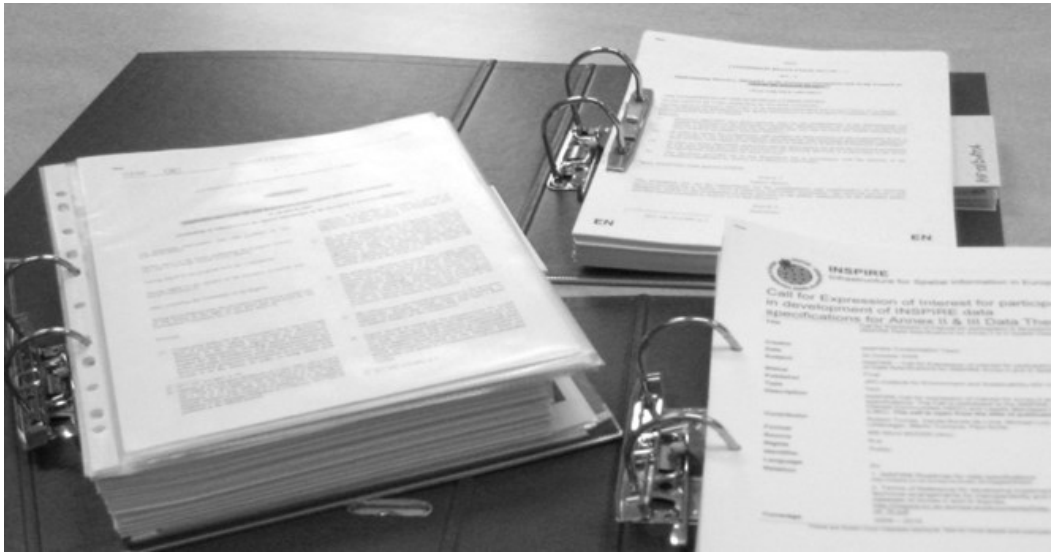
European Commission

Model-driven INSPIRE – blessing or curse?

Michael Lutz, Clemens Portele

Workshop on data modelling and model-driven implementation of data distribution, Copenhagen, 28-30 January 2015



www.jrc.ec.europa.eu

Serving society
Stimulating innovation
Supporting legislation

First of all…

Apologies for not making it to the workshop

And **thanks to Clemens** for presenting these slides

INSPIRE data specifications

- Concurrent development of data models by
  - 8 experts groups for the 9 Annex I themes (2008-2010)
  - 19 expert groups for the 25 Annex II+III themes (2010-2013)
  - 74 core application schemas (plus several extensions)
- Probably one of the biggest ever endeavours of developing conceptual data models and data schemas in the geospatial domain
- Adopted approach
  - UML data models
  - Common version-controlled model repository
  - Automatic generation of GML schemas (ShapeChange)
  - Common development roadmap

Model-driven approach – Blessings

- Consistent and well-structured approach across all 34 themes (incl. modelling of cross-theme relationships)
- Allowed detecting many inconsistencies and errors
- Easy generation of encodings and other artefacts (feature catalogue, mapping tables, …)
- The model repository was the "master database" for all data specifications and even the legal acts on data interoperability!

