



LABORATÓRIO NACIONAL  
DE ENGENHARIA CIVIL

DEPARTAMENTO DE HIDRÁULICA E AMBIENTE  
Núcleo de Tecnologias de Informação em  
Hidráulica e Ambiente

Proc. 0602/11/16281

# THE WATER FRAMEWORK DIRECTIVE GEOSPATIAL DATA MODEL

## Article 3 Dataset

Developed at Joint Research Centre, Institute for  
Environment and Sustainability (Rural, Water and  
Ecosystem Resources Unit)

Lisboa • Outubro de 2007

**I&D** HIDRÁULICA E AMBIENTE

RELATÓRIO 314/2007 – NTI



## ACKNOWLEDGMENTS

---

I would like to express my gratitude to my supervisor, Dr. Alfred de Jager, whose expertise, understanding, and patience, added considerably to my training experience. I appreciate his vast knowledge and skills in many GIS related areas.

I must also acknowledge Erika Rimaviciute for her wise suggestions and interest in helping me with several issues related with this work.

A special thanks goes also to my thesis supervisors, Dra. Maria Alzira Santos and Prof. João Catalão Fernandes, that provided me with directions, and scientific support.

I would also like to thank my family for the support they provided me through this months, in particular to my companion Sofia Costa Lopes.

Finally, I recognize that this research would not have been possible without the financial, technical, and scientific assistance of FCT (Fundação para a Ciência e a Tecnologia), the LNEC (National Civil Engineering Laboratory), and the Joint Research Centre (IES). To these agencies I would like to thanks this opportunity.



# SUMÁRIO

---

O principal objectivo do trabalho de estágio foi o desenvolvimento e implementação de uma estrutura de base de dados geográficos centralizada, especialmente contínua para armazenar, gerir, editar, e distribuir os dados geográficos referentes ao Artigo 3.º da Directiva-Quadro da Água (DQA). O desenvolvimento do catálogo de objectos geográficos (*Feature Catalogue*) baseou-se na norma ISO 19103<sup>1</sup>, sendo que o modelo conceptual de dados (MCD) se baseou na norma ISO 19110<sup>2</sup> e nas especificações OGC-GML<sup>3</sup>.

O MCD foi desenvolvido recorrendo a uma ferramenta CASE usada para declarar os diagramas UML que descrevem os elementos da base de dados (geográficos e não geográficos). O MCD foi posteriormente traduzido num modelo lógico de dados (MLD), adaptando-o às regras de uma *ESRI Geodatabase*.

Recorreu-se a *Tag Values*, para especificar os parâmetros das classes UML, e à especificação de domínios para facilitar a validação dos dados armazenados. Finalmente, o MLD foi convertido num modelo físico de dados (MFD) que resultou na implementação de uma estrutura de dados sobre vários sistemas de gestão de base de dados. Foram utilizados três sistemas de base de dados distintos: MS Access, *file-based geodatabase* da ESRI, e Oracle Spatial.

O documento está estruturado em nove secções com o intuito de descrever em detalhe o conhecimento teórico e tecnológico que permite desenvolver um modelo de dados geográficos e implementá-lo sobre um sistema de gestão de base de dados.

O documento inicia-se com uma introdução onde é descrito o contexto técnico, científico e pedagógico em que se insere este trabalho. A segunda, terceira, quarta e quinta secções sumarizam os aspectos teóricos e técnicos em que se baseia o desenvolvimento de um modelo de dados geográficos. A sexta secção descreve o processo de desenvolvimento do modelo de dados tendo em consideração todos os aspectos técnicos e tecnológicos que devem ser tidos em conta para construir e implementar o modelo de dados geográficos. A sétima secção é dedicada aos SGBD em que a implementação do modelo de dados foi testada. A oitava secção é dedicada às diferentes possibilidades para carregamento de dados. Após estas secções são feitos alguns comentários finais e descritos os futuros desenvolvimentos do trabalho.

O relatório termina com 14 anexos e contém um CD-Rom com todos os ficheiros criados no âmbito deste trabalho.

---

<sup>1</sup> ISO/TS 19103 – Conceptual schema language

<sup>2</sup> ISO/TS 19110 – Methodology for feature cataloguing

<sup>3</sup> OpenGIS® Geography Markup Language (GML) Implementation Specification (OGC 03-105r1)



## ABSTRACT

---

The major goals of this traineeship was the development and implementation of a centralized, spatially continuous geospatial database structure to store, manage, edit and distribute the geographic reference data of Article 3 of the Water Framework Directive (WFD-Art3). Based on the ISO 19103<sup>4</sup>, ISO19110<sup>5</sup>, and OGC-GML<sup>6</sup> specifications, a Feature Catalogue and a Conceptual Data Model (CDM) were developed to describe all object classes, attributes and relationships that were presented in previously established database.

The CDM was created with a CASE-tool, where UML class diagrams were developed to describe all features (geographic and non-geographic). Next, the CDM was translated to an Esri's geodatabase model, specifying the UML classes according with this specification. Tagged values were used to define class parameters, and domains of values were established to help data validation. Finally, the resulting Logical Data Model (LDM) was used to create the physical model of the geospatial database. For testing the implementation of the physical data model, a MS Access database, an ESRI file geodatabase, and an Oracle Spatial database, were used.

The document is structured into nine sections, with the aim to explain in detail both the knowledge and technology behind a geospatial data model design and implementation. It starts with an introduction describing the context on which this work was developed. The second, third, fourth, and fifth sections summarize the theoretic and technological aspects that support the geospatial data modelling process. The sixth section is one of the most important and describes the geospatial database construction process regarding technical and technological aspects that should be taking into account to construct and implement a geospatial data model. The seventh section is dedicated to database platforms and database management systems where the implementation of the geospatial database structure was tested. The eighth section is dedicated to the data loading tests made in several database platforms. After these sections some final remarks and future developments are presented and described, as well as the bibliography. The document is also complemented with fourteen annexes and a CD-Rom containing all files used and produced in the context of this work.

---

4 ISO/TS 19103 – Conceptual schema language

5 ISO/TS 19110 – Methodology for feature cataloguing

6 OpenGIS® Geography Markup Language (GML) Implementation Specification (OGC 03-105r1)





# ÍNDICE

1	INTRODUCTION .....	1
1.1	Framework .....	1
1.2	The Article 3 of the Water Framework Directive .....	2
1.3	The GIS Guidance Document .....	3
2	DATA MODELLING.....	5
2.1	Data modelling concepts .....	5
2.2	Data modelling steps.....	6
3	UNIFIED MODELING LANGUAGE (UML) .....	9
3.1	UML Concepts.....	9
4	GEOGRAPHIC MARKUP LANGUAGE (GML) .....	13
4.1	GML Concepts .....	13
5	METHODOLOGY .....	16
5.1	Framework .....	16
5.2	CASE Tools.....	16
5.3	MS Visio .....	17
5.4	Rational Rose .....	18
5.5	UML packages and static structure diagrams .....	18
5.6	Geodatabase diagrammer.....	20
6	GEOSPATIAL DATABASE CONSTRUCTION PROCESS.....	23
6.1	Framework .....	23
6.2	The geospatial database modelling.....	24
6.3	The XML Metadata Interchange file (XMI) .....	28
	6.3.1 Overview .....	28
	6.3.2 The creation of XMI.....	30
6.4	Physical data model implementation.....	33
7	DATABASE MANAGEMENT SYSTEMS.....	38
7.1	Overview.....	38
7.2	Implementation scenarios .....	40
	7.2.1 Scenario 1: MS Access (personal geodatabase).....	40
	7.2.2 Scenario 2: File-based geodatabase .....	41
	7.2.3 Scenario 3: Oracle .....	41
	7.2.4 The ArcSDE Compressed Binary Storage.....	42
8	DATA LOADING .....	45
8.1	Overview.....	45

8.2	Loading data with Simple Data Loader (ArcCatalog) .....	45
8.3	Loading data with Object Loader (ArcMap) .....	51
8.4	Loading data with FME .....	52
9	FINAL REMARKS AND DISCUSSION OF RESULTS .....	55
10	BIBLIOGRAPHY .....	57
	ANNEXES .....	61
	ANNEX A – Article 3 of the WFD .....	63
	ANNEX B – XML Interchangeable file (XMI) .....	67
	ANNEX C – Geodatabase Settings .....	71
	ANNEX D – Geodatabase Diagrammer .....	85
	ANNEX E – UML Semantics Checker in ArcCatalog .....	89
	ANNEX F – UML Tools .....	93
	ANNEX G – Feature Catalogue .....	99
	ANNEX H – Geodatabase elements .....	109
	ANNEX I – Data Types .....	113
	ANNEX J – Geodatabase UML Static Structure Objects .....	117
	ANNEX L – Tagged Values .....	121
	ANNEX M – CD-ROM Contents .....	129
	ANNEX N – Terms, definitions and abbreviations .....	133
	ANNEX O– Traineeship work plan .....	141

## TABLE INDEX

Table 1	ArcGIS geodatabase types .....	38
Table 2	Geodatabase creation parameters .....	40
Table 3	Feature geometry storage types in Oracle database.....	41
Table 4	Data type relation between Oracle Spatial and Shapefiles.....	49
Table 5	Tags to geodatabase UML classes .....	123

## FIGURE INDEX

Figure 1	The four steps of data modelling .....	6
Figure 2	GML version 3.1.1 schema definition. (Source: <a href="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd">http://schemas.opengis.net/gml/3.1.1/base/gml.xsd</a> ) .....	8
Figure 3	UML notation used in the class diagrams.....	11
Figure 4	Package diagram of GML 3.1.1. (Source: <a href="http://www.opengeospatial.org/standards/gml">www.opengeospatial.org/standards/gml</a> ) .....	14
Figure 5	ArcInfo UML model objects .....	17
Figure 6	ArcInfo UML model template load .....	18
Figure 7	Packages hierarchy implemented in Visio.....	19
Figure 8	Basic UML diagram to generate tables and feature classes .....	20
Figure 9	Activation of <i>geodatabase diagrammer</i> from ArcCatalog .....	21
Figure 10	Geodatabase modelling and implementation schema (Source: Esri) .....	24
Figure 11	Part of the conceptual model for Article 3 objects. It presents the feature classes and the relationships amongst them .....	25
Figure 12	Esri geodatabase: super-class inheritance diagram .....	26
Figure 13	Logical data model: class inheritance static structure diagram .....	27
Figure 14	UML package properties stereotyped <i>FeatureDataset</i> .....	28
Figure 15	XMI Export tool in MS Visio .....	29
Figure 16	Succeeded export of XMI file.....	29
Figure 17	Running of the Semantic checker to evaluate the UML model .....	30
Figure 18	Semantic checker dialog box .....	31
Figure 19	Semantics checker report.....	32
Figure 20	Errors check result.....	32
Figure 21	Schema wizard to implement the physical model.....	33
Figure 22	Creation of an empty geospatial database .....	34

Figure 23	The use of the XMI to generate the database structure .....	34
Figure 24	Schema wizard used to import the XMI files .....	35
Figure 25	Schema wizard parameters definition .....	36
Figure 26	Geodatabase spatial reference system parameters.....	37
Figure 27	Database connection properties.....	40
Figure 28	Data types of feature classe Basins .....	46
Figure 29	Logical model design of feature class Basins and range domains.....	47
Figure 30	Field data types applied to feature class Basins.....	48
Figure 31	Simple data loader wizard .....	48
Figure 32	Creation of shapefiles through FME .....	49
Figure 33	Field data types of shapefiles .....	50
Figure 34	<i>Simple Data Loader</i> matrix between shapefile and personal geodatabase .....	50
Figure 35	Object Loader validation interface .....	51
Figure 36	Validate Features command.....	52
Figure 37	Dialog boxes resulting from the application of the <i>Validate Features</i> command.....	52
Figure 38	FME Workbench.....	53
Figure 39	Data transformation using FME.....	53
Figure 40	FME conversion from Oracle Spatial to several ArcGIS data formats .....	54
Figure 41	Range Domain definition .....	124
Figure 42	The edit of an attribute table with coded values .....	125
Figure 43	Definition of a coded values domain.....	125
Figure 44	Data type definition with a domain.....	125
Figure 45	Samples of coded values and range domains .....	126
Figure 46	Domains system tables in Oracle DBMS (Source: Esri) .....	127
Figure 47	UML Class properties of an abstract class.....	127
Figure 48	The use of <i>IsAbstract Class</i> .....	128

## 1.1 Framework

The work here described resulted from a traineeship held in the Joint Research Centre (JRC), Institute for Environment and Sustainability, at the Rural, Water and Ecosystems Resources Unit. The training work was supervised by Alfred de Jager according with a work plan previously established. The work plan is included in Annex O.

This document describes the modelling steps taken to design the conceptual, logical, and physical model of a geospatial data structure to support the geographic information produced by the European Member States according with the Water Framework Directive (WFD) article 3, following the specifications published in the GIS Guidance document. The chosen conceptual schema language was the Unified Modelling Language (UML) (ISO 19103).

The WFD geospatial data model was developed and described using the Object-Oriented approach expressed in UML class diagrams. The data model was tested and implemented using Esri's technology. The geospatial database generated on the basis of the conceptual and logical models is represented as a relational database, which contains spatial and non-spatial elements, defined as:

- Object classes;
- Feature classes;
- Feature datasets;
- Networks;
- Tables;
- Relationship classes;
- Domains.

Classes are mapped as feature classes or tables. For organisational and logical purposes, feature classes are organized into feature datasets, if they concern the same area or type of information. Feature classes and tables can be related to each other in relationship classes. Each feature class or its attributes can be defined in detail, by adding special behaviour or validation rules. These could be pre-defined ranges of values or lists of permissible values for an attribute (coded values). These constraints prevent collected data and information from errors and they assure their accuracy and consistency on the level of encoding through the GIS environment.

The Annex M of this document contains the detailed explanation of all the files produced in the scope of this work, and also the relevant bibliography that supports the knowledge applicable.

This work is part of a doctorate thesis that aims to assist the implementation, in Portugal, of the WFD geospatial database, concerning mainly the support of the geographic information to be produced within the River Basin Management Plans. The author hopes that this work could contribute to a better

understanding and representation; not only of the water resources itself, but also of the ecosystems directly related.

The WFD, and the upcoming second generation of river basin management plans, would benefit from a well structured geospatial database that assist in sharing of information between the actors and stakeholders, involved in the water management, at the several levels of administration.

A well structured GIS starts with an appropriate data model design that could support and guarantee the data integrity, and at the same time, being flexible and universal enough to be adapted to specific realities.

Nowadays, European GIS market shows a great complexity and heterogeneity between how different Member States deal with information about water resources. This fact results in a great challenge when trying to establish a unique platform to support the data produced in the scope of the Water Framework Directive.

The thesis will focus its activity in the design and implementation of a database structure capable to deal with the present river basin management plans needs; considering a common understandable platform, respecting all the actual rules and standards available to store and exchange geographic information. The geographical focus of the thesis will be the Iberian territory and the international hydrographical region of Guadiana River.

The work here documented intends to be a contribution to the understanding of the geospatial database structure, as well as a way to facilitate the implementation of this structure over several common database management systems.

## **1.2 The Article 3 of the Water Framework Directive**

The Article 3 of the WFD concerns the “Coordination of administrative arrangements within river basin districts”. It refers that the Member States shall identify:

- individual river basins;
- river basin districts;
- international river basins;
- main rivers;
- coastal waters;
- transitional waters;
- groundwater;
- competent authorities.

The data model design supports all the above referenced datasets and creates a common base to support the geographic data produced also for article 5 and article 8 of the WFD. The Article 3 of the Water Framework Directive is transcribed in Annex A of this report.

### 1.3 The GIS Guidance Document

The *GIS Guidance document on implementing the GIS Elements of the WFD* is a document to guide GIS technicians, consultants, and stakeholders in the implementation of WFD, establishing a framework for community action in the field of water policy regarding the geographic information to be produced. It also focuses on the implementation of the GIS elements in the broader context considering the integrated river basin management plans as required by the Directive (Vögt, 2002).

The geospatial data model is inspired in the definitions given in the GIS Guidance Document about the geographic feature classes to be stored in the scope of the WFD. Because WFD is not explicit in defining how water bodies should be represented in a GIS, the GIS Guidance Document gives additional definitions and specifications to make these representations.

