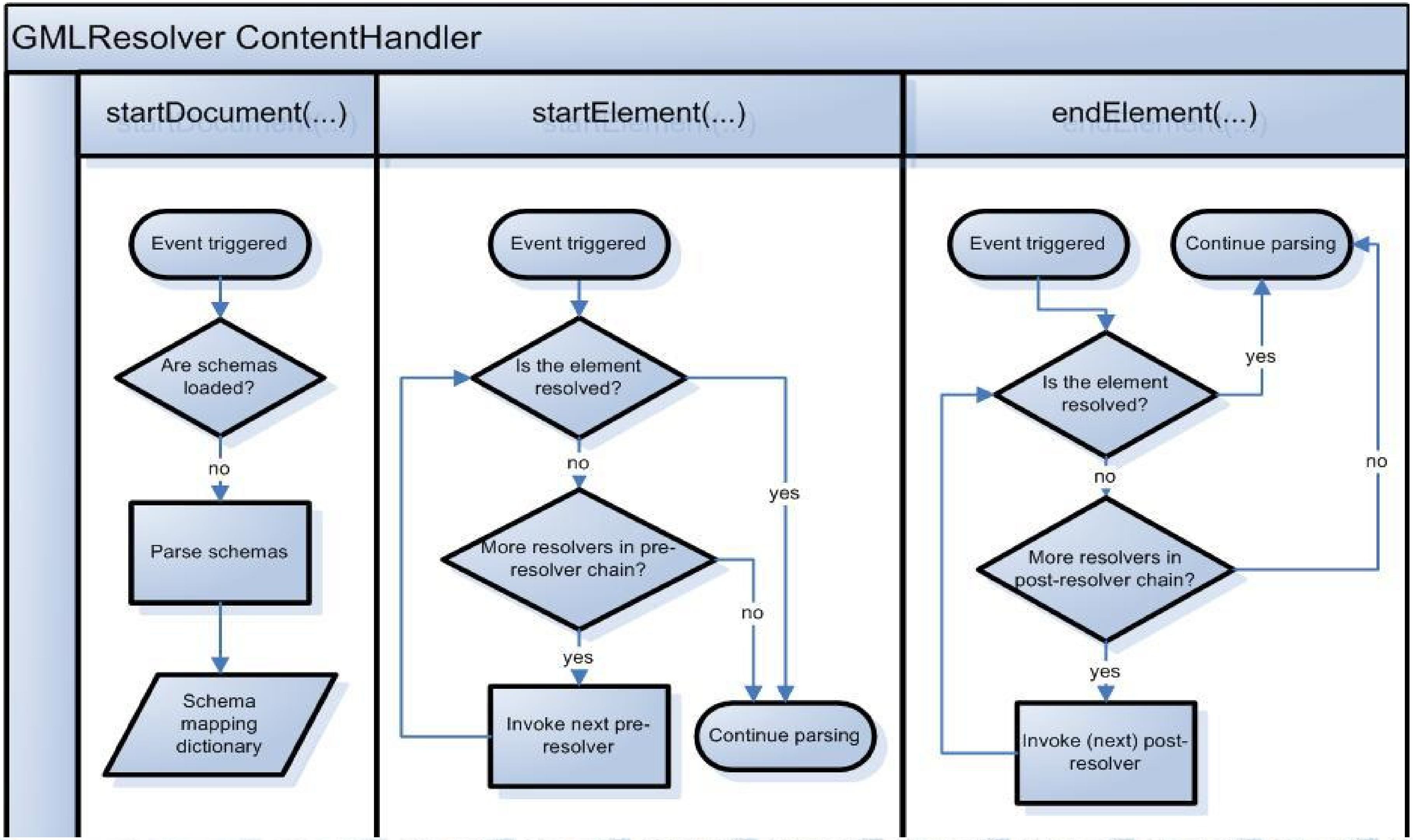
Manual mapping

- Last way out, implemented in a framework with the other two methods...

