Title: Models and schemas in HUMBOLDT

Title: Models and schemas in HUMBOLDT

Author(s)/Organisation(s):

Marian de Vries (TUD), Astrid Fichtinger (TUM)

Working Group:

WP7, WP5

References:

A7.1D1 Concept of application-specific harmonised data models

WP7 Glossary

Workshop results Task Force "schema translation"

Short Description:

This document clarifies terms related to „conceptual schema specification and mapping“. It contains a matrix where data models are categorized using 2 dimensions: from abstract to implementation, and from generic to application-specific.

Keywords:

History:

Version	Author(s)	Status	Comment
000	Marian de Vries	new / rfc	
001	Astrid Fichtinger		Review, added ISO and mdWFS definitions
002	Marian de Vries	update / rfc	Added intro and further explanation, removed WP5 internal text

Table of contents

1 Introduction

1.1 Purpose and scope

During the specification work in WP5 the need was felt to have a better understanding of different terms used in the context of (conceptual) schema translation. Especially the terms ‚application schema' and ‚conceptual schema' needed more clarification.

Therefore in this document an attempt is made to give an overview of terms that contain the words ‚model' and ‚schema', and give examples in addition to the textual definitions already given in other HUMBOLDT documents, see for instance deliverable A7.1D1 Concept of application-specific harmonised data models, and the "work document" called "WP7 Glossary".

The definitions in this document are related to data models and the role they play in HUMBOLDT, not to software engineering, or to models of e.g. climate change, weather or other forecasts, or simulation.

This is therefore only about the terms 'model' and 'schema' in the context of data modelling in HUMBOLDT.

2 Model, schema

Model (in general): an abstraction of a collection of things or of one thing, not the real-world (real or virtual) objects themselves.

Data model: a model of the (geographic) data that is stored and/or exchanged.

Schema: a schema is a 'kind-of' model. Some use the words as synonyms, for others there is a distinction.

In the ISO 19100 series, the two terms are distinguished as follows [ISO 19101]:

- conceptual model: "model that defines concepts of a universe of discourse"
- conceptual schema: "formal description of a conceptual model". A conceptual schema is described in a conceptual schema language (in case of the ISO 19100 series in the respective UML profile described in ISO 19103).

The ISO definition of conceptual schema is very broad. Under this definition also an ontology is a conceptual schema.

Looking not at ISO, but at data modelling and data management practice: the word 'schema' is chosen often in the context of data management and model implementation (by database administrators for example), the word 'model' is used more by system designers and data modellers in the context of design and development. This different roles and subtle distinctions between 'schema' and 'model' can also be seen in Table 1 below, where the following pattern becomes apparent: a schema is lower at the abstraction scale than the model with the same name: what is called a 'conceptual schema' in data management, is called a 'logical data model' by data modellers.

3 Levels and scope of models: a matrix

For the discussion in WP5 especially the following terms needed clarification: What is a 'conceptual schema' in HUMBOLDT? What is the relation with the term 'application schema' in HUMBOLDT?

When we compare the two terms we actually compare two dimensions of a data model: abstraction level (or: platform independent vs. platform dependent), and scope (generic vs. application-specific). The matrix in Table 1 is meant to clarify this also with examples.

With **scope** of a data model is meant: from generic to application-specific data models (from left to right in the table).

With **abstraction level** is meant: from conceptual view on the information content to implementation-specific scripts and schemas (from top to bottom in the table). The abstraction level of data models has to do with:

- the data model being implementation-neutral (in software engineering aka Platform Independent Model) or implementation-specific (aka Platform Specific Model). In database modeling and implementation the 3 levels 'conceptual, logical and physical' is more common.
- phases in data modeling: from first sketch of the information content and how it can be structured, to a final model where all question marks are decided on.

In the cells there are examples of data modeling artifacts that belong on that spot in the matrix.

See Table 1, starting on next page.