The main geographic feature classes presented in the data model (Article 3 of the WFD) have the following definitions in the GIS Guidance Document:

- **River Basin Districts:** are “*collections of river basins, transitional waters and coastal waters*”. Thus, despite duplication of some geometry, they are defined as a separate polygon feature class;
- **River Basin:** means “*the area of land from which all surface run-off flows through a sequence of streams, rivers and, possibly, lakes into the sea at a single river mouth, estuary or delta.*”, river basins shall be assigned “*to individual river basin districts*”;
- **River water body:** means a “*body of inland water flowing for the most part on the surface of the land but which may flow underground for part of its course*”. A River Water Body may consist of several component river segments;
- **Lake water Body:** means a “*body of standing inland surface water*”. Lakes are termed as *Lake Water Body* to allow the subdivision of individual lakes into distinct bodies;
- **Coastal waters:** means a “*surface water on the landward side of a line, every point of which is at a distance of one nautical mile on the seaward side from the nearest point of the baseline from which the breadth of territorial waters is measured, extending where appropriate up to the outer limit of transitional waters*”.
- **Transitional waters:** are “*bodies of surface water in the vicinity of river mouths which are partly saline in character as a result of their proximity to coastal waters but which are substantially influenced by freshwater flows*”.
- **Groundwater body:** the WFD does not provide standard criteria for the characterization of groundwater bodies. It was asked to the Member States to provide information on pressures, overlying strata and dependent surface water and terrestrial ecosystems. For groundwater bodies considered to be at risk, further detail on these geological and hydrogeological

characteristics could have been also provided. Information concerning the impact of human activity may also be collected. The model does not deal with these parameters, but this might be an area that merits increased standardization of the information gathered. Discussion continues on how such bodies should be delineated, and subsequently represented. For the purposes of the present model, it is assumed that groundwater bodies will be polygon features. Unlike surface water bodies, the delineated boundaries of groundwater will rarely coincide exactly with river basins. The Directive requires that all the groundwater bodies should be assigned to a River Basin District. This relation is established in the data model;

- **Competent Authority:** means “*an authority or authorities identified under Article 3(2) or 3(3)*”. Because in some cases it is not possible to aggregate river basin districts to form the boundary of the competent authority, they are defined as a separate point feature class.

This features classes were all modelled using UML diagrams to produce, first the conceptual data model, then the logical data model. The following sections explain the theory and the methodology applied.



### 2.1 Data modelling concepts

For many years, engineers, scientists and other professionals who build complex structures or systems have been creating models of what they intended to build. Sometimes the models are physical, such as scaled mock-ups of airplanes; sometimes the models are less tangible, as seen market trading simulations and electrical circuit diagrams. In all cases, a model serves as an abstraction -- an approximate representation of the real item that is being built.

Why should we model something before build it? Simple things do not necessarily need a model preceding its construction - such as a simple checkbook register, or a simple macro in a word processor. Such projects share all or most of the following characteristics:

- The problem domain is well known;
- The solution is relatively easy to construct;
- Very few people need to collaborate to build or use the solution (often only one);
- The solution requires minimal ongoing maintenance;
- The scope of future needs is unlikely to grow substantially.

Why do some professional disciplines bother to create models? Why do they not just build the real thing right away? The answer has to do with the complexity, the risk and the fact that original practitioners are not always appropriate or even available for completing the task.

It is neither technically wise nor economically practical to build certain kinds of complex systems without first creating a design, a blueprint or another abstract representation. While professional architects might build a doghouse without a design diagram, they would never construct a 15-story office building without first developing an array of architectural plans, diagrams and some type of a mock-up for visualization. Modeling provides professionals with the ability to visualize entire systems, assess different options, and communicate designs more clearly before taking on the risks of actual construction.

This philosophy is also applied to what is related with information and data modelling. There are many explicit advantages in constructing a data model. In order to develop object-oriented conceptual models, and then, to describe their structure from different points of view and at different stages of development (from requirements to implementation), one should use a formalized and commonly accepted language or notation. Currently, this role has been taken by UML (Unified Modeling Language). The UML notation is used in many different fields from the description of business processes to environmental issues such as hydrology or hydrogeology (Muller, 2000; Quatrani, 2002).

## 2.2 Data modelling steps

A data model is an abstract model that describes how data is characterised, represented, and used. The term data model has two generally accepted meanings:

1. A data model *theory*, i.e. a formal description of how data may be structured and used.
2. A data model *instance*, i.e. applying a data model *theory* to create a practical data model *instance* for some particular application.

Having given several definitions, the process of data modelling can be classified in four different stages:

1. mental model of the real world (universe of discourse);
2. conceptual model (CDM);
3. logical model (LDM);
4. and the physical model.

Figure 1 shows the four steps used in the data modelling process.

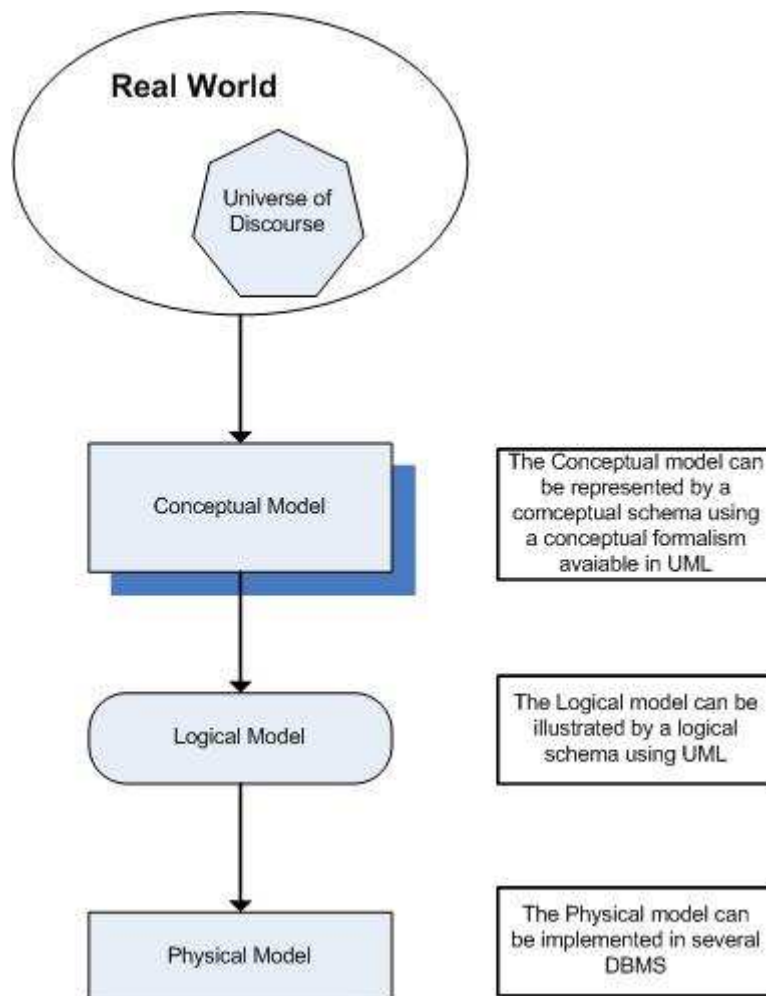


Figure 1| The four steps of data modelling

The mental model permits to know what kind of data to use and how to treat it according to the nature of the universe of discourse. The conceptual model should describe the knowledge needed, as far as geomatics and domain knowledge are concerned in a system design (in this case the system is the Water Framework Directive). Logical and physical models are needed to understand data organisation, structure, and treatment methods to build the system.

The mental model, the universe of discourse, can be seen as the entire knowledge about a domain. This knowledge is needed to represent the reality in a computational system, which will work within certain conditions and meeting specific requirements. As far as the data modelling and the database design are concerned, it is essential to start a design, a blueprint of the data structure to represent the reality of the domain about which the database will be constructed. The international standard, ISO 19110:2005 (Geographic information – methodology for feature cataloguing), may be used as a basis for defining the universe of discourse being modelled in a particular application, or to standardize general aspects of real world features being modelled in more than one application.

The conceptual model is the first practical step in the database design. It defines the concepts of a universe of discourse and can be described by a formalised notation such as UML to become a conceptual schema. It defines the entities and the relationships amongst them, and it requires a particular terminology to properly describe geomatics terms and the WFD concepts.

The logical model is more precise and formalised than the conceptual model. It can be still illustrated by a logical schema, using the UML notation, and it represents a set of data entities, their associations, and several specific rules that could be applied to a specific application. This corresponds to a set of specified information requirements, with the definition of logical constraints on these attributes such as:

- unique identifiers;
- sub-typing;
- data types;
- domains of permissible values.

The logical model promotes normalised and graphical representation of data requirements and related rules. It reinforces understanding and communication between concerned actors and modellers. It permits to correct and validate the assumptions about the specifications.

The goal for the geospatial data models is to provide a practical template for implementing GIS projects. Beyond the benefits to a specific organization, this common starting point results in the creation of data model design templates that simplify the integration of similar data sets at several administration levels.

The physical model of data is a representation of the data design and it takes into consideration the facilities and constraints of the implementation platform. The model here reported could be written, for instance, in Geography Mark-up Language (GML - OpenGIS, 2004), which is an application schema of eXtensible Markup Language (XML). The Figure 2 shows an extract of the GML file header with references to the XML schemas (.xsd) that could be used to validate the GML file produced. This GML file header defines the location of needed XML schemas (XSD) for the GML application The

development of this GML file was not in the scope of this work, but could be considered a future product to be developed based on the work here reported.

It is basically the opportunity to simultaneously make a successful GIS implementation more accessible to average organizations with limited budgets, along with the power to bring consistency and synergy between similar systems. In the long term these common data models are a way to take better decisions based on available geographic information. But these efforts are not designed to create formal standards, rather, they are designed to provide immediate and long-term benefits to people working on real GIS projects while supporting existing standards (like ISO and OGC).