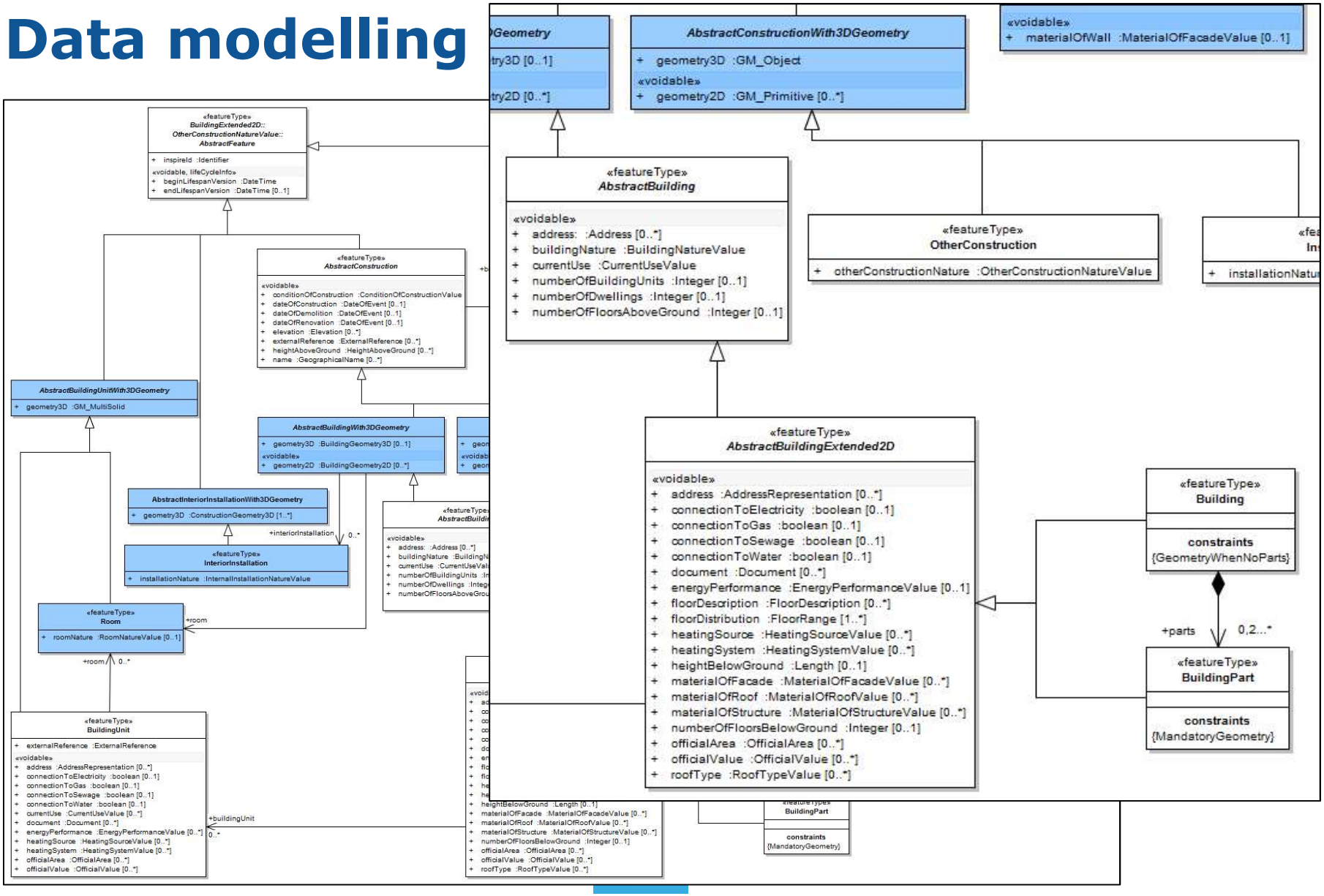
Model-driven approach – Curses

- Many GI experts don't know how to read & write UML
- UML-to-encoding rules understood by even fewer experts
- UML leaves many degrees of freedom for modelling → need for best practices & frequent exchange
- Missing support for managing dependencies between models
- Missing tools for describing model mappings

Data modelling

- Issues
  - Most GI experts are no UML modelling experts
  - UML leaves a lot of freedom → different modelling styles and approaches
  - In some cases overly complex models

Data modelling



Data modelling

- Issues
  - Most GI experts are no UML modelling experts
  - UML leaves a lot of freedom → different modelling styles and approaches
  - In some cases overly complex models
- How they were addressed in INSPIRE
  - Introductory training (UML & EA) to all TWG editors
  - Regular cross-theme meetings to discuss modelling issues
  - Several iterations, reviews and automatic checks of data models
  - JRC contact points helped TWGs to adopt common modelling approaches
- Areas for further work
  - Training material/courses & best practices
  - Communities of practice for data modellers
  - Tools for automatic consistency checking of data models