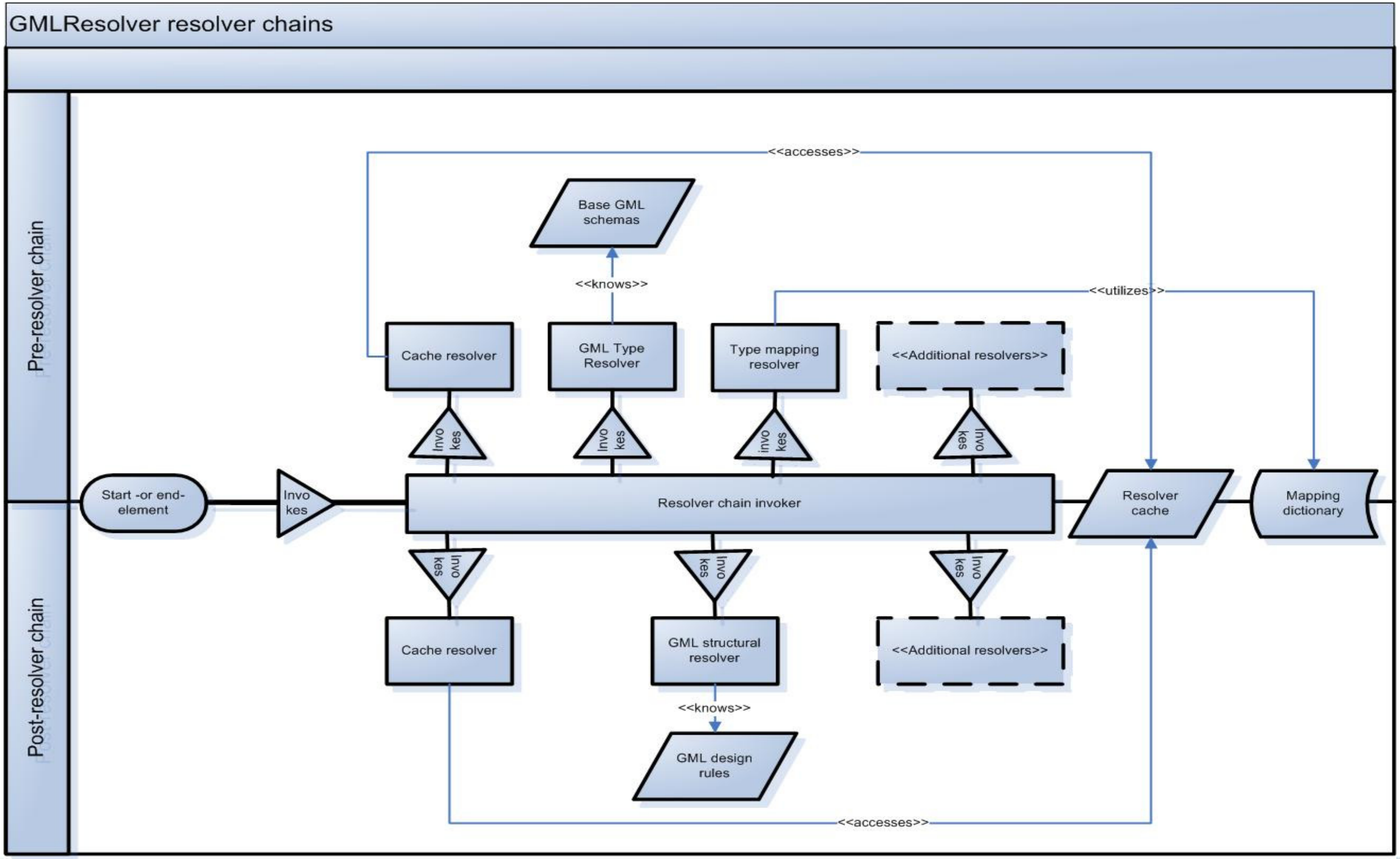
The cascading framework

- Extensible framework, implemented using Java and JAXP
- Different resolvers implements the *TypeResolver* interface. This is where the logic goes!
- Main resolver type: A mapping dictionary resolver
- For each unresolved element, resolvers are invoked in sorted order, until element is resolved or there are no more resolvers.
- Pre- and postresolving