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/gml"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
elementFormDefault="qualified" attributeFormDefault="unqualified"
version="3.1.1">
  <annotation>
    <appinfo source="urn:opengis:specification:gml:schema-
xsd:gml:3.1.1">gml.xsd</appinfo>
    <documentation>Top level GML schema
Copyright (c) 2002-2005 OGC, All Rights Reserved.
For conditions, see OGC Software Notice
http://www.opengeospatial.org/about/?page=ipr</documentation>
  </annotation>
  <!--
===== -->
  <include schemaLocation="dynamicFeature.xsd"/>
  <include schemaLocation="topology.xsd"/>
  <include schemaLocation="coverage.xsd"/>
  <include schemaLocation="coordinateReferenceSystems.xsd"/>
  <include schemaLocation="observation.xsd"/>
  <include schemaLocation="defaultStyle.xsd"/>
  <include schemaLocation="temporalReferenceSystems.xsd"/>
  <!--
===== -->
</schema>
```

Figure 2| GML version 3.1.1 schema definition. (Source: <http://schemas.opengis.net/gml/3.1.1/base/gml.xsd>)

To create different data models schemas, according with the modelling steps referred, the Unified Modeling Language (UML) was used. The UML was used to be compliant with the geographic information ISO standards applied.

The next chapter makes a summary of the UML concepts used in the data model schemas developed.

### 3.1 UML Concepts

The Unified Modeling Language (UML) model is a collection of UML elements, diagrams, and data, such as relationships and requirements information that describe a real-world system. UML is used to build models that describe the quantifiable characteristics of a system such as the system's structure, behavior, constraints, and associations. The building blocks of a UML model are called model elements. Model elements are the data representation of a real world object, actor, or system. (IBM, 2004).

The UML developers wanted to address different scales of architectural complexity and different possible domains of application. The UML notation delivers also extensibility and specialization mechanisms to extend the core concepts. Some of the fundamental advantages of UML as a common standard conceptual schema language are that:

- the complexity is reduced, key components are highlighted and distracting details are temporarily suppressed;
- it lowers development costs;
- in comparison to the Entity-Relationship method, UML has explicit property names, properties expressing relationships between complex objects;
- both IT engineers and hydrogeologists can understand the essence of the system (Vögt 2002);
- it provides an integration possibility between different tools, processes and domains;
- it is possible to create the Catalogue<sup>7</sup> of Feature Types for the application domain, where ISO 19110 (2005) defines a general method for describing the feature-types in a feature catalogue;
- there are several procedures for maintenance and update of definitions of items of interest, described in ISO 19135 (2005), along with a register model for hosting (ISO/CD 19126, 2006);
- it permits a simple implementation of other three ISO standards concerning spatial and temporal aspects of geoinformation (ISO 19107, 2002) for spatial schema including topology, ISO 19111 (2001) presents a model for co-ordinate reference system; ISO 19108 (2002) defines a schema for temporal geometry, topology and temporal reference systems (calendars, time co-ordinate systems, ordinal reference systems);
- it is easy to implement the UML class diagram in various programming environments such as: C++, COBRA, object-oriented database or in XML schema.

The creation of UML models should preferably take place as a structured process. There are many proposed alternative processes for UML modelling, such as: the Unified Software Development Process; the Catalysis; the Select Perspective process; the Rational Unified Process; the UML Components, and others. All these processes are focused on the general development of information systems and components, and do not address the specific concerns relating to standardization of geographic information.

In the geospatial database development process, there are advantages in using UML notation for designing the data structure. Firstly, Object-Oriented modelling techniques have already become the industry trends of convergence and interoperability. Secondly, when conceptual models of geographic information are explored and applied by geospatial analysts, they assure data exchange between different project actors by pre-defined data structures or by using *openweb* standards for data transfer between geographic information systems platforms. Furthermore an unambiguous conceptual model described using UML is easily adaptable and extensible with new components or any existing or available modules.

The model objects used to create the WFD-Art3 data model were the following:

- classes;
- attributes;
- data types;
- relationships and associations;
- stereotypes and tagged values;
- conditional and mandatory attributes and associations;
- naming and name spaces;
- packages;
- notes.

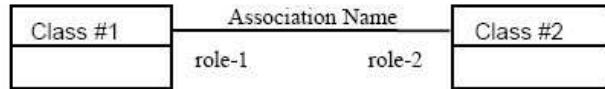
A conceptual schema conformant with the ISO/TS 19103:2005, shall pass all of the requirements described in the abstract test suite (ISO/TS 19103:2005 - Annex A). Non-UML schemas shall be considered conformant if there is a well-defined mapping from a model in the source language into an equivalent model in UML and that this model in UML is conformant.

Many diagrams that appear in this document are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this document are described in Figure 3.

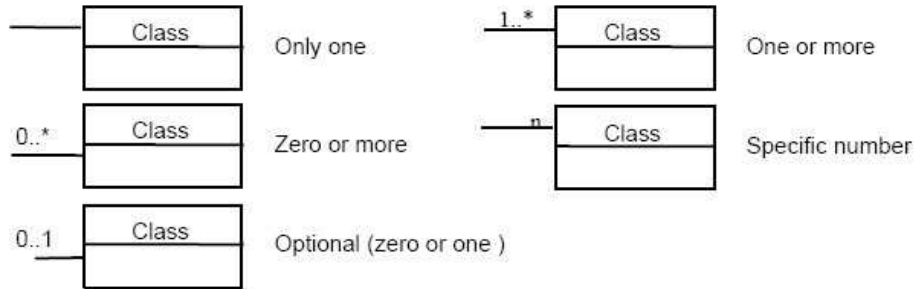
---

<sup>7</sup> A feature catalogue contain the definitions and descriptions of the feature types (feature classes), feature attributes, and feature associations occurring in one or more sets of geographic data, together with any feature operations that may be applied.

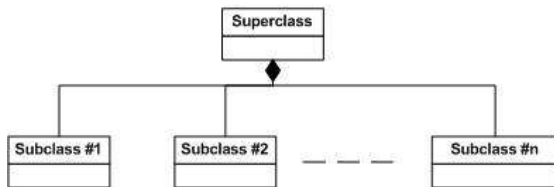
### Association between classes



### Association Cardinality



### Class Inheritance (subtyping of classes)



### Aggregation between classes

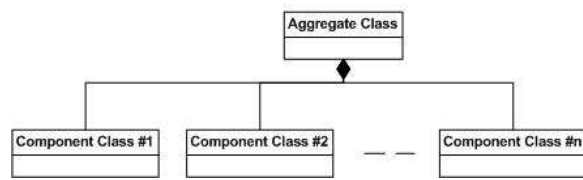


Figure 3| UML notation used in the class diagrams

UML can be useful for relational database design (for example, for schemas that primarily contain feature classes, attribute tables, and a few other properties). However, UML has generally not been useful for designing richer geographic behavior—topologies, networks, terrains, raster catalogs, map layers, map symbols, metadata, cartographic representations, semantic classifications, address locators, cadastral fabrics, linear referencing, and geoprocessing models. These data elements are used to define geographic behavior and associations.

Much of the richness of the geospatial database cannot be universally expressed in a UML design. More important, no special GIS insight is achieved through UML design. Graphing a hierarchy of object-oriented classes, subclasses, and inheritance in UML does not provide insight on how to model the spatial relationships in your geographic data; for example:

- How the river district boundary lines connect to form closed district polygons?
- How basin boundaries, and areas share coincident geometry with one another
- What integrity rules you expect to be maintained as part of their geographic representation in the system.

The users GIS communities found some ways to express their geographic data elements as UML. But because with UML is possible document many, but not all the geographic rules it was developed the

Geography Markup Language (GML), more suitable to declare geographical features. The next chapter presents a short introduction to GML.



### 4.1 GML Concepts

The Geography Markup Language (GML) standard was not used to declare the WFD-Art3 data structure, even though, it was considered useful and pertinent to summarize its fundamental notions.

The Geography Markup Language (GML) is a standard for the modelling, transport and storage of geographic information including the spatial and non-spatial properties defined by the Open Geospatial Consortium (OGC). It is a generic schema to be adjusted to specific needs.

GML is based on the XML 1.0 standard, and uses an XML Schema Definition (XSD file) to validation (OGC, [www.opengeospatial.org/standards/gml](http://www.opengeospatial.org/standards/gml)). The latest GML version (3.1.1) meets the ISO 19136 international standard for geographic information.

It is necessary to distinguish between the GML core schema, a GML profile and a GML application schema:

- The GML core schema provides the objects needed to define the properties of the application objects (e.g. geometry, topology, units);
- A GML profile is a subset of the huge collection of basic object types defined by GML. It fits to a domain and can be created by the "subset tool" as part of the GML specification. Usually these profiles are used to create application schemas;
- A GML application schema defines the objects of interest in a particular application domain (e.g. rivers, lakes, river basins). Thus defining the properties of the object with number and kind of attributes (e.g. discharge, debit).

Figure 4 presents the package diagram of GML 3.1.1.

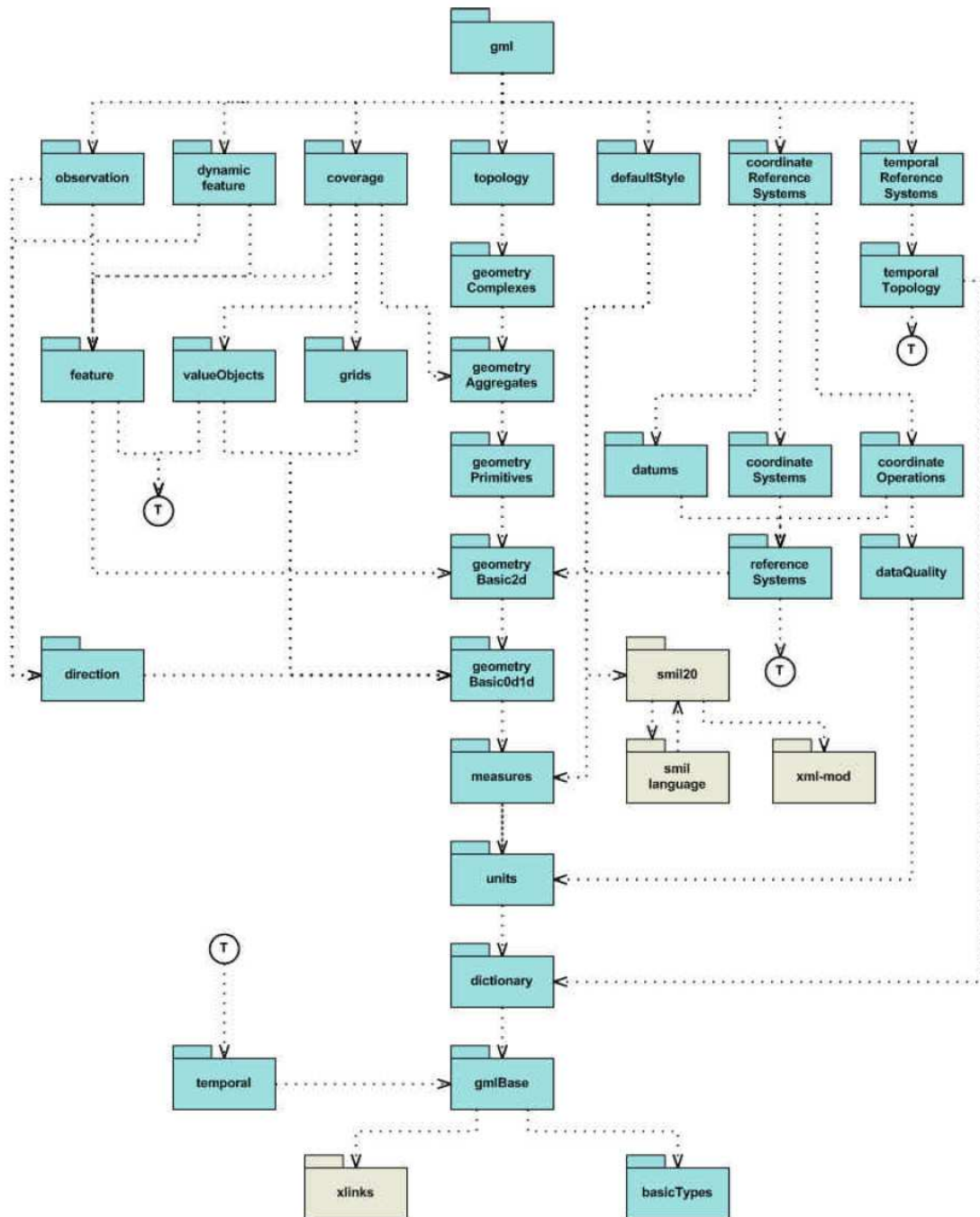


Figure 4| Package diagram of GML 3.1.1. (Source: [www.opengeospatial.org/standards/gml](http://www.opengeospatial.org/standards/gml))

**GML profiles** are logical restrictions to GML, and may be expressed by a document, an XML schema or both. These profiles are intended to simplify the adoption of GML, to facilitate rapid adoption of the standard. The following *profiles*, as defined by the GML specification, have been published or proposed for public use (Wikimedia, GML Article):

- **GML Point** Profile is a very restrictive GML subset including *gml:Point* and its dependencies as well as a coordinate reference system specification;

- **GML Simple Features Profile** is a subset of GML 3.1.1 that supports GML features and a limited set of linearly interpolated geometric types (*gml:Point*, *gml:Curve*, *gml:Surface*, *gml:Geometry*, *gml:MultiPoint*, *gml:MultiCurve*, *gml:MultiSurface*).

There are three compliance levels specified by this profile.

- **Level 0** compliance is the simplest and easiest profile: non-spatial build-in property types are limited to string, real, integer, measurement, date, binary, boolean, URI. The cardinality of properties must not exceed [0..1]. No user defined property types are allowed;
- **Level 1** compliance allows user-defined property types and unbounded cardinalities [0..1];
- **Level 2** compliance is the most complex profile and includes all feature encoding capabilities provided by the OpenGIS Simple Features for SQL Implementation Specification (version 1.1); non-spatial build-in property types are not limited and property references are not restricted.

### 5.1 Framework

The objective of the work was the development of a geospatial data model for article 3 datasets of the WFD. We started with the study of WFD Article 3 requirements, and with the analysis of the Article 3 geographic datasets stored in an Oracle Spatial database, already built to support and manage these data. These primary procedures helped in the definition of the universe of discourse and in the definition of a feature catalogue.

The conceptual model was designed using a CASE tool, called Microsoft Visio V11, to construct the blueprints of the structure of the geospatial database using a graphical language – the Unified Modelling Language (UML). Using UML class diagrams it was possible to represent the geospatial database features, such as: feature types, feature attributes, feature associations, geometric networks, etc.