Code lists

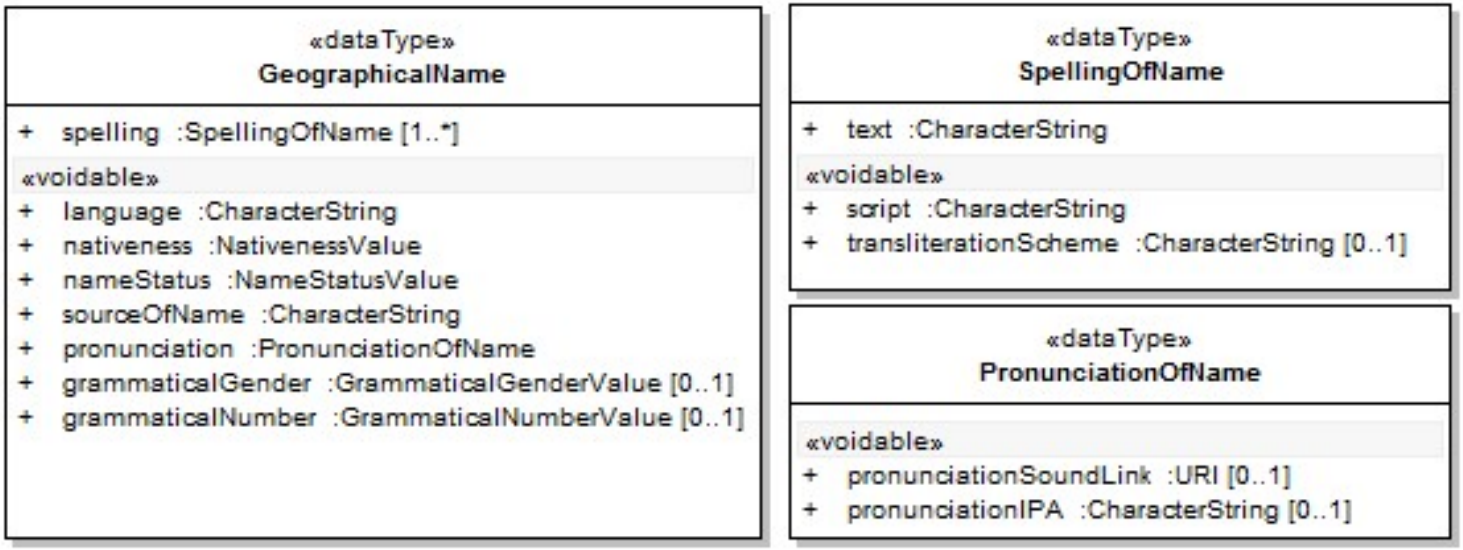
- Issues
  - Where to manage code list values during data model development?
  - If managed outside the UML model (e.g. in a register), how to connect the two?
- How they were addressed in INSPIRE
  - Initially, code list values were managed in the UML model
  - In Annex II+III, it was agreed to publish them in the INSPIRE registry (during development, Excel tables were used)
  - The link was established through a «CodeList» class and a tagged value with the code list URI
- Areas for further work
  - Common guidelines and tools for managing code lists, also during data model development

Presenting data models

- Issues
  - Many GI experts have difficulties with understanding UML class diagrams
  - Data providers have difficulties identifying relevant application schemas and feature types to map their INSPIRE-relevant data to
- How they were addressed in INSPIRE
  - Different representations (e.g. feature catalogues, searchable feature concept dictionary, legal acts) automatically created from UML models
- Areas for further work
  - Developing other, more intuitive representations to explore data models
  - Training for readers/consumers of UML data models ?

Directly generating encodings from UML

- Issue
  - Conceptual models may not always be directly suitable for generating data encodings, e.g. when
    - advanced modelling constructs (e.g. association classes) are used
    - Complex data types are used, e.g. GeographicalName



Directly generating encodings from UML

- Issue
  - Conceptual models may not always be directly suitable for generating data encodings, e.g. when
    - advanced modelling constructs (e.g. association classes) are used
    - Complex data types are used, e.g. GeographicalName
- How they were addressed in INSPIRE
  - Custom encoding rules for specific modelling constructs
  - Complex data types were usually mapped 1:1
- Areas for further work
  - Guidelines for deriving implementation models from conceptual models
  - Methods for simplifying complex data types when generating the encoding

Managing dependencies & versioning

- Issues
  - Many dependencies between data models for 34 INSPIRE themes
  - Dependencies also reflected as imports in XML schemas
  - When updating individual application schemas and XML schemas, this creates a cascade of required updates in dependent schemas
- How they were addressed in INSPIRE
  - Version-controlled data model repository
  - Distinct releases of data models and schemas
  - Proposal to maintain the XML schemas for deprecated models for a certain period of time
- Areas for further work
  - Methods and tools for identifying and managing dependencies of data models and encodings

Mapping between data models

- Issues:
  - Model mappings may be required between
    - Conceptual and implementation models
    - Exchange data models and existing standards (e.g. GeoSciML, CityGML)
    - Exchange data models and data models of source data sets
  - Mappings are often expressed using mapping tables
    - Not easy to use
    - Not easy to maintain if one of the model changes

Mapping between data models

- Example: the mapping between the INSPIRE Addresses and the ISO 19160-1 data model required 39 mapping tables

Table B.4.3 – Core: Mapping Address (INSPIRE Addresses model) → Address (INSPIRE profile)