Cascading framework: ContentHandler



Cascading framework: resolver chains

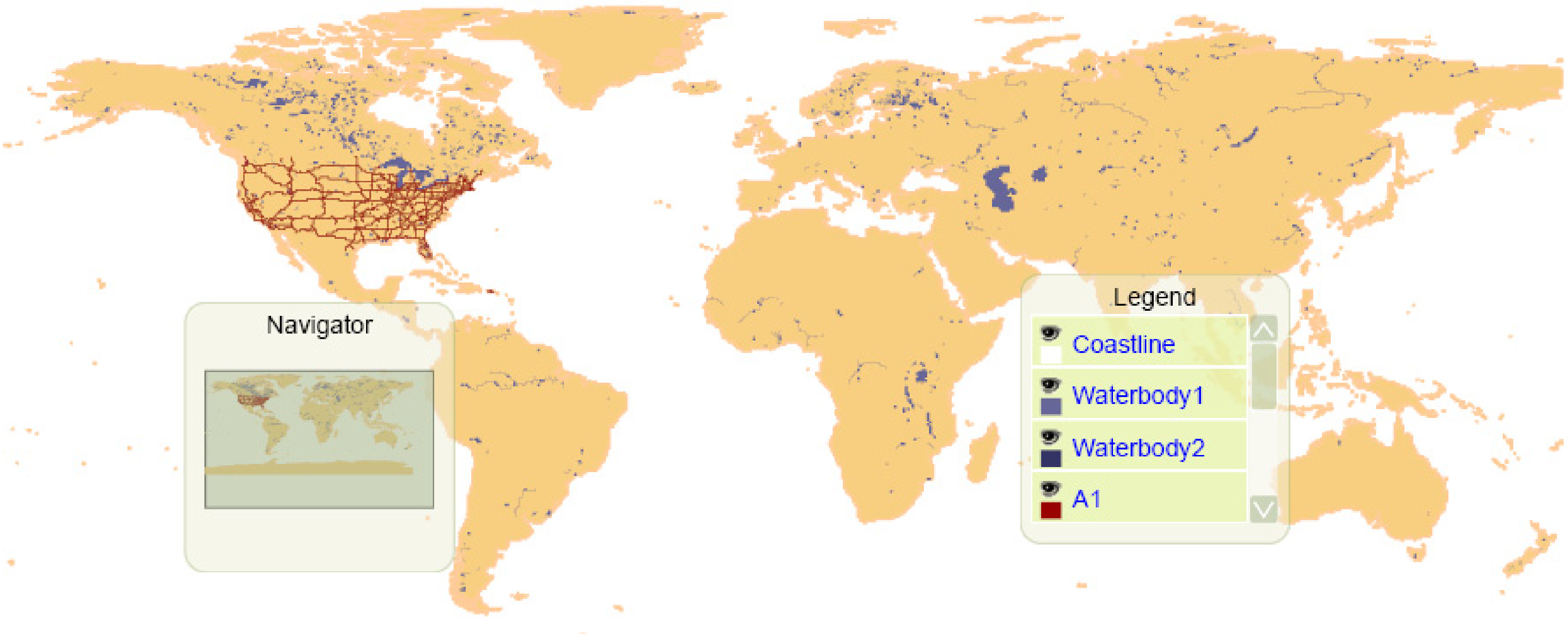


Project OneMap

- Implementation
  - Using Open Content, Open Source and Open Tools
- Open for public use since two years ago
  - Serves both vector data (WFS) and raster data (WMS)
- Used as a testbed for the realization of different services
  - Project based (by students)