The logical data model was produced taking into account the conceptual model previously developed. This logical data model was declared to be compliant with Esri Geodatabases<sup>8</sup> rules, and to be implemented in several database management systems (DBMS), namely: Oracle, DB2, Informix, SQL Server, MS Access, and Esri file geodatabase. The viability testes were applied to Oracle, MS Access, and the Esri file geodatabase.

To implement and adjust the physical data model several software components were used. The details of their usage are mentioned in the following chapters.

### 5.2 CASE Tools

The abbreviation CASE stands for Computer-Aided Software Engineering (CASE). CASE tools could be understood as the software tools to assist in the development and maintenance of object models that could be applied to data modelling or software development.

The CASE tool used to construct the WFD-Art3 data models was Microsoft Visio (version 11 – 2003). Also Rational Rose is suggested to be used in this process. A template for constructing the geodatabase models is available for both applications.

The Visio CASE tool was used to produce the UML class diagrams of the different data modelling steps. The Annex F of the document provides a list of proprietary and non-proprietary UML tools. The theory and examples of the data modelling steps are explained in the next chapters.

---

<sup>8</sup> The geodatabase data model is an object-oriented data model for geographic data developed by Esri.

### 5.3 MS Visio

The Microsoft Visio<sup>®</sup>, version 11, was used to design the geospatial database models here documented. To perform these design steps it was used a collection of objects compliant with the UML notation described in the ISO standards, called ArcInfo UML model (Perencsik *et al.*, 2004b).

The ArcInfo UML model (Visio 2003), is a collection of objects, provided by Esri, that were used to design the UML class diagrams of the WFD-Article 3. The Figure 5 illustrates these UML objects.

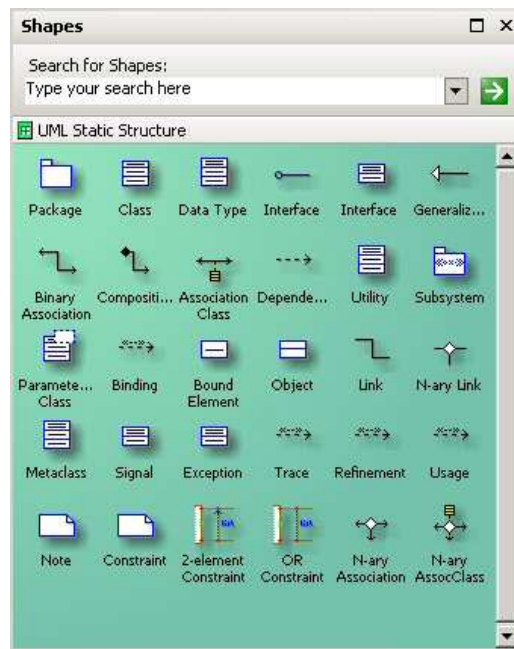


Figure 5| ArcInfo UML model objects

The ArcInfo UML model template (.vst), can be added to MS Visio doing *File/Open*. The template file can be found in the installed ArcGIS folders (*..ArcGIS\CaseTools\Uml Models*). Figure 6 shows the template files in the “Uml Models” installation folder.

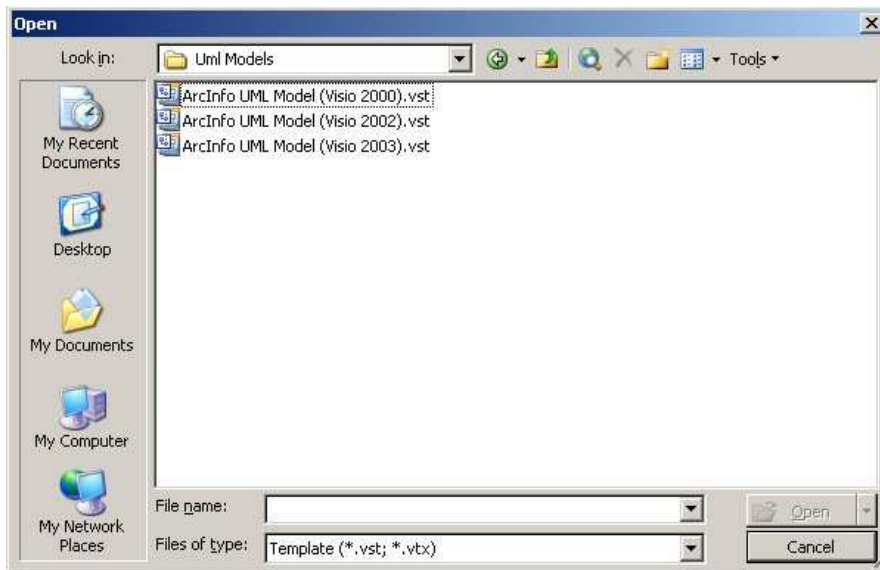


Figure 6| ArcInfo UML model template load

Besides MS Visio there is another CASE tool suggested to design the geospatial data models. The Rational Rose from IBM is the option to use the ArcInfo UML model template as a design pattern. The next chapter gives a short description of this software.

## 5.4 Rational Rose

Rational Rose data modeller is a visual modelling tool for database designers, analysts, architects, and developers. It provides the realization of the Entity-Relation methodology using UML notation. Using UML, the database designer can capture information like constraints, triggers and indexes directly on the diagram rather than representing them with hidden properties. Rational Rose Data Modeler gives the freedom to transfer between object and data models and take advantage of basic transformation types such as many-to-many relationships, providing an intuitive way to visualize the architecture of the database and how it ties into the application ([www-306.ibm.com/software/rational](http://www-306.ibm.com/software/rational)).

The Esri publication "*Designing Geodatabases with Rational Rose*" describe the procedures to construct the UML class diagrams using the ArcInfo UML Model template for Rational Rose

The next chapter introduces the main rules to organize the UML model structure and the static structure diagrams.

## 5.5 UML packages and static structure diagrams

Packages are a convenient way to organize an UML model. They act as folders that can group the model elements. As with directories on a file system, it is possible to create a hierarchy of packages in the model. A package can contain any number of UML elements, such as other packages, classes, interfaces, and static structure diagrams. The static structure diagrams are the "design sheets", and each package can contain one or more static structure diagrams.

There is no limit to the number of packages an object model may contain. For example, a model can have the following packages:

- Classes;
- Domains;
- Network;
- Subtypes;
- Relationships.

The packages created for the geospatial data model are illustrated in the Figure 7 (classes, domains, relationships).

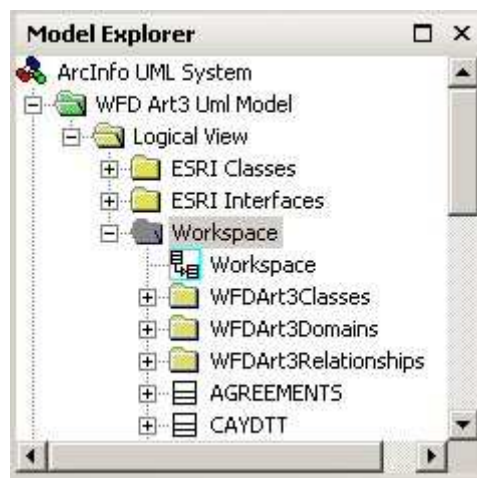


Figure 7| Packages hierarchy implemented in Visio

The UML class diagrams were used to declare the model components following specific procedures and using the ArcInfo UML model template (.vst). This model template has original objects that were cloned to create the WFD-Art3 geospatial data model.

The Figure 8 shows a base diagram that includes the super-classes *Object* and *Feature*. These classes allow the generation of tables and feature classes, respectively.

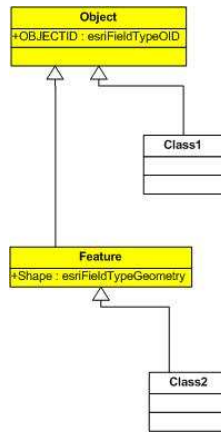


Figure 8| Basic UML diagram to generate tables and feature classes

The feature classes are homogeneous collections of common features (e.g. points, lines), each one having the same spatial representation, and a common set of attributes. The four most commonly used feature classes in a geodatabase are points, lines, polygons, and annotations (the geodatabase name for map text). There is an important rule to follow when designing the data models; all classes that will become a table (like “class1” in Figure 8) should be stored at the “Workspace” package level, as showed in Figure 7 for classes “AGREEMENTS” and CAYDTT”.

The following chapters describe in detail the WFD-Art3 geospatial data model construction process using MS Visio.

## 5.6 Geodatabase diagrammer

The *geodatabase diagrammer* is a tool that can be loaded in ArcCatalog interface to produce diagrams of a geodatabase structure. It behaves like a reverse engineering tool that automatically creates a MS Visio diagram where are represented all the objects of the geodatabase, including features, relationships, topological rules, etc.

This tool does not produce finished diagrams, but the sketch elements that could be used to publish the geodatabase data model, in a more comprehensive design compared with the UML class diagrams. Figure 9 shows the customization window of ArcCatalog from where the tool can be made available.



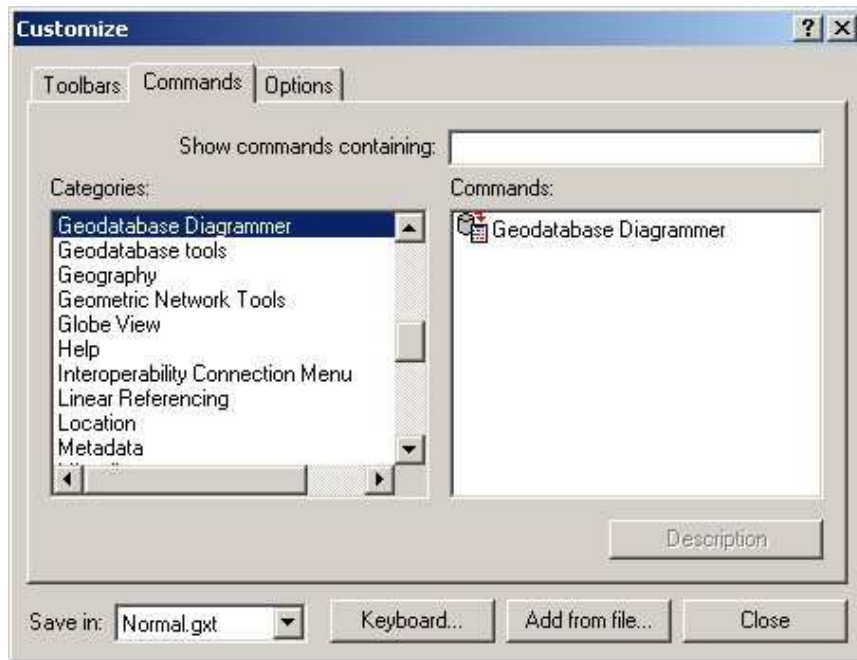


Figure 9| Activation of *geodatabase diagrammer* from ArcCatalog

The installation procedure of this tool is detailed in Annex D.



### 6.1 Framework

The data modelling steps referred to in section 2.2 were executed following international accepted standards: for the declaration of a mental model, it was used the ISO 19110 (Geographic information - Feature catalogue), and for the conceptual model it was used the ISO 19103 (Geographic information - Conceptual schema language). The logical model, as referred before, was developed applying the specificities of Esri geodatabase. The geodatabase logical model was developed using MS Visio as CASE tool, and UML notation to construct the class diagrams.

ArcGIS supports the use of CASE tools to import Unified Markup Language (UML) models for geodatabase designs. However, support for all geographic data types, geometrical relationships, and spatial behaviours could not be fully declared using only UML. This is one of the reasons to choose the Esri Geodatabase approach. No spatial behaviour can be defined through UML design, because UML was not designed to model spatial features with their behaviour in space and time.

Graphing a hierarchy of object-oriented classes, subclasses, and inheritance in UML does not provide insight on how to model the spatial relationships applicable to geographic data. The Esri geodatabase approach is not the only solution available to define spatial behaviours and spatial relationships within a geospatial database, even though, because of the knowledge previously acquired in this type of technology, and because of the market position in government agencies, it was decided to produce an Esri geodatabase design and test several implementation scenarios using tree different DBMS.

The geodatabase is a collection of geographic datasets of various types used in ArcGIS and managed in either a file folder or a relational database. It is the native data source for ArcGIS and is used for editing and data automation in ArcGIS (Esri, ArcGIS 9.2 online help). The Annex H of this document explains the geodatabase feature class types and elements supported.

The geodatabase construction process, followed in this work, is summarized in Figure 10, and could be divided in 3 stages:

1. the first stage represents the construction of the mental, conceptual and logical data models;
2. the second stage is characterized by the creation of the XML file that encapsulates the data structure and geodatabase rules;
3. the third stage is characterized by the implementation of the physical data model in a Database Management System (DBMS).

The generation of code to implement custom behaviour of geographic objects can also be derived from the geodatabase logical data model. This functionality was not used within the work here described.

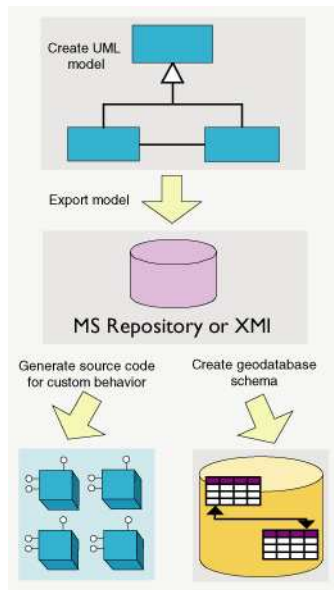


Figure 10| Geodatabase modelling and implementation schema (Source: Esri)

The next sections describe in detail the three stages of the geospatial database construction process.

## 6.2 The geospatial database modelling

The first stage began with the creation of the mental model, declaring the feature catalogue, in accordance with the international specification (ISO 19110 – Methodology for feature cataloguing).

The feature catalogue is a formatted repository for a set of definitions to classify real-world phenomena of significance to a particular universe of discourse. The catalogue provides a means for organizing into categories the data that represent these phenomena, so that the resulting information is as unambiguous, comprehensible, and useful as possible (ISO 19110). A feature catalogue extract is given in Annex G. Based on these preliminary definitions a complete Conceptual Data Model (CDM) was created, integrating all the information of the feature catalogue. The Feature catalogue and the Conceptual data model (CDM) are both platform independent and can be implemented through any application that can read and translate the UML classes' diagram. A part of the CDM model is shown in the Figure 11.

The CASE tool MS Visio was used to draw the CDM. This CASE tool is mentioned in section 5.3.

After the creation of the CDM, the ArcInfo UML model template was loaded in the MS Visio to translate the CDM into the Logical Data Model (LDM). The translation from the CDM into the LDM was based on the ArcInfo UML model template objects. The usage of this UML model permitted to apply the Esri geodatabase rules and specifications to the CDM, converting it into an Esri Geodatabase LDM (Perencsik *et al.* 2004a).

For organising purposes the LDM was split in tree packages, each package containing a part of the model that describes a specific theme, the packages implemented were the:

- Classes package, where the classes that will become feature classes and tables were declared;