Note about the relation with ISO definitions:

The term "application schema" is used widely in the ISO/OGC and INSPIRE world. For that reason we also use it in HUMBOLDT. The term will be used as synonym for HUMBOLDT's 'application-specific data model' (see deliverable 7.1D1). Application schema or application-specific data model is then: the data model for a specific application, such as a HUMBOLDT Scenario.

The term 'conceptual schema' is used in ISO/OGC standards, and also in GIS practice and discussions, to express the contrast with the physical data format (the actual encoding/storage format of the data). A conceptual schema gives (logical/conceptual) information about the content and structure of the data, not how it is actually (physically) stored.

According to ISO, an application schema is a "conceptual schema for data required by one or more applications" [ISO 19101]. The last part about 'required by one or more applications' accords to the HUMBOLDT definition. But the 'conceptual schema' part of the definition can lead to misunderstanding. In practice in ISO/OGC and INSPIRE the term 'application schema' is used at two levels: at the conceptual schema level (in INSPIRE as UML model), and also at the logical/physical schema level, as for example in 'GML application schema'.

Table 1: Matrix of data modeling artifacts relevant to HUMBOLDT

(PIM = Platform Independent Model, PSM = Platform Specific Model)

	Abstraction level: Conceptual -> implementation	Explanation	Scope: Generic -> application-specific		
			information field (in our case geographic information)	application domain (for example: hydrography, cadastre, …)	application (= HUMBOLDT Scenario)
PIM	Ontology (of concepts)	Can take different forms: from thesaurus in plain English (loose structure), to ontology in ontology language (processable by software)	The INSPIRE Feature Concept Dictionary can evolve into a cross-application ontology for the geographic information field.	For example the thesaurus about Marine terms and knowledge maintained by NERC/BODC.	Codelists and classifications can be published on a Scenario portal in the form of an ontology.
PIM	Conceptual data model = The 'what', and first 'how': what is the relevant information for us (in our information field, application domain or application), and how can it be divided into classes, attributes,	How humans see the information that is involved in their use cases. Does not have to be specified in detail yet (this is the second aspect: phases in data modeling).	ISO 19109 (General Feature Model), ISO 19123 (Coverages): text and UML models.	Classification schemes (or simpler: standardized enumeration lists) for specific application domains: NUTS, CORINE, Habitat classifications, Risk levels, …	HUMBOLDT Scenario-specific conceptual data model, details can be left out, implementation platform does not have to be taken into account. Purpose: this model 'freezes'

	and relations between classes.				what information will be used as input and/or published as output in the Scenario.
PIM/ PSM	Logical data model, or: conceptual schema = Describes and prescribes the logical data structure and constraints on attribute values using a conceptual schema language = The 'what', and partly the 'how'. = Classes, attributes, associations, operations, enumerations, data types, cardinality / multiplicity.	Precise enough to be 'unambiguously understood' by software ('deterministic'?) Takes implementation constraints into account, for example yes/no multiple inheritance, composite attributes or repeating groups. These constraints have to do with: meta-model of the data storage solution ('plain old' relational vs. object-relational vs. m-to-n network, …), and with the capabilities of the DDL and/or the programming language (Java, C++)	UML models of ISO 19107 (spatial schema), ISO 19108 (temporal schema).	The INSPIRE Annex Theme data models, in two representations: UML data model, called 'UML application schema' Feature Catalogue	HUMBOLDT Scenario-specific application schema in UML (to discuss: also as ISO 19110 Feature Catalogue ?). This must be precise enough (for example with unambiguous data types) so that software can process it. Purpose: this UML model will be the target model for data transformation (the restructuring and reclassifying aspect of data transformation).
PSM	Physical data model A. Logical schema:	Can be derived by software from the logical data model.	For example the XML Schema files (*.xsd) that express the GML	In case of GML encoding: the	Same terms as for application-domain.

	description of data structure and constraints in a Data Definition Language (DDL) such as XML Schema, or SQL 'Create table etc' constructs Which DDL to use depends on the data storage format that is chosen (XML, ascii, database category) = The 'how' for a specific encoding format.	Needed for that: rules/mappings for vertical schema translation from the higher level to this level.	specification. Or the 'create spatial data types and metadata tables' scripts for PostGIS.	'GML application schema' (official OGC term), often abbreviated to 'GML schema'. Examples: SensorML.xsd CityGML.xsd NetCDF.xsd	Examples: our Scenario xsd's in case of GML, inline headers in case of NetCDF files, file-specific metadata in case of image/raster files, sql create table scripts in case of Oracle or PostGIS.
PSM	B. Physical schema = The 'how' for a concrete implementation.	The above (physical data model), plus indexes, table space allocation, MIME-type settings, etc. This depends on the chosen encoding format, and on 'local' conditions.			