INSPIRE Addresses model attribute and data type		INSPIRE profile attribute	Mapping description
inspireId	Identifier	inspireId	
position		position	Refer to table B.4.19 (GeographicPosition → AddressPosition) for the mapping.
locator		addressComponent	Only map address components that are of type AddressLocator
alternativeIdentifier	CharacterString	id	Map CharacterString to Oid (from 19152) data type.
component		addressComponent	
parentAddress		parentAddress	

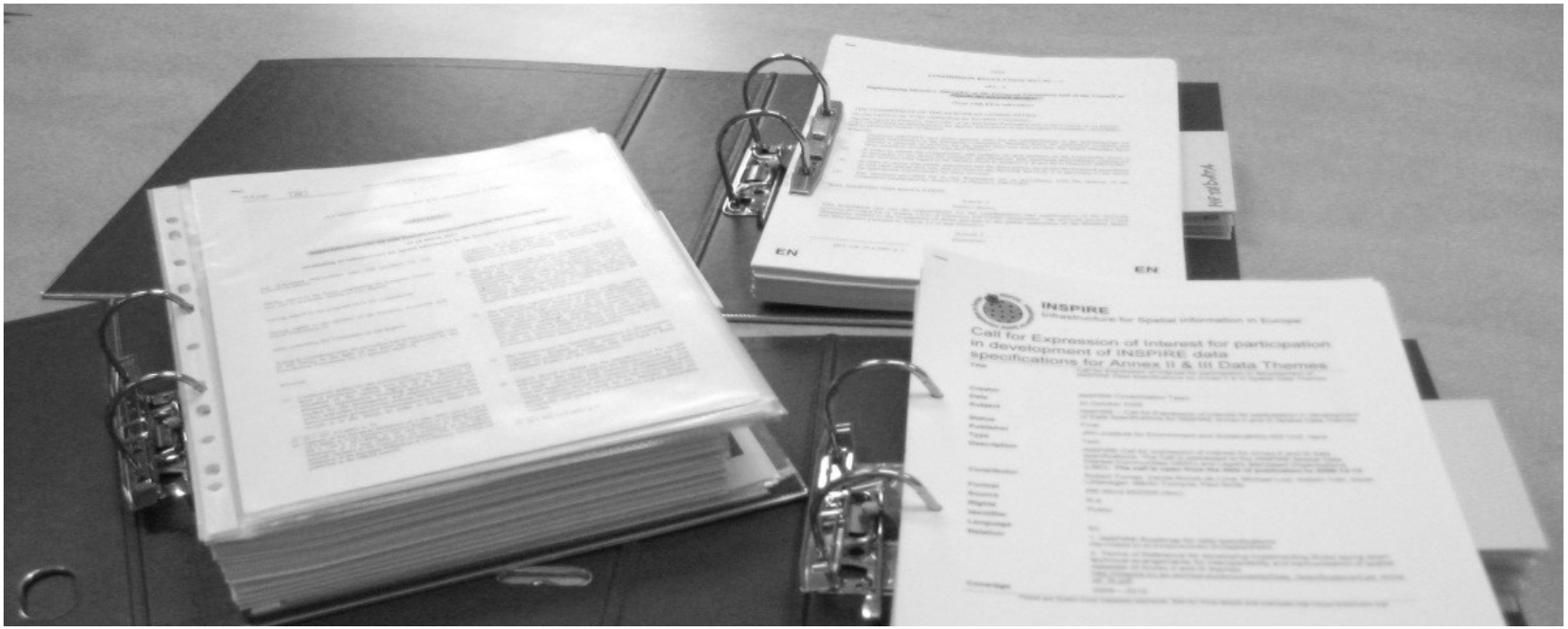
Mapping between data models

- Issues
  - Model mappings may be required between
    - Conceptual and implementation models
    - Exchange data models and existing standards (e.g. GeoSciML, CityGML)
    - Exchange data models and data models of source data sets
  - Mappings are often expressed using mapping tables
    - Not easy to use
    - Not easy to maintain if one of the model changes
- How they were addressed in INSPIRE
  - Mapping tables automatically generated to support mapping from source to INSPIRE data models
- Areas for future work
  - Development of other representations of mappings that can be linked / kept in sync with the data models

Conclusions

- The model-driven approach clearly was the key for successfully delivering the INSPIRE data specifications and data interoperability Regulation
- But it requires thorough understanding (→ training) and good tool and methodological support
- Work is still needed on
  - General training & guidelines
  - Model mappings
  - Management of dependencies and versioning
  - Guidelines and tools for generating (simple) encodings

Have a good workshop!



The INSPIRE Regulation on data interoperability – probably the first “model-driven legal act”