- Relationships package, grouping the associations between all classes presented in the model, including the associations between feature classes and tables;
- Domains package, where all the coded value domains and range domains of the table attributes, were declared.

These packages and different diagrams helped the project team to understand and communicate the significance of each feature type and their attributes, and were very useful to visualize the different relationship classes that linked the feature types to one another.

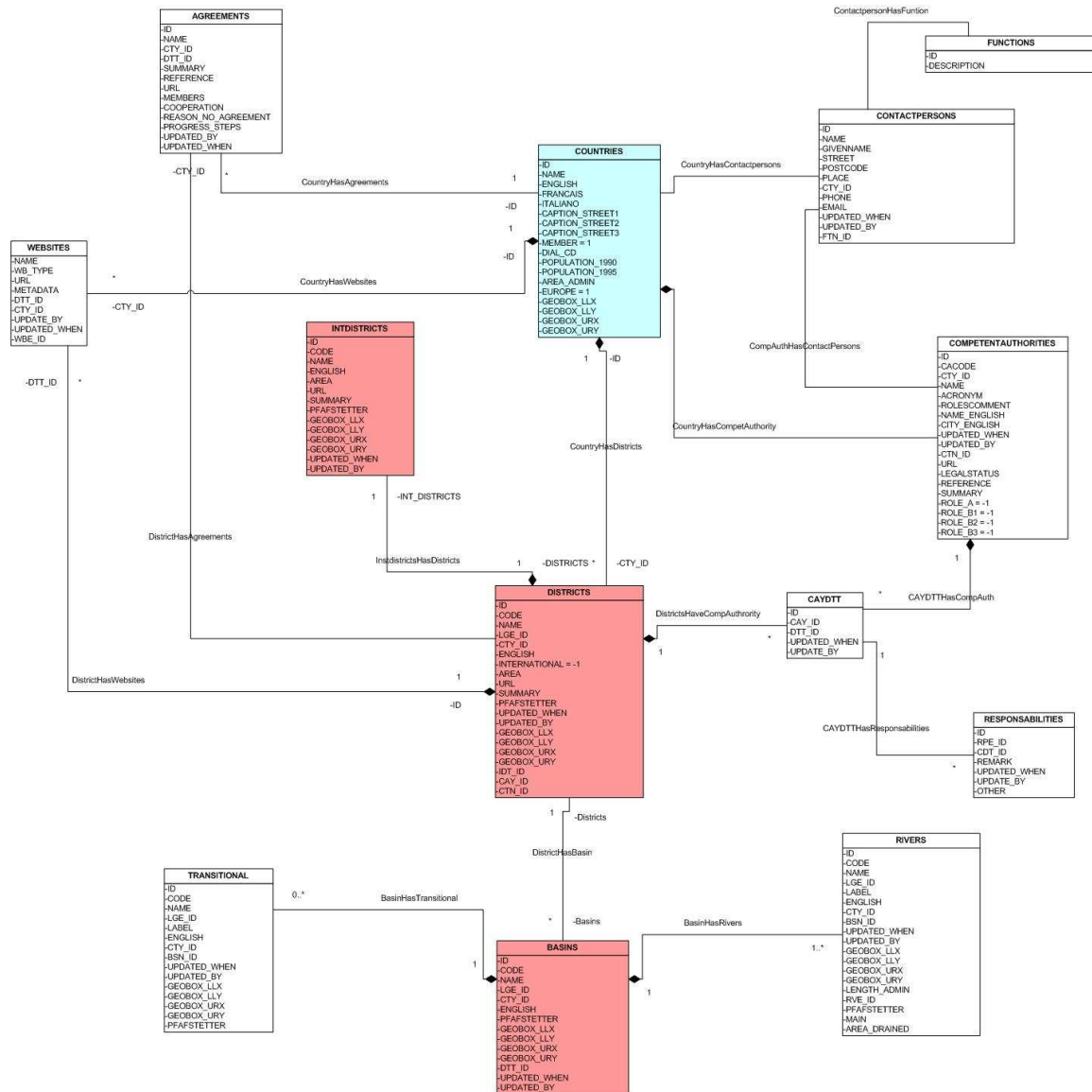


Figure 11| Part of the conceptual model for Article 3 objects. It presents the feature classes and the relationships amongst them

The LDM construction has started with the declaration of the basic generalization associations. This means that the classes "Object" and "Feature" were used to establish which classes had geometric and non-geometric properties. The non-geographic object types were associated to "Object" class, while geographic objects were associated with "Feature" class, inheriting its shape property and

effectively identifying them as (geographical) feature classes. A UML class diagram of the CDM was created for every feature type, feature attributes, their data types and domains. Associations between the UML objects were modeled using binary, composition and generalization relations. While most associations between feature types were modeled using binary associations, characterized by their name and cardinalities as defined in the feature catalogue, in a limited number of cases, composition relations were used to indicate relationships between feature types.

Figure 12 shows the base schema of an Esri Geodatabase. The classes are connected using a generalization association, meaning that all the properties of the super-class are inherited by the sub-class.

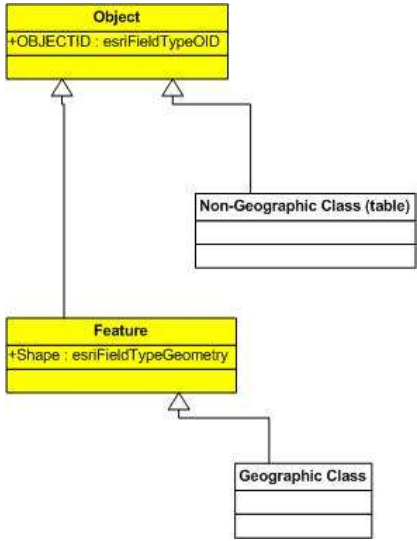


Figure 12| Esri geodatabase: super-class inheritance diagram

The static structure diagram to declare the class inheritance (in the Classes package) within the logical data model is represented in Figure 13. The classes colored in yellow represent the super-classes from which all other classes inherit their basic properties; the classes colored in red represent central classes of the database. The colors attributed to the classes don't influence the UML diagram and have only a visual effect.

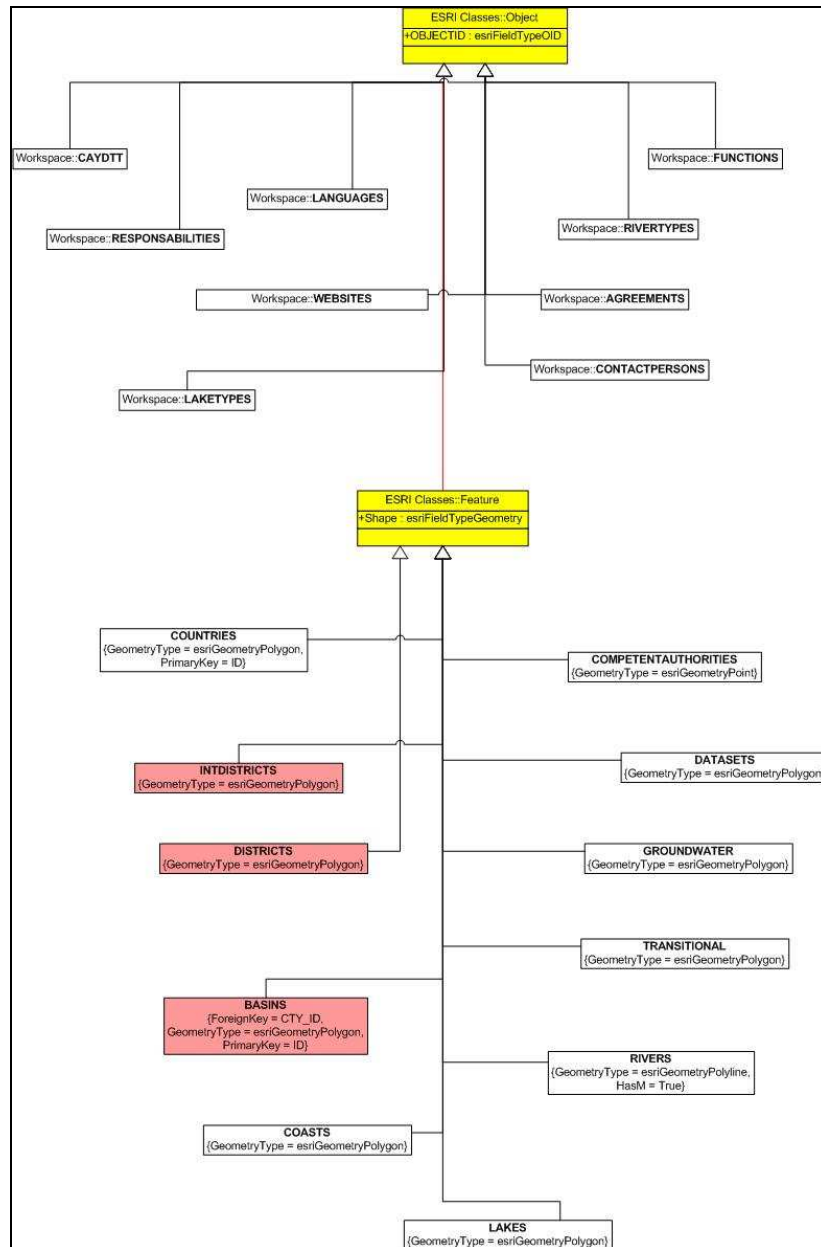


Figure 13| Logical data model: class inheritance static structure diagram

The Classes package, containing the feature class definitions, was stereotyped as a feature dataset, as showed in Figure 14. This stereotyping mark corresponds to a dataset that will be created during the generation of the physical database schema. The non-geographic classes (tables) were declared outside the feature dataset package; these types of classes (table classes) cannot exist inside a feature dataset.

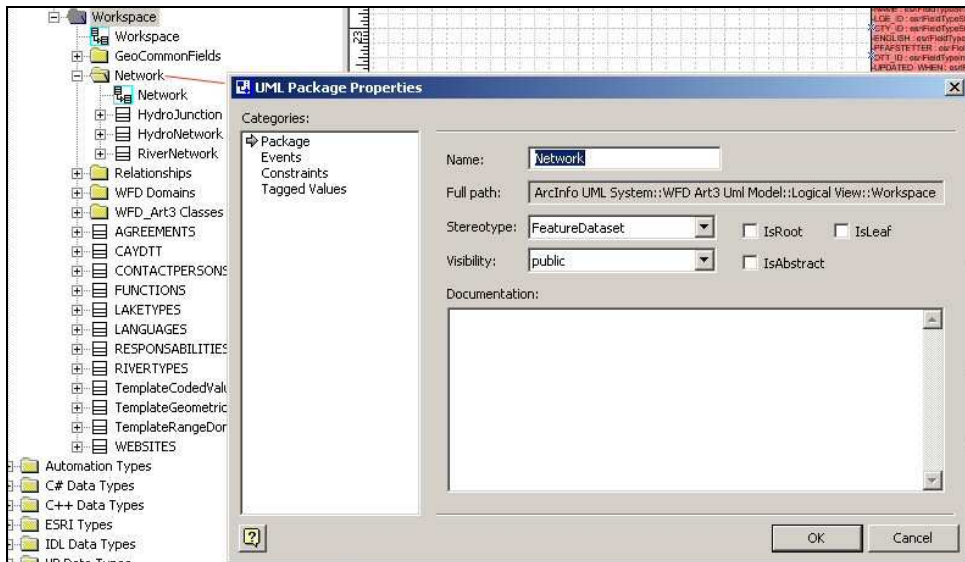


Figure 14| UML package properties stereotyped *FeatureDataset*

After the creation of the geodatabase LDM, the model needed to be checked for compliance with the geodatabase rules. Before execute the compliance check, the UML class diagrams had to be exported to an XML file.

The next section explains the procedure to create de XML file that encapsulates the geodatabase structure and rules.

## 6.3 The XML Metadata Interchange file (XMI)

### 6.3.1 Overview

The creation of the XMI file, that includes the UML class diagram of geospatial data model, corresponds to the second of the three stages of geodatabase construction.

The XML Metadata Interchange (XMI) is an OMG<sup>9</sup> standard for exchanging metadata information via eXtensible Markup Language (XML). The most common use of XMI is as an interchange format for UML models, although it can also be used for serialization of models of other languages (metamodels). XMI is also commonly used as the medium by which models are passed from modelling tools to software generation tools as part of model-driven engineering (Wikimedia, 2007).

The XMI file was created using MS Visio and an add-on called *Esri XMI Export*. The instructions about the installation of this tool are detailed in Annex B of this document. The tool is available through the MS Visio interface at *Tools/Add-Ons/ESRI XMI Export*, like showed in Figure 15.

<sup>9</sup> Object Management Group (OMG) is a consortium, originally aimed at setting standards for distributed object-oriented systems, and is now focused on modeling (programs, systems and business processes) and model-based standards.



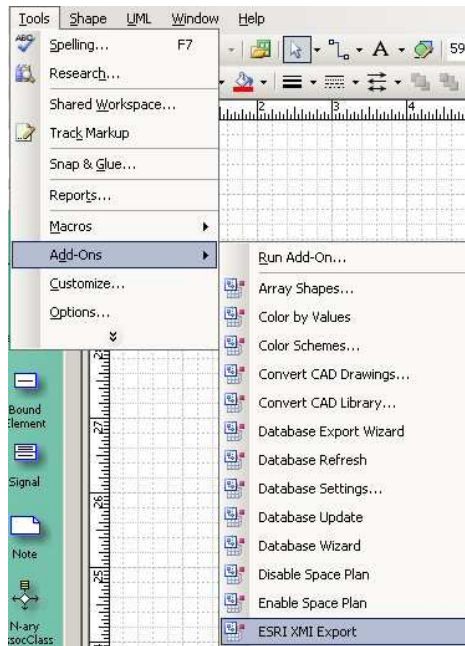


Figure 15| XMI Export tool in MS Visio

The tool was used to export the WFD-Art3 geodatabase logical data model. The XMI file needs to be exported to a directory containing a DTD<sup>10</sup> file (uml.dtd), used to validate the XMI file exported according with the geodatabase rules. It defines the correct XMI document structure by providing a list of legal elements.

This file can usually be found in the ArcGIS installation folders in “*..Program Files\ArcGIS\CaseTools\Utilities*”. If the XMI file passes the validation check against the DTD file, a message box like the one shown in Figure 16 appears.



Figure 16| Succeeded export of XMI file

Because an XML file can be valid without the correct semantic construction, the XMI file exported need also to be checked for its semantic construction. The next section explains the details of this process

---

<sup>10</sup> Document Type Definition (DTD), is one of several XML schema languages, and is also the term used to describe a document or portion thereof that is authored in the DTD language. A DTD is primarily used for the expression of a schema via a set of declarations that conform to a particular [markup](#) syntax and that describe a class, or *type*, of XML documents, in terms of constraints on the structure of those documents.

### 6.3.2 The creation of XMI

The *Esri UML Semantics Checker* is a tool used to check the general integrity of XML Metadata Interchange (XMI) files. This tool is most commonly run inside of MS Visio as part of a geodatabase template macro, but can also be launch from ArcCatalog. The tool uses a DLL file called *UmlSemCheck.dll* which can be found in the Arcgis installation folder. Before the macro can be executed, a reference had to be made to the *UmlSemCheck.dll* file in the Visual Basic Editor.

The macro also uses the DTD file (*uml.dtd*), which again can be found in the Arcgis installation folder. For the Semantics Checker macro to work, the *uml.dtd* file needs to be placed in the same directory as the XMI file containing the model.

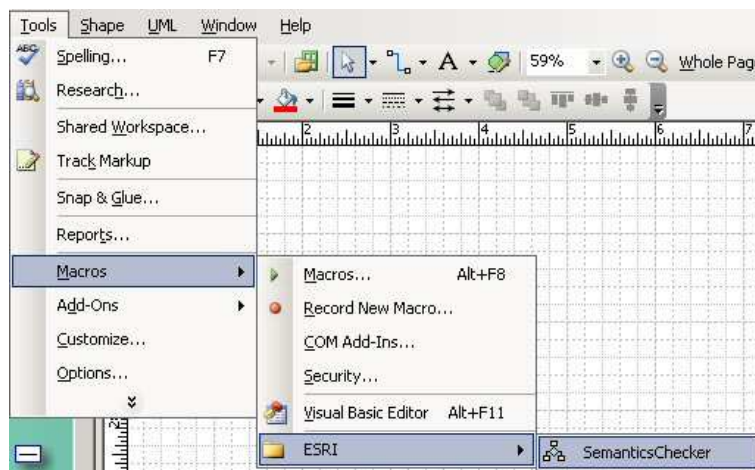


Figure 17| Running of the Semantic checker to evaluate the UML model

The instructions to run the *Esri UML Semantics Checker* are explained in Annex E of this document.

The Semantics Checker was run on the exported model, with all types of warnings selected, as illustrated in Figure 18.

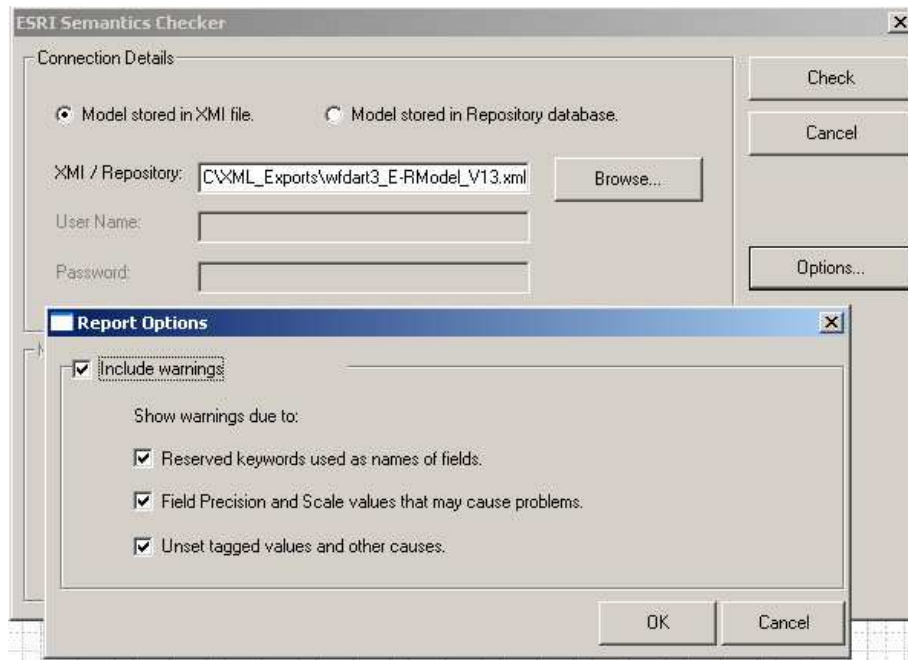


Figure 18| Semantic checker dialog box

The resulting report contained the references for a number of warnings and errors that were found in the model, documenting the names of the elements involved, the location of these elements within the model, and a brief description of the problem. Most of these could be attributed to human errors in entering data in MS Visio and were easily resolved. Figure 19 shows an example of the *Semantic Checker Report*.

If the tool report warnings, usually it is still possible to implement the physical data model, but, if errors are reported, it is strongly recommended to solve the errors in the logical data model first. To solve an error, it is advisable to change the UML class diagram and export again it to a new XMI file.

**UML Model Error Report.** 8/2/2007 3:34:29 PM  
**Model : WFD Art3 Uml Model**

Element name	Type	Num	Description	Severity	Path
River_Network	COMClass	20	Invalid name for custom feature River_Network.	Warning	Logical View:Workspace:Network:River_Network
INT_DISTRICTS	COMClass	20	Invalid name for custom feature INT_DISTRICTS.	Warning	Logical View:Workspace:WFD_Art3 Classes:INT_DISTRICTS
CAY_DTT	COMClass	20	Invalid name for custom feature CAY_DTT.	Warning	Logical View:Workspace:WFD_Art3 Classes:CAY_DTT
Countries:Europe	Domain	5	Invalid Coded Value Domain: Domain : Countries:Europe Check : Yes	Error	Logical View:Workspace:WFD Domains:Countries:Europe
Countries:Europe	Domain	5	Invalid Coded Value Domain: Domain : Countries:Europe Check : No	Error	Logical View:Workspace:WFD Domains:Countries:Europe
Members	Domain	5	Invalid Coded Value Domain: Domain : Members Check : Yes	Error	Logical View:Workspace:WFD Domains:Members
Members	Domain	5	Invalid Coded Value Domain: Domain : Members Check : No	Error	Logical View:Workspace:WFD Domains:Members
Competent_Authorities:Roles	Domain	5	Invalid Coded Value Domain: Domain : Competent_Authorities:Roles Check : Yes	Error	Logical View:Workspace:WFD Domains:Competent_Authorities:Roles
Competent_Authorities:Roles	Domain	5	Invalid Coded Value Domain: Domain : Competent_Authorities:Roles Check : No	Error	Logical View:Workspace:WFD Domains:Competent_Authorities:Roles

Figure 19| Semantics checker report

If there are no errors in the data model, a warning message, like the one showed in Figure 20, is exhibited.

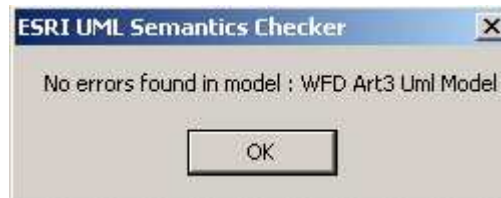


Figure 20| Errors check result

From the corrected XML file, a physical database schema was created using ArcCatalog, with the *Schema Wizard* tool. While this tool provides the opportunity to set a number of properties of the geodatabase, most of these had already been incorporated in the WFD-Art3 LDM, using tagged values. Because of this, the interactive work during the schema creation was very limited. Only the spatial reference system of feature datasets was added to complete the model.

The implementation of the physical data model within a DBMS initiates the third stage of the geodatabase construction. The details of this process are referred in the next section.

## 6.4 Physical data model implementation

Since the choice to implement the geospatial database structure was to use the geodatabase technology from Esri, is strongly recommended to use the Esri software modules to implement the physical data model in the supported DBMS. There are several DBMS supported by this technology, including: MS Access implementation (Personal Geodatabase), the file based geodatabase, and the enterprise geodatabase (that could be implemented over Oracle, SQL Server, DB2 and Informix). The geodatabase implementation doesn't require ArcSDE (*Spatial Database Engine*) if it uses MS Access or the file based solution. To implement an enterprise geodatabases in the DBMS's referred, ArcSDE is required as a geospatial data gateway.

The implementation of the data model was tested with MS Access, the file based geodatabase, and with an Oracle DBMS. It was used ArcGIS and a software component called *Schema wizard* was used to generate the database structure in a semi-automatic process.

The *schema wizard* can be used in ArcCatalog but is not available by default. It has to be added doing *Customize*, to choose, and add, the *Schema Wizard* button. Figure 21 shows this procedure.



Figure 21| Schema wizard to implement the physical model

The schema wizard tool becomes active when an empty geodatabase is selected. An empty geodatabase means that only the system tables exist with no data loaded on them. The name and structure of these system databases depends on which database system is being used. More information about this issue can be found in "How geodatabases are stored in a DBMS", at: <http://webhelp.esri.com/arcgisdesktop/9.2>. Figure 22 shows how to create an empty geodatabase in ArcCatalog.