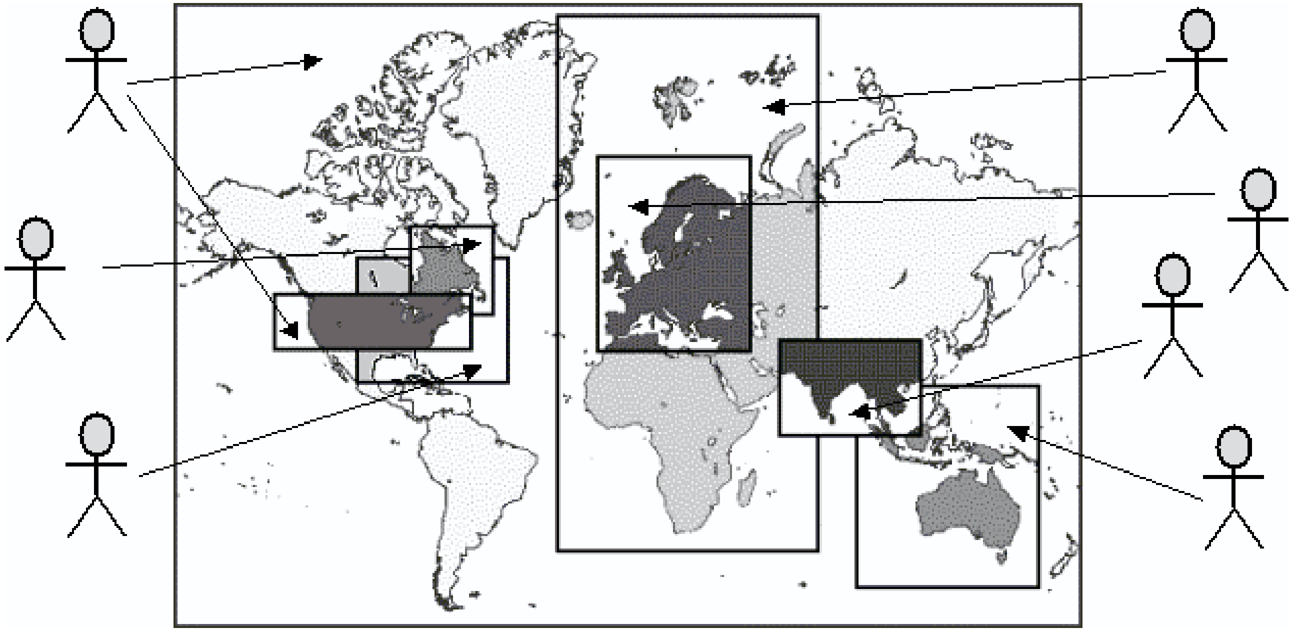
Demo – The OneMap Gateway

- Built solely by using SVG and JavaScript
- GML is transformed into SVG on the Server Side and loaded directly into the SVG plugin