4 Comparison with 'schema translation' architecture

In this Section there is a short comparison between the HUMBOLDT definitions and examples in the matrix table, and the following figure, which is also in the 'conceptual schema translation task force' document at https://intranet.esdi-humboldt.eu/documents/details/835/1

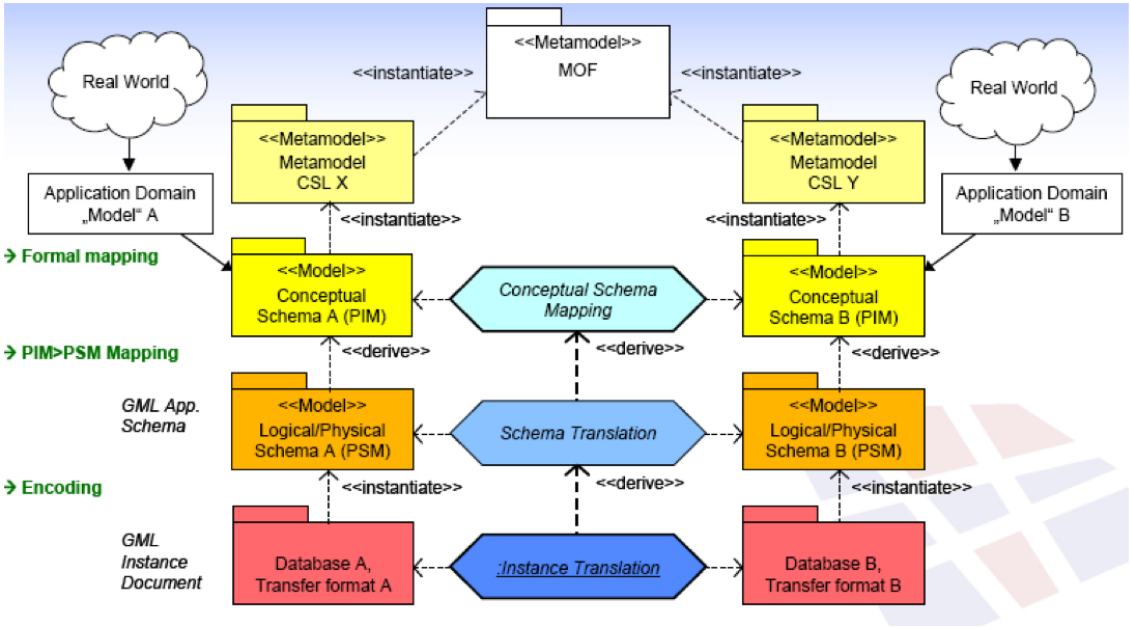

Figure 1: Illustration of OMG's model-driven architecture and approach on data modelling and data harmonisation (courtesy of Peter Staub, ETHZ)

Differences between the matrix table and this figure are:

- the figure shows the meta-model of the conceptual schema language, the matrix table does not. For the figure the relation with the meta-model is important because this shows how 2 schemas with a different meta-model can still be mapped (because they are both instances of the MOF meta-meta-model).
- the matrix table has 'conceptual data model', and the figure of Staub has something that could be comparable: 'application domain model A or B', with model between quotes. This 'application domain model' is a description of the relevant universe of discourse in natural language. In the HUMBOLDT case this does not have to be 'natural language', but can also be e.g. a conceptual model in UML or an ontology.

Overall however the definitions in the matrix of HUMBOLDT terms are in harmony with the levels and terms in the figure.