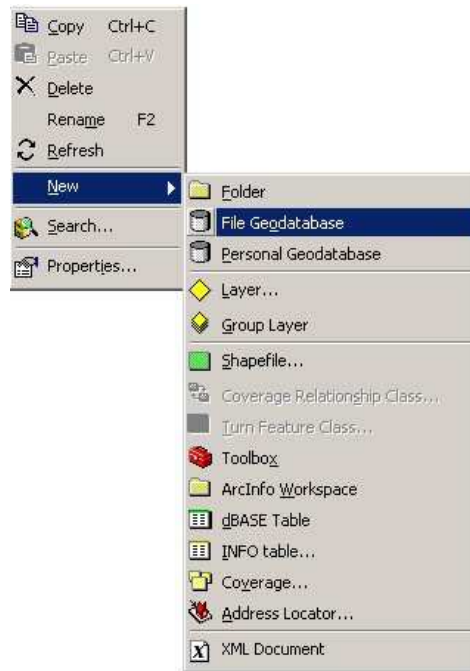


Figure 22| Creation of an empty geospatial database

After creating an empty geodatabase, the schema wizard tool becomes active and is possible to use the XMI file to generate automatically the data model schema, as Figure 23 shows.



Figure 23| The use of the XMI to generate the database structure

Some new errors that had not been detected by the *Semantic Checker* appeared during the final steps of the Schema Wizard. It turned out that some of the names of features were reserved SQL keywords and had to be changed in the LDM. Other examples of errors were due to the fact that: the classes names in the model: must be unique, must not use \_(underscore) in their names, must not start with a number, and must not have blank spaces. All these issues were solved in the UML class diagram and the process of exporting the XMI file and checking were performed again, resulting in a correct XMI file.

Figure 24 shows the first screen of *schema wizard* tool, where it is possible to select the XMI file used to generate the geodatabase structure.



Figure 24| Schema wizard used to import the XMI files

The tool provides the means to establish additional settings that aren't possible to settle through the UML class diagram of the logical data model: the geographic reference system, the coordinate domains and the precision are examples of those settings. Additional information about these settings could be found in Annex C.

Figure 25 shows one of the steps of *schema wizard*, where the user can access a summary of the objects to be created, and establish the properties of the geodatabase.



Figure 25| Schema wizard parameters definition

The spatial reference system, the coordinates domains, and precision should be established taking into account the dataset to be stored. In this case, the world geographic reference system (WGS84) and a system in decimal degrees are being used. Figure 26 shows the interface where the spatial reference system parameters are defined when implementing the physical data model in a DBMS.



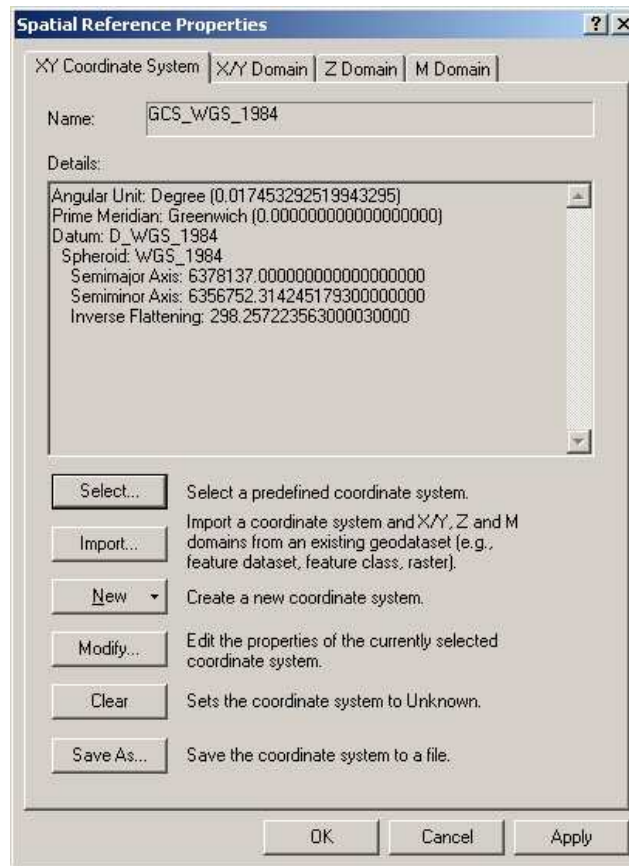


Figure 26| Geodatabase spatial reference system parameters

Depending on the DBMS being used, there are several configuration instructions that can be settled concerning the parameters of storage. These configuration instructions are called configuration keywords and represent a setting or group of settings that establish where or in what format to store data contents in each dataset. Configuration keywords are used with file geodatabases and with ArcSDE enterprise geodatabases, but they cannot be specified for personal geodatabases (MS Access) or for ArcSDE Personal or Workgroup geodatabases.

Each of these geodatabase types has its own set of keywords. In the case of ArcSDE geodatabases, there are a number of configuration keywords that can be used for specifying geometry storage (which varies by DBMS) as well as for managing geometric networks and topologies. More information about configuration keywords can be found in Esri documentation about geodatabases.

After the implementation of the WFD-Art3 geodatabase schema, several tests were made concerning data loading, coming originally from an Oracle Spatial DBMS. To translate the data from one database to the other, it was used the FME translator.

### 7.1 Overview

To construct the geospatial database, several phases were considered: i) the design phase, where the data models were produced; ii) the XMI creation phase, where the data structure was encapsulated in an interchangeable format; and iii) the physical implementation phase, where the structure was implemented within a DBMS.

Within the ESRI technology there are three types of geospatial databases:

1. Personal geodatabases (MS Access): all datasets are stored within a Microsoft Access data file, which is limited in size to 2 GB;
2. File geodatabases: all datasets are stored as files in a file directory system. Each dataset is held as a file that can scale up to 1 TB in size; this option is recommended over personal geodatabases;
3. ArcSDE geodatabases: stored in a relational database using Oracle, Microsoft SQL Server, IBM DB2, or IBM Informix. These multiuser geodatabases require the use of ArcSDE and can be unlimited in size and number of users.

Table 1| ArcGIS geodatabase types

Characteristics	ArcSDE Geodatabase	File Geodatabase	Personal Geodatabase (MS Access)
<b>Description</b>	A collection of various types of GIS datasets held as tables in a relational database This is the recommended native data format for ArcGIS stored and managed in a relational database.	A collection of various types of GIS datasets held in a file system folder. This is the recommended native data format for ArcGIS stored and managed in a file system folder.	Original data format for ArcGIS geodatabases stored and managed in Microsoft Access data files This is limited in size and tied to the Windows operating system.
<b>Number of Users</b>	Multiuser ArcSDE can be used at three levels: <ul style="list-style-type: none"> <li>• Personal ArcSDE</li> <li>• Workgroup ArcSDE</li> <li>• Enterprise ArcSDE</li> </ul>	Single user and small workgroups Some readers and one writer per feature dataset, standalone feature class or table. Concurrent use of any specific file eventually degrades for large numbers of readers.	Single user and small workgroups with smaller datasets  Some readers and one writer. Concurrent use eventually degrades for large numbers of readers.
<b>Storage Format</b>	<ul style="list-style-type: none"> <li>• Oracle</li> <li>• Microsoft SQL Server</li> <li>• IBM DB2</li> <li>• IBM Informix</li> </ul>	Each dataset is a separate file on disk A file geodatabase is a file folder that holds its dataset files.	All the contents in each personal geodatabase are held in a single Microsoft Access file (.mdb).
<b>Size Limits</b>	Up to DBMS limits	One TB for each dataset. Each file geodatabase can hold many datasets Each feature class can scale up to hundreds of millions of vector features per dataset.	Two GB per Access database  Effective limit before performance degrades is typically between 250 and 500 MB per Access database file.
<b>Versioning Support</b>	Fully supported across all DBMSs; includes cross-database replication and updates	Not supported	Not supported

<b>Platforms</b>	Windows, Unix, Linux, and direct connections to DBMSs that can potentially run on any platform on the user's local network	Cross-platform	Windows only
<b>Security and Permissions</b>	Provided by DBMS	Operating file system security	Windows file system security
<b>Database Administration Tools</b>	Full DBMS functions for backup, recovery, replication, SQL support, security, and so on	File system management	Windows file system management
<b>Notes</b>	Requires the use of ArcSDE	Allows you to optionally store data in a read-only compressed format to reduce storage requirements	Often used as an attribute table manager (via Microsoft Access). Users like the string handling for text attributes.

The geodatabase structure is made of the following building blocks:

- **Tables**, are the basic storage objects in the database. In these, tables can store descriptive attributes as well as spatial attributes. The geodatabase uses tables to store and manage the attributes and properties of geographic objects.
- **Spatial indexes**, the geodatabase uses a system of grids to create a spatial index. When performing such tasks as panning, zooming, or selecting features, the spatial index is used to quickly locate features. The geodatabase uses the spatial index to increase the efficiency of spatial searches on the data
- **Spatial reference**, describes where features are located in the real world. The spatial reference is defined when creating a geodatabase feature dataset or stand-alone feature class. The spatial reference includes a coordinate system for x-, y-, and z-values as well as tolerance and resolution values for x-, y-, z-, and m-values. The geodatabase uses the spatial reference to accurately display a features location and carry out geoprocessing functions.
- **Configuration keywords**, specify how data is stored in the geodatabase. Configuration keywords represent a setting or group of settings that tell the geodatabase where or in what format to store data contents in each dataset. The geodatabase uses configuration keywords to optimize the storage parameters.

To manage the spatial database, it is possible to create a file or personal geodatabase. A file geodatabase offers all the functionality of a geodatabase, but it is stored in a file structure, whereas, with a personal geodatabase the data will be stored in a Microsoft Access database. For a multi-user spatial database, it is possible to use ArcSDE (Spatial Database Engine), which lets many people in an organization simultaneously update data stored in a centrally located DBMS.

It is possible to access file and personal geodatabases directly in ArcGIS, but to access data stored in a DBMS, a spatial database connection, or an OLE DB<sup>11</sup> database, a connection should be created to

---

<sup>11</sup> Microsoft's OLE DB providers come with the personal computers. They let the user to access Jet (Microsoft Access), SQL Server, and Oracle databases. Another provider communicates with Open Database Communication (ODBC) drivers. Additional OLE DB providers may be available from other sources. The Connection tab is different for each provider, although they all require similar information (user name, password, and the database to which to connect).

access and manage the data. Information, such as the user name, password, and the database to which connect would be necessary. For ArcSDE Personal and Workgroup geodatabases, to access the geodatabase the option of creating a connection to the database server on which the geodatabase resides is available. Figure 27 shows the connection properties to an SDE geodatabase based on Oracle.

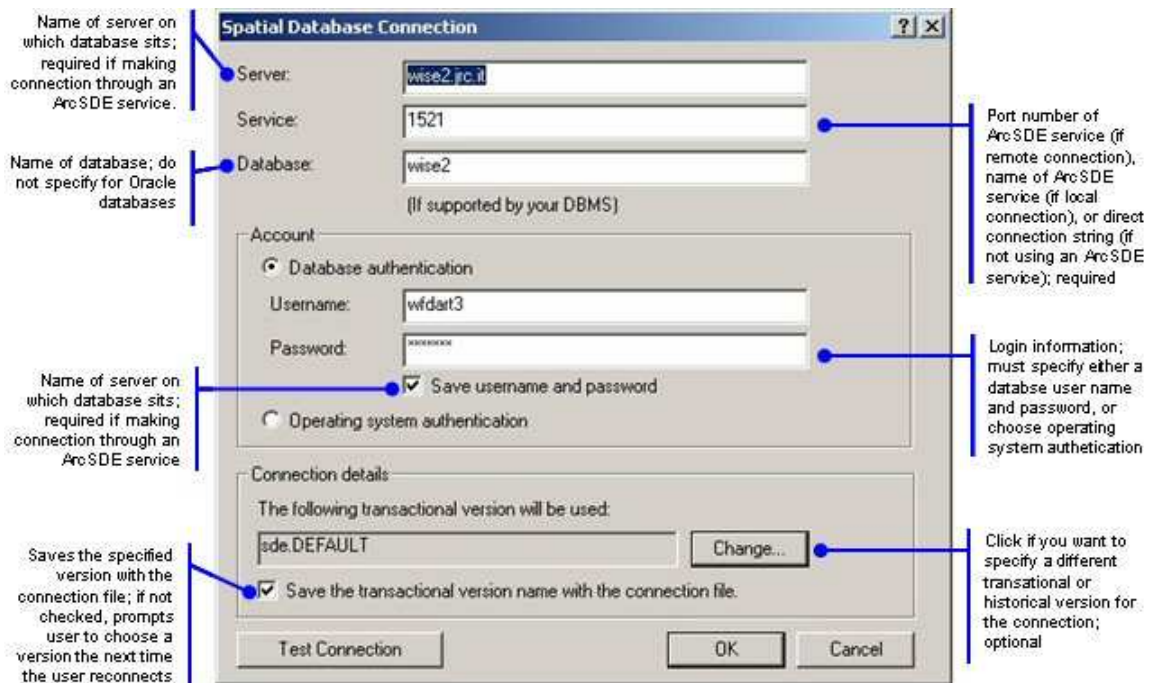


Figure 27| Database connection properties

## 7.2 Implementation scenarios

### 7.2.1 Scenario 1: MS Access (personal geodatabase)

The personal geodatabase was created using ArcCatalog by running the *schema wizard*, to implement the physical data model, according with the parameters showed in Table 2.

Table 2| Geodatabase creation parameters

Coordinate System	Geographic Coordinate System: WGS84		
X/Y Domain	MinX: -180	MaxX: 90071992547.4099	Precision: 100000
	Min Y: -90	MinY: 90071992547.4099	
M Domain	MinM: 0	MaxM: 90071992547.4099	Precision: 100000
Z Domain	MinZ: -200	MaxZ: 90071992547.4099	Precision: 100000

The creation of the geodatabase structure did not generate any errors and all the objects were created without warnings.

## 7.2.2 Scenario 2: File-based geodatabase

The creation of a file geodatabase produce a file directory populated with a specific structure of files within. The main characteristics of a file geodatabase are:

- each dataset can scale up to 1 TB in size;
- designed to be edited by a single user and do not support geodatabase versioning;
- works across operating systems;
- uses a data structure that is optimized for performance and storage. File-geodatabases use about 1/3 of the feature geometry storage required by shapefiles;
- includes a read-only compression option that allows users to create a read-only, highly compressed geodatabase.