Incremental Map Construction

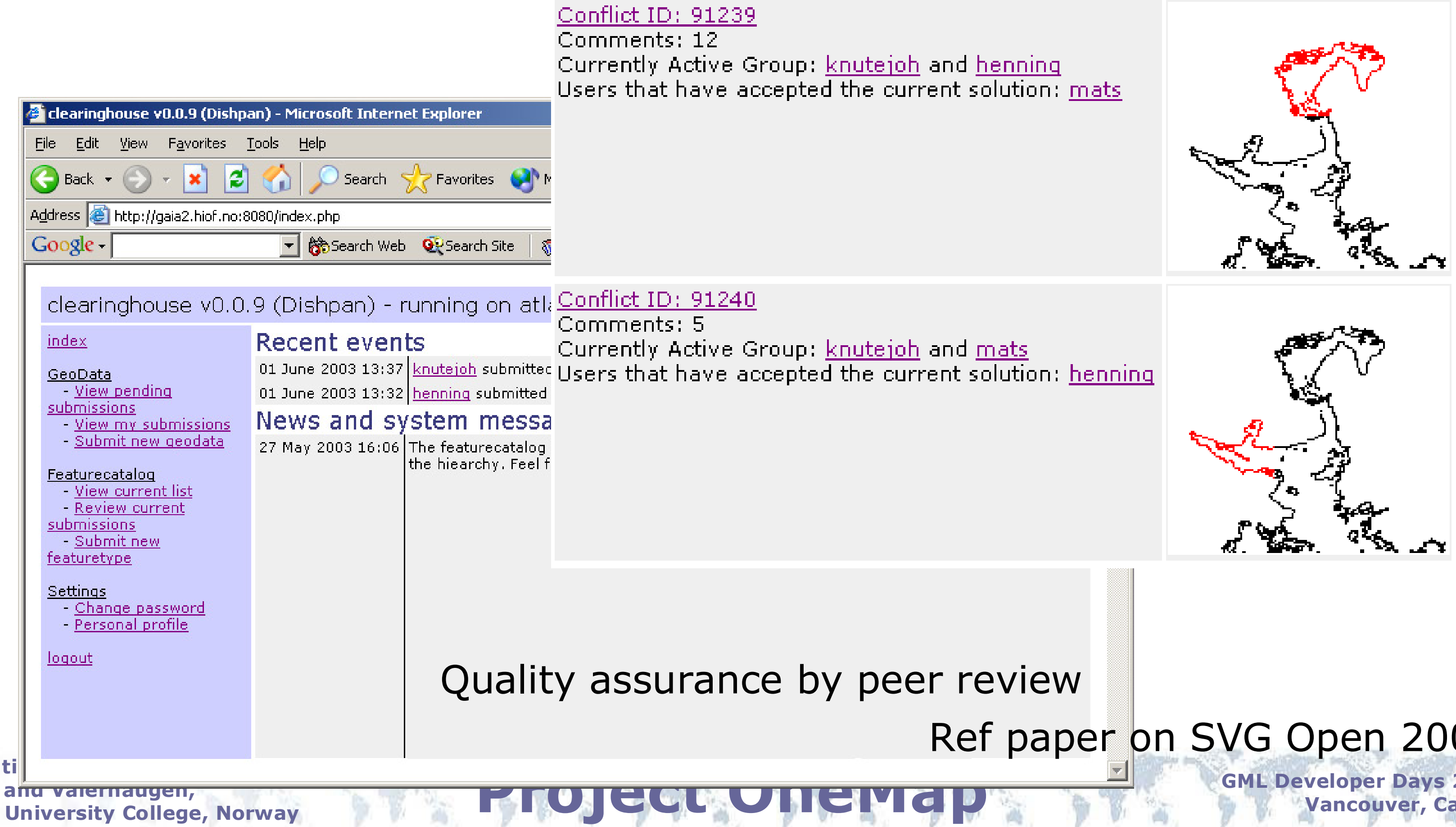
Submissions will be harmonized and accepted/rejected in peer review processes.



The Feature Catalog will be dynamically constructed and maintained...also by peer review processes.

OneMap Clearinghouse

Any party or person may submit their geodata
(or modifications of existing geodata)



clearinghouse v0.0.9 (Dishpan) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://gaia2.hiof.no:8080/index.php
Google
Search Web
Search Site

clearinghouse v0.0.9 (Dishpan) - running on atl

index

GeoData
- View pending submissions
- View my submissions
- Submit new geodata

Featurecatalog
- View current list
- Review current submissions
- Submit new featuretype

Settings
- Change password
- Personal profile

logout

Recent events

01 June 2003 13:37	knutejoh submitted
01 June 2003 13:32	henning submitted

News and system messa

27 May 2003 16:06	The featurecatalog the hiearchy. Feel f

Conflict ID: 91239
Comments: 12
Currently Active Group: knutejoh and henning
Users that have accepted the current solution: mats

Conflict ID: 91240
Comments: 5
Currently Active Group: knutejoh and mats
Users that have accepted the current solution: henning