The File Geodatabase is recommended over a MS Access geodatabase.

## 7.2.3 Scenario 3: Oracle

To implement the data model using ESRI technology, a software component from Esri to Oracle databases, the *Spatial Data Engine* (ArcSDE), was used.

The Oracle Spatial extends the Oracle database with the addition of spatial data management functions. Oracle Spatial provides a spatial geometry type (MDSYS.SDO\_GEOMETRY), spatial metadata schema, indexing methods, functions, and implementation rules, which are described below. Detailed information can be found at: <http://webhelp.esri.com/arcgisdesktop/9.2>, searching by: Oracle spatial geometry type.

A summary of the available feature geometry storage types for feature geometry within each DBMS is provided in the table below (ESRI, 2007).

Table 3| Feature geometry storage types in Oracle database

DBMS	Feature geometry storage	Column type	Notes
Oracle	ArcSDE Compressed Binary	Long Raw	- Provides high performance, scalability, and reliability - Default for ArcSDE for Oracle
	Well-Known Binary (OGCWKB)	BLOB	- OGC simple features type - Can only be used with 2-dimensional geometry
	LOB	BLOB	Can be deployed to use Oracle Replication Services
	Spatial Type	ST_GEOMETRY*	- Uses extended spatial type for managing vector data - Based on the ISO SQL MM specification for spatial

## 7.2.4 The ArcSDE Compressed Binary Storage

The ArcSDE Compressed Binary storage type has traditionally been the primary storage type for ArcSDE geodatabases in Oracle and SQL Server. It uses a binary storage mechanism for storing feature geometry and adheres to the Open GIS Simple Features specification for binary. No SQL API is available to access feature geometry in the DBMS stored using a compressed binary storage type. However, it is accessible via ArcSDE and ArcGIS and is widely used in a majority of ArcSDE geodatabase implementations because of its low storage and high performance. Detail information can be found at:

<http://webhelp.esri.com/arcgisdesktop/9.2>, searching by: ArcSDE Compressed Binary.

The well-known binary representation for OGC geometry (*WKBgeometry*) provides a portable representation of a geometry value as a contiguous stream of bytes. It permits geometry values to be exchanged between an ODBC client and a database in binary form. It is not compressed.

This geometry storage type can be used with ArcSDE geodatabases stored in Oracle or SQL Server to store two-dimensional geometry. Feature classes stored in the OGC Well-Known Binary type are also made up of three tables—the business table, the feature table, and the spatial index table—just like feature classes stored in ArcSDE Compressed Binary. Detail information could be found at:

<http://webhelp.esri.com/arcgisdesktop/9>, searching by: OGC Well-Known Binary.

ArcSDE supports Oracle Spatial's Object Relational Model as a method to store spatial data. Oracle Spatial extends the Oracle database with the addition of spatial data management functions, spatial metadata schema, indexing methods, and implementation rules. Oracle Spatial provides a spatial geometry type (*MDSYS.SDO\_GEOMETRY*), and a spatial raster type (*MDSYS.SDO\_GEORASTER*).

Oracle Spatial is an extension to the Oracle DBMS that adds a spatial type and spatial query functions to Oracle. It is offered by Oracle using two primary options:

- Oracle Spatial is an optional feature of the Oracle Database Enterprise Edition; in addition to providing the *SDO\_Geometry type*, Oracle Spatial provides a number of additional geospatial capabilities;
- *Oracle Locator* provides a subset of Oracle Spatial capabilities; it is included as a standard feature of Oracle Database Standard and Enterprise Editions. Among other capabilities, it provides the Oracle Spatial geometry type (referred to as *SDO\_Geometry*) and a SQL API to this content.

ArcSDE supports Oracle Spatial or Oracle Locator for storing and managing geometry in an Oracle database. To use it, *SDO\_GEOMETRY* must be specified for the *GEOMETRY\_STORAGE* parameter of one of configuration keywords. When wanting to use Oracle Spatial geometry storage by default, the *SDO\_GEOMETRY* should be specified for the *GEOMETRY\_STORAGE* parameter as default configuration keyword. To use only for some datasets, *SDO\_GEOMETRY* should be specified as keyword when creating each individual dataset.

When ArcSDE is installed, the ArcSDE compressed binary geometry schema is the default storage type. The default settings for ArcSDE storage are defined in the *DBTUNE* table by the *GEOMETRY\_STORAGE* parameters. To change the default ArcSDE geometry storage, the *GEOMETRY\_STORAGE* parameter should be changed from *SDEBINARY* to *SDO\_GEOMETRY*. After the default *GEOMETRY\_STORAGE* setting has been changed to *SDO\_GEOMETRY*, ArcSDE creates feature classes with *SDO\_GEOMETRY* columns by default.





## 8.1 Overview

After the creation of the geodatabase within a DBMS, it is possible to load data into the existing feature classes and tables using the following methods:

- the *Simple Data Loader* in ArcCatalog;
- the *Object Loader* in ArcMap;
- an universal translator such as the *Feature Manipulation Engine* (FME).

Each of the methods referred to above have its particularities and dependencies, regarding the storage platform used and the parameters settled during the implementation of the geodatabase.

The data model applied has feature classes, tables, relationships, and domains, that establish a complex and intelligent structure to help data to be consistent and to respect the rules of the universe of discourse represented. Because of these rules, the data loading must respect a specific order and specific methods.

The following sections explain how each of these three methods work.

## 8.2 Loading data with Simple Data Loader (ArcCatalog)

The *Simple Data Loader* is a wizard, available in ArcCatalog, that allow to specify a number of source tables and feature classes, providing a schema match process. It allows specify which fields in the input data are loaded into which fields of the target feature class or table.

This method was tested to load Esri shapefiles within a personal geodatabase (using a MS Access file), and within a file based geodatabase, where the data model have been created. The Shapefiles were exported from a central Oracle database using Feature Manipulation Engine (FME). The data types of the original data, stored in Oracle Spatial, are shown in Figure 28.

The tests here reported were performed with the feature class “Basins”.

TABLE_CAT	TABLE_SCHEM	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	COLUMN_SIZE
(null)	WFDART3	BASINS	ID	3	NUMBER	10
(null)	WFDART3	BASINS	CODE	12	VARCHAR2	24
(null)	WFDART3	BASINS	NAME	12	VARCHAR2	100
(null)	WFDART3	BASINS	LGE_ID	12	VARCHAR2	2
(null)	WFDART3	BASINS	CTY_ID	12	VARCHAR2	2
(null)	WFDART3	BASINS	ENGLISH	12	VARCHAR2	100
(null)	WFDART3	BASINS	PPAFSTETTER	3	NUMBER	6
(null)	WFDART3	BASINS	GEOBOX_LLX	3	NUMBER	6
(null)	WFDART3	BASINS	GEOBOX_LLY	3	NUMBER	6
(null)	WFDART3	BASINS	GEOBOX_URX	3	NUMBER	6
(null)	WFDART3	BASINS	GEOBOX_URY	3	NUMBER	6
(null)	WFDART3	BASINS	DTT_ID	3	NUMBER	10
(null)	WFDART3	BASINS	POLYGON	1111	SDO_GEOMETRY	1
(null)	WFDART3	BASINS	UPDATED_WHEN	91	DATE	7
(null)	WFDART3	BASINS	UPDATED_BY	12	VARCHAR2	15

Figure 28| Data types of feature class Basins

The physical data model constitutes a replica of original fields' names and data types. In this case, the matching matrix should have a relation of 1 to 1, this means that, all the fields stored originally in Oracle spatial database should match with the fields described in the data model applied to personal geodatabases, file based geodatabases, and ArcSDE geodatabases.

The physical data model applied to personal geodatabase have produced, to the fields "GEOBOX\_" a Long Integer data type, which means that, the original data, defined in Oracle as *Number (6,3)*, could not be stored with the right decimal places. To solve this problem, the logical data model needed to be changed to comply with this specific storage type.

In the logical data model, the "GEOBOX\_" fields had a *Range Domain* defined; this means that, only values between the ones that were defined in the domain were permitted, in this case these fields could accept values between -180 and 180 to X coordinates, and -90 to 90 to Y coordinates, both with three decimal places. According to the geodatabase rules, the fields that have domains defined are automatically transformed in a data type *Integer*, which means that, the field can only accept *Integer* numbers without decimal places; Figure 29 shows the feature class structure and the respective domains to "GEOBOX\_" fields.

Simple feature class: <b>BASINS</b>						Geometry	Polygon
						Contains M values	No
						Contains Z values	No
Field name	Data type	Allow nulls	Default value	Domain	Precision	Scale	Length
OBJECTID	Object ID						
Shape	Geometry	Yes					
ID	Long integer	No			0		
CODE	String	No					24
NAME	String	No					100
LGE_ID	String	No					2
CTY_ID	String	No					2
ENGLISH	String	No					100
PFAFSTETTER	Long integer	No			0		
GEOBOX_LLX	Long integer	Yes		dGeoBox_LLX	0		
GEOBOX_LLY	Long integer	Yes		dGeoBox_LLY	0		
GEOBOX_URX	Long integer	Yes		dGeoBox_URX	0		
GEOBOX_URY	Long integer	Yes		dGeoBox_URY	0		
DTT_ID	Long integer	Yes			0		
UPDATED_WHEN	Date	No			0	0	8
UPDATED_BY	String	No					15
Shape_Length	Double	Yes			0	0	
Shape_Area	Double	Yes			0	0	

Range domain	
RangeDomain: dGeoBox_URX	
Description	
Field type: Long integer	
Split policy: Default value	
Merge policy: Default value	
Minimum value	Maximum value
-180	180

Range domain	
RangeDomain: dGeoBox_URY	
Description	
Field type: Long integer	
Split policy: Default value	
Merge policy: Default value	
Minimum value	Maximum value
-90	90

Range domain	
RangeDomain: dGeoBox_LLX	
Description	
Field type: Long integer	
Split policy: Default value	
Merge policy: Default value	
Minimum value	Maximum value
-180	180

Range domain	
RangeDomain: dGeoBox_LLY	
Description	
Field type: Long integer	
Split policy: Default value	
Merge policy: Default value	
Minimum value	Maximum value
-90	90

Figure 29| Logical model design of feature class Basins and range domains

Summarising, the use of domains in the geodatabase limits the data types that the fields connected to a domain can receive, and this fact limits the implementation of logical rules. Because of this limitation it was decided not to use domains in these types of fields. The Logical data model was then changed, defining the "GEOBOX\_" fields as *Double* (Precision = 7, Scale = 3, Allowing Nulls).

The change made could then support the data properly. Figure 30 shows the final fields and data types applied to feature class basins. All the feature classes that have fields "GEOBOX\_" where changed to comply with this need of storage.

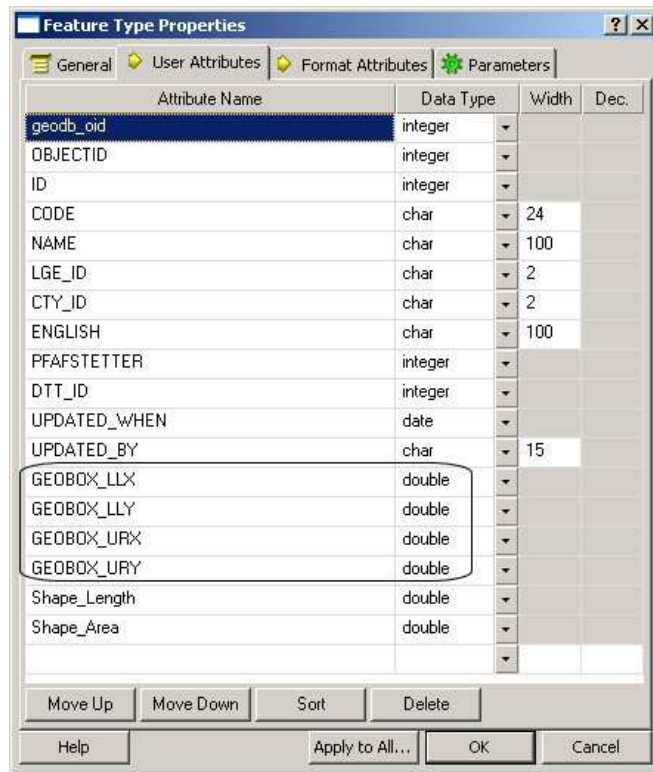


Figure 30| Field data types applied to feature class Basins

Like as referred to in the first screen of the wizard, it is possible to load data into the geodatabase structure from: ESRI shapefiles, coverages, geodatabase feature classes, dBASE files, or other tables from a DBMS. Figure 31 shows the first screen of *Simple Data Loader* wizard.

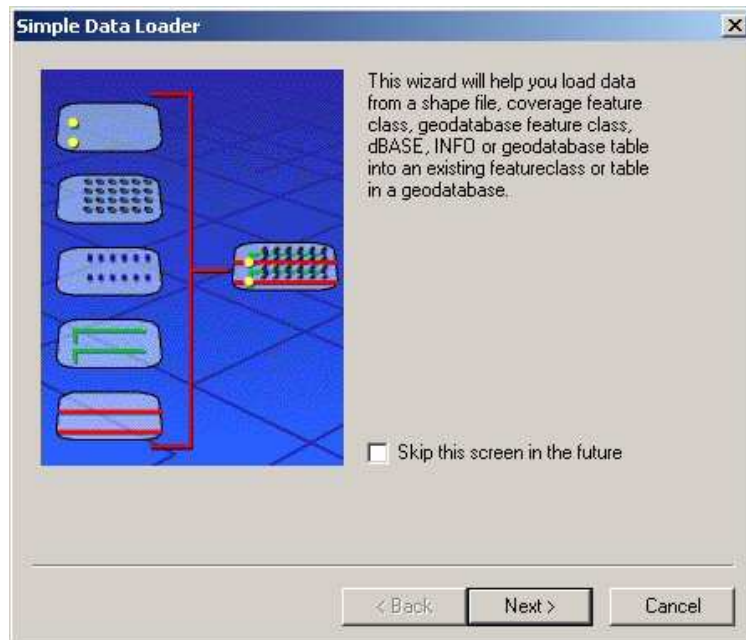


Figure 31| Simple data loader wizard

To test this type of data loading, ESRI *shapefiles* previously exported from the central production database have been used. This exportation was performed to comply with the Esri shapefile limitations, concerning for example, the number of characters available to field names (10 to *shapefiles*), and the conversion from the origin data types to the *shapefile* data types. Figure 32 shows the shapefile creation through FME.

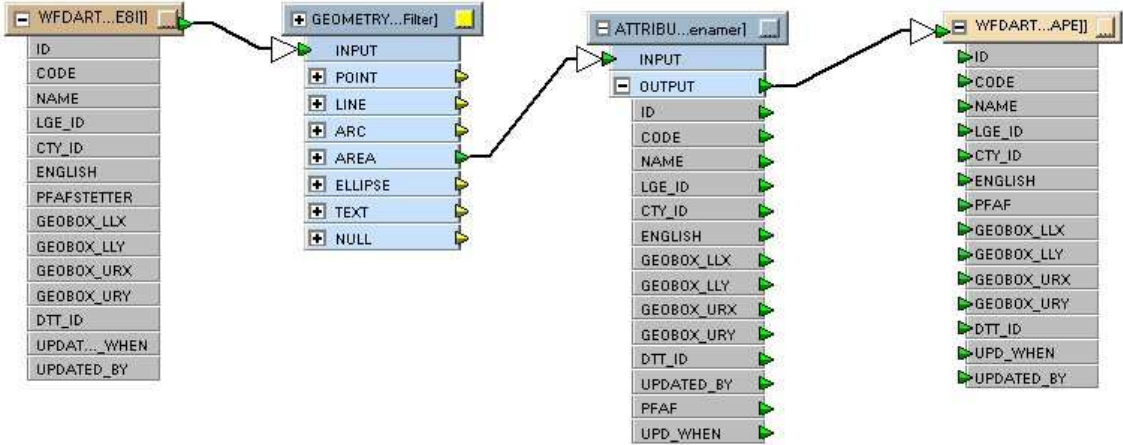


Figure 32| Creation of shapefiles through FME

To assure that fields defined in Oracle as “Number (10)” would generate “Integers” when a *shapefile* is generated, the data types should be changed in the Translator interface (FME). The relation concerning the translation between data types is showed in Table 4.

Table 4| Data type relation between Oracle Spatial and Shapefiles

Oracle	Shapefile
Number (1 to 4)	Short Integer
Number (5 to 9)	Long Integer
Number (10)	Double

According with the table above, the field data types to generate the most suitable shapefile attribute table were defined. Figure 33 shows the data types defined to generate the shapefile attribute table.

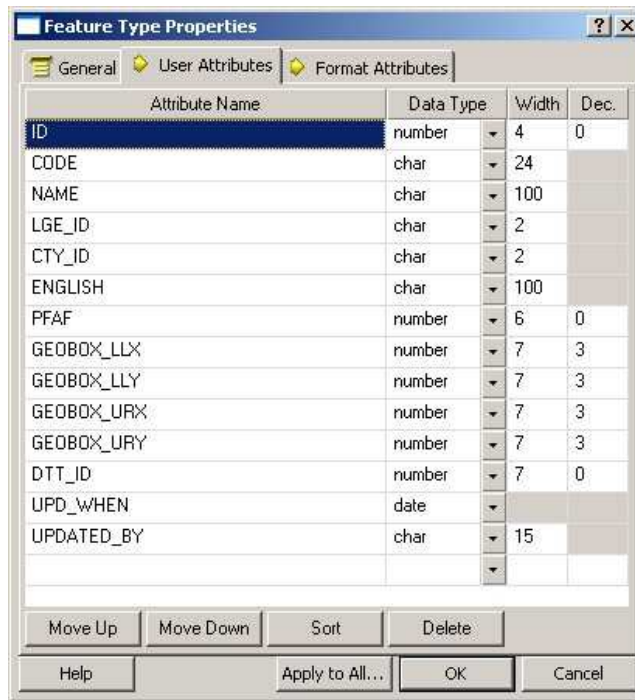


Figure 33| Field data types of shapefiles

The shapefile created could then be loaded into the geodatabase structure using the *Simple Data Loader*. The matching relation from the source *shapefile* fields to the target geodatabase fields can be done through the *Simple Data Loader* interface. Figure 34 shows the matching fields between the source shapefile and the personal geodatabase structured with the WFD-Art3 data model.

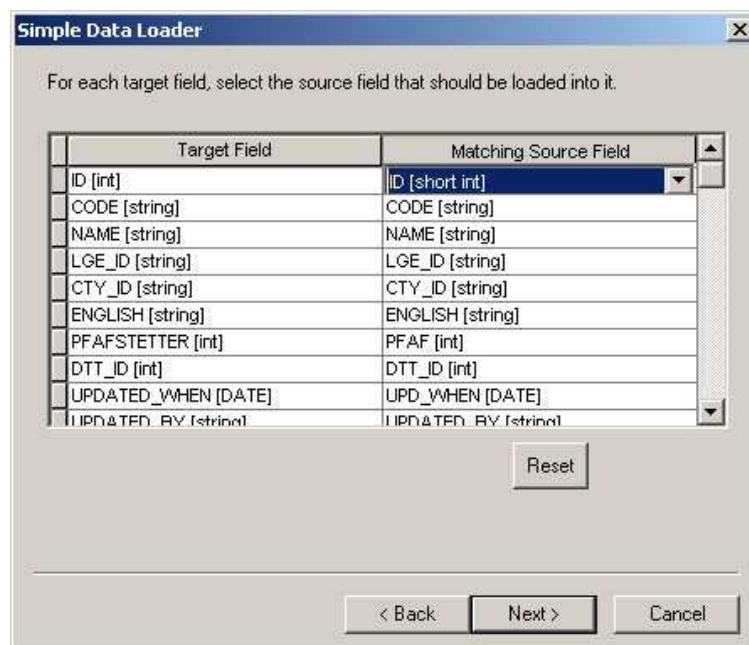


Figure 34| *Simple Data Loader* matrix between shapefile and personal geodatabase

The *Simple Data Loader* should not be used to load data into some types of non-simple feature classes, including:

- Network feature classes;
- Feature classes with feature-linked annotation;
- Origin feature classes in composite relationships.

To load data into this type of features, it is necessary to use the *Object Loader* method through ArcMap.

The *Simple Data Loader* also gives the option of loading all the source data into a subtype of the target and lets the user to specify a query to limit the features you load.

The next section explains how to load data through ArcMap.

### 8.3 Loading data with Object Loader (ArcMap)

The *Object Loader* wizard in ArcMap allows specifying a number of source tables and feature classes, providing a schema match. It also allows specifying which fields in the input data are loaded into which fields of the target feature class or table.

This method is similar to *Simple Data Loader*, the difference is that *Object Loader* can do a validation when the data is loaded into the database. This option evokes the command *validate features* to read all the logical rules described in the data model, and report if there is any errors or warnings. Figure 35 shows the *object loader* interface where is possible to activate a snapping, and a validation of feature on insert.

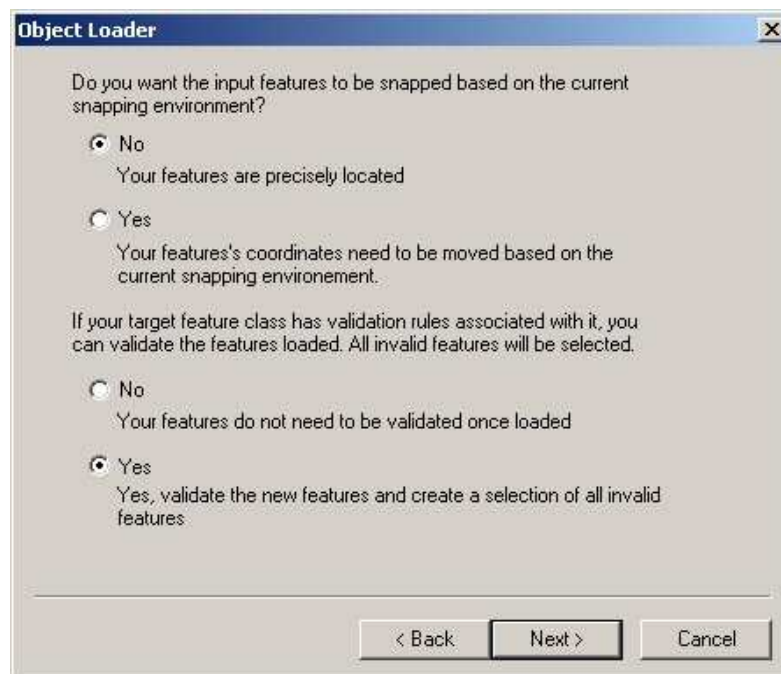


Figure 35| Object Loader validation interface

The validation can also be made after loading data through an editing session in ArcMap. This type of validation is made through the attributes table with the command *Validate Features*, which can be found in the *Editor Toolbar* in ArcMap. Figure 36 illustrates this procedure.



Figure 36| Validate Features command

The elements not compliant with the data model would be highlighted, and a message boxes appears to help identify the problem. Figure 37 shows some of these message boxes.

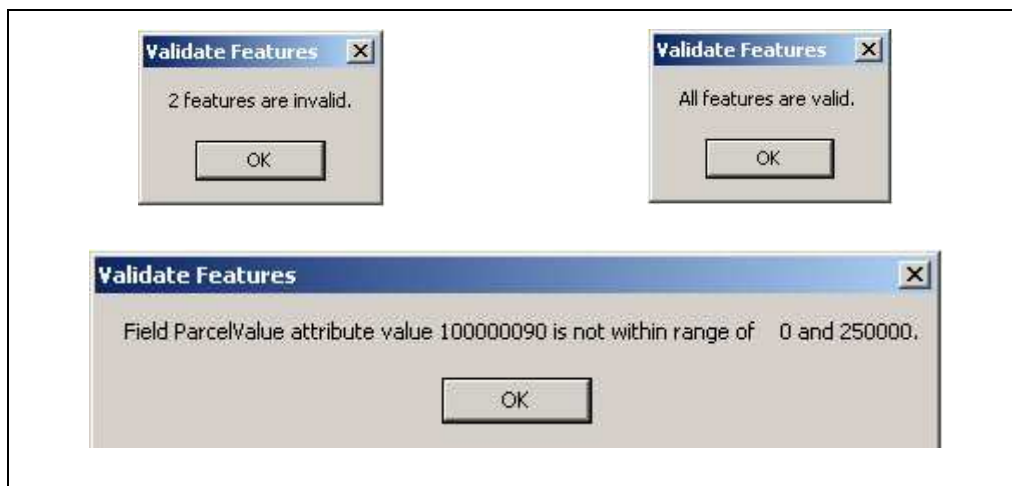


Figure 37| Dialog boxes resulting from the application of the *Validate Features* command

## 8.4 Loading data with FME

FME, which stands for Feature Manipulation Engine, is a software tool used as a universal data translator to convert data from one format to another format according to a preset conversion design. Translators specify how elements in the input format are directly translated and written into the output format.

Data transformation is a more sophisticated and involved process for data translation. With a data transformation process, it is possible to redefine the structure of the input data and to write it to the output file according to a series of custom rules. Essentially, specifying the "mapping" between elements in the input data structure and the output - how the inputs are restructured into the output data structure. Such data transformation is called *semantic data translation*.



For example, data transformation can be used to take a series of individual coordinate and attribute inputs, structure the coordinates into a series of line features, join selected attribute values onto each line feature, and re-classify the attribute values.

The FME Workbench, illustrated in Figure 38, was used to define custom data sources and data transformation procedures. Combining this with the ArcGIS geoprocessing framework helps to leverage and directly use almost any GIS data format.

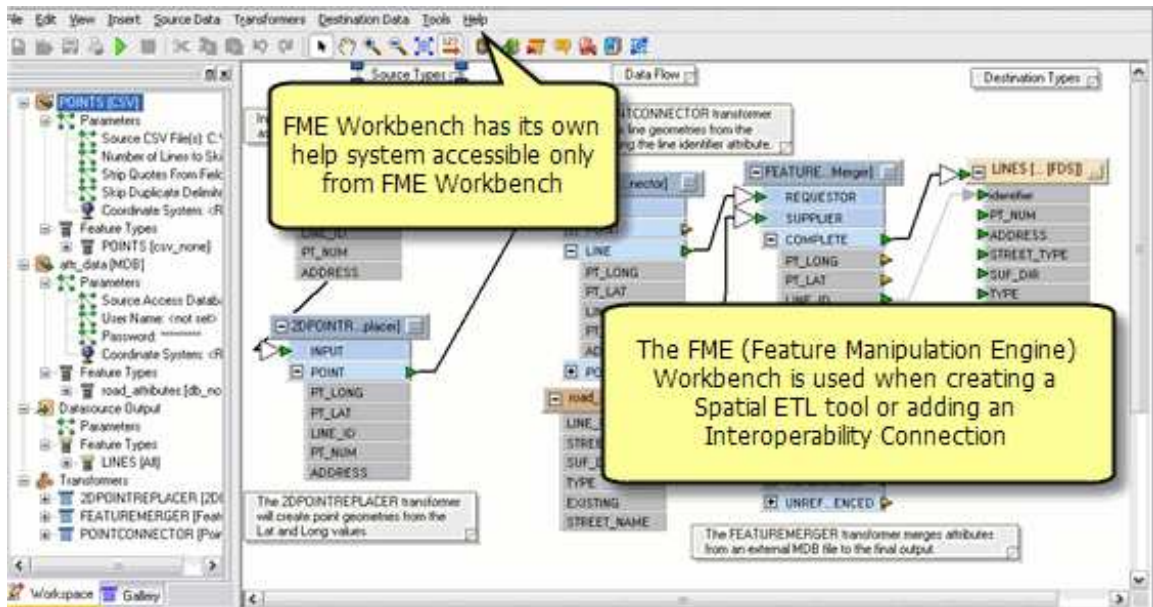


Figure 38| FME Workbench

An ETL (Extract-Transform-Load) process was tested to load data directly into the data structure produced for WFD Article 3 contents. The steps were the following:

- extract information from the format of a data source (Oracle Spatial);
- transform that information into a new data structure (such as to create linear features with attributes or to reclassify an attribute column);
- load the new data into the geodatabase format.

The Figure 39 illustrates this process.



Figure 39| Data transformation using FME

Depending on the FME license available, it is possible to activate the ArcGIS Data Interoperability extension. This extension permits to use the FME within the ArcGIS environment. More details about the data interoperability extension could be found at:

<http://webhelp.esri.com/arcgisdesktop/9.2>, searching by: *Data Interoperability*.

Figure 40 shows an FME diagram to export the feature class “Basins” to several ArcGIS formats (Shapefile, personal geodatabase, file based geodatabase, and SDE geodatabase).

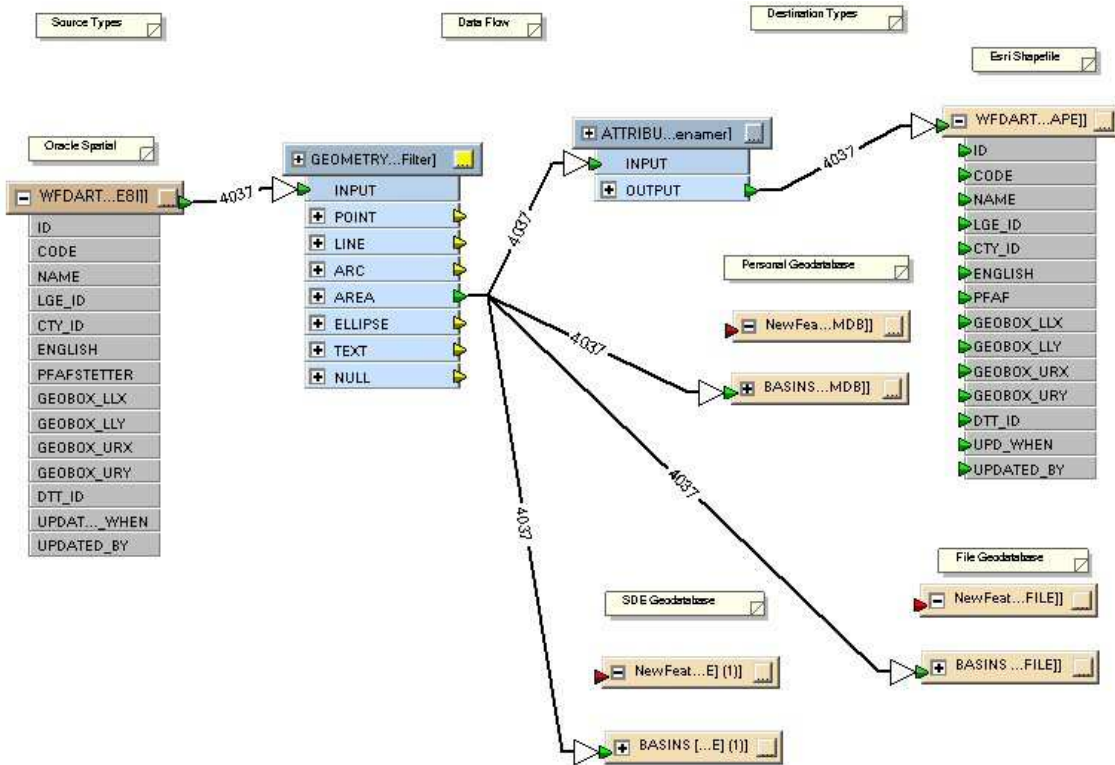


Figure 40| FME conversion from Oracle Spatial to several ArcGIS data formats

## 9 | FINAL REMARKS AND DISCUSSION OF RESULTS

---

The author expects that, with the methodology here presented, the data exchange within member states and the European Commission can be enhanced significantly in comparison with the other options to report and deliver water resources data, namely, the *Shapefile* option and the MS Access tool option used until now to deliver the Article 3 and Article 5 datasets, respectively.

Based on the objectives of the WFD project, and on the analysis of the existing WFD-Art3 database, a conceptual data model and a logical data model were created.

Having created the conceptual data model in MSVisio, using UML class diagrams, it was decided to make use of the proposed methodology to translate the conceptual model to an Esri geodatabase model, creating with this procedure, a logical data model. This logical data model was tested to automatically create a physical database schema in several database platforms.

The adoption of Esri technology has three main justifications: firstly, it provides *out-of-the-box* solutions to model geographic data; secondly, it is used by more than 90% of the member states; and thirdly, GIS technicians are more skilled in this software when compared with other proprietary software or even open-source and free software.

There are also disadvantages in using the Esri software, and one of those is the collection of objects to design the UML class diagrams. This collection of objects is exclusively to be used in MS Visio or Rational Rose software, and there are no objects available to use in open-source or free UML modelling tools. Another important limitation is the lack of possibility of using the Geography Markup Language (GML) within the Esri technology API. As previously referred to, UML is not suitable to describe many spatial behaviours and the usage of GML would be helpful to complement these UML limitations. A way to use GML in Esri software would be welcomed and appreciated.

The geospatial data model proposed is fully compliant with the Water Information System for Europe database (WISE), and in this sense it can be used to store and exchange digital geographic information in several formats, including XML. The proposed geospatial data model complies with the WFD Article 3 datasets and can be extended to comply with Article 5, 6 and 8 datasets.

The advantages in using the geospatial data model, in comparison with the *shapefile* or the MS Access tool are: i) the ease of usage, because the users do not need to learn new software or write any line of code to change it; ii) the database loading process is all performed using ArcGIS; iii) the data validation rules are encapsulated in the model and can be performed after data loading; iv) it is multi-platform applicable (several database management systems can be used); v) the data exchange is available in several formats, including XML. All these advantages make this methodology to design database schemas, an option to be considered and an important way to store, manage and exchange water resources data.

Considering the data model design process, there is an important limitation related to the declarations of data values domains (admissible values in specific table fields). It was not possible to store decimal

values within the fields affected by *range domain* declarations. So, fields dedicated to store decimal values, like, floats or double data types, could not be validated with range domains rules. This fact has limited the application of range domains validation within the data model.

The implementation of the geospatial data model was tested in three different DBMS: MS Access, file-based geodatabase, and Oracle spatial. The MS Access and the file-based geodatabase can be implemented without any additional installation or configuration, but to use Oracle Spatial it was needed to install Oracle DBMS and ArcSDE. Concerning the loading tests, it was possible to load data into MS Access database and file-based geodatabase without any problem of geometric quality of the data loaded, but in the case of Oracle it was not possible to load geographic data with geometric deficiencies. According with the tests performed with these three DBMS, the use of the file-based geodatabase solution is recommended.

- BOOCH, G., JACOBSSON, I. and RUMBAUGH, J (1999) - *UML User Guide*. Addison-Wesley ISBN 0-201-57168-4
- ESRI, (2007) - ArcGIS 9.2 online help (visited: August 2007)
- IBM (2004) - The UML Modelling hierarchy.
- ISO 19103:2005, *Geographic information – Conceptual schema language*
- ISO 19107:2003, *Geographic information - Spatial schema*
- ISO 19108:2002, *Geographic information - Temporal schema*
- ISO 19109:2005, *Geographic information - Rules for application schema*
- ISO 19110:2005, *Geographic information - Methodology for feature cataloguing*
- ISO 19115:2003, *Geographic information - Metadata*
- ISO 19118:2005, *Geographic information - Encoding*
- ISO 19119:2005, *Geographic information - Services*
- ISO 19136:1), *Geographic information - Geography Markup Language (GML)*
- FOWLER, M. *UML Distilled* (2003) - *A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional
- MULLER, R. (2000) - *Database Design for Smarties. Using UML for Data Modeling*. ISBN:1-55860-515-0, 442 p.
- PERENCSIK, A. *et al.*(2004a) - *Designing Geodatabases with Visio*. ESRI.
- PERENCSIK, A. *et al.* (2004b) - *Introduction to CASE Tools*. ESRI.
- QUATRANI, T. (2000) - *Visual Modeling with Rational Rose 2000 and UML*. Addison-Wesley, ISBN 0-201-69961-3.
- Open Geospatial Consortium, Inc (2006) - *OpenGIS GML Encoding