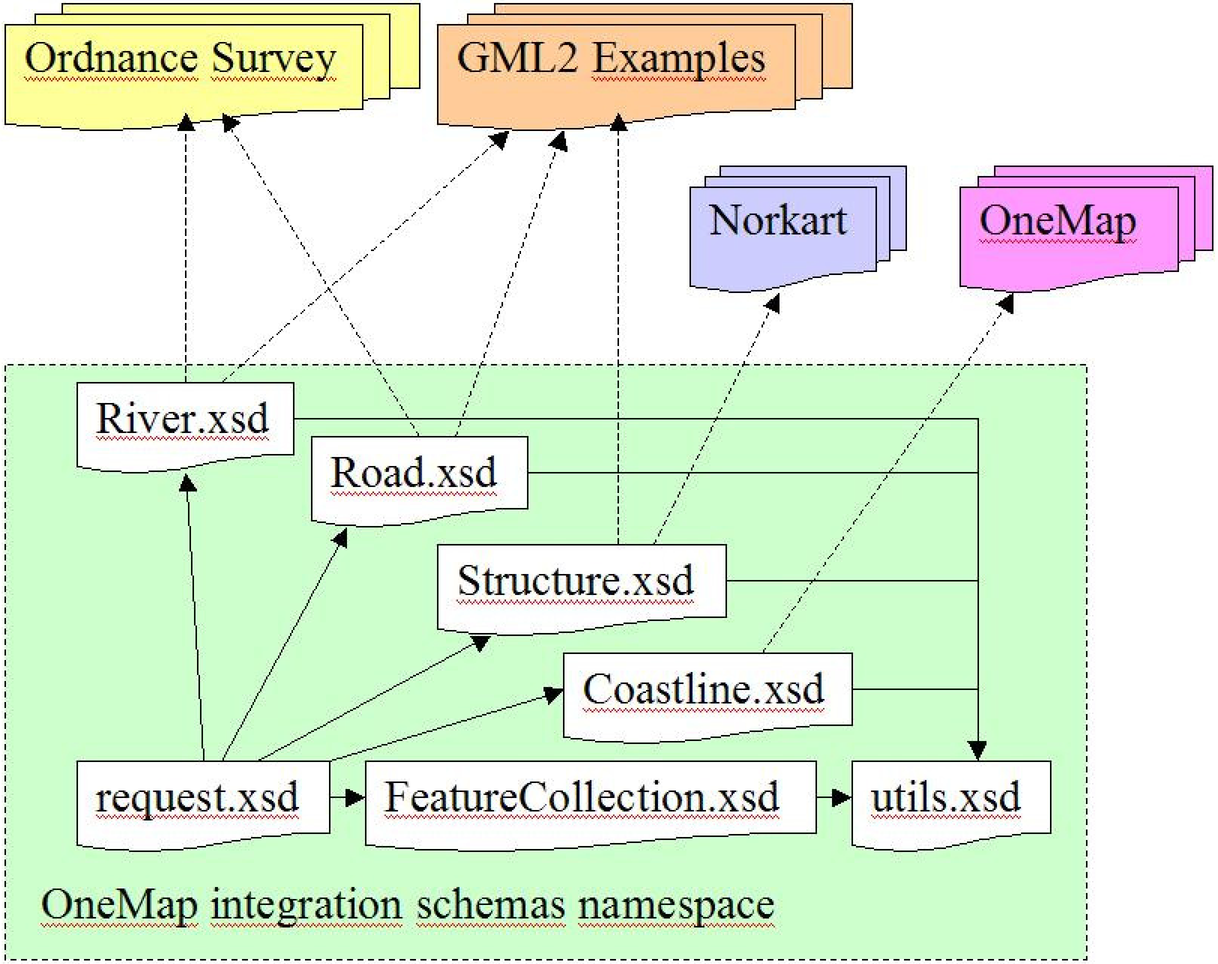
Quality assurance by peer review

Ref paper on SVG Open 2003

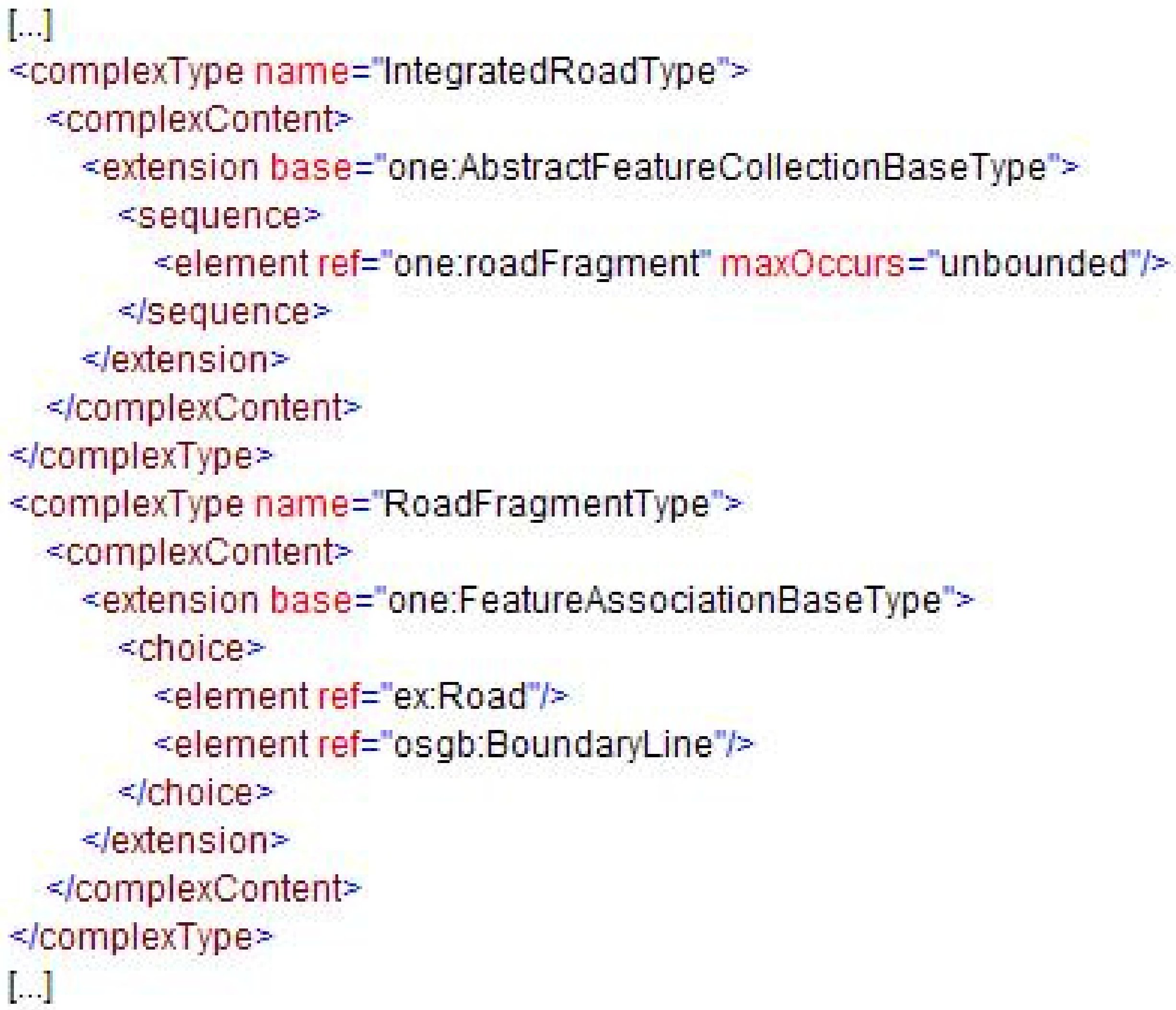
Lazy Integration

- An approach to integrate features as are into our repositories
- Dependent upon generic knowledge of the data
- Feature membership is provided through layer schemas
- All features in one layer describes the complete or part of a real world object.
- Scope of project: Semantic integration
- Important subject: Geographic integration

Lazy integration schema hierarchy



Integrating road fragments



```
[...]
<complexType name="IntegratedRoadType">
   <complexContent>
      <extension base="one:AbstractFeatureCollectionBaseType">
         <sequence>
            <element ref="one:roadFragment" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<complexType name="RoadFragmentType">
   <complexContent>
      <extension base="one:FeatureAssociationBaseType">
         <choice>
            <element ref="ex:Road"/>
            <element ref="osgb:BoundaryLine"/>
         </choice>
      </extension>
   </complexContent>
</complexType>
[...]
```

Generic GML Browser

- Demos
  - Visualization of “unknown” GML
  - Visualization of instance document, based on the lazy integration principles

Final Remarks

- GML can be accessed in a generic way, but there will always be trade-offs compared to propriatery viewers or applications built to utilized one application schema
- We are playing with the subject, not yet implementing
- Algorithm issues regarding schema parsing
- Redefines are not supported
- Local namespace declarations are not tested
- Integration of sources using *XLink* and *XPointer*
- Mixing of CRS/SRS systems
- Styling!
- Challenge..

Questions?

For more information:

www.onemap.org

hv@norkart.no
gunnar.misund@hiof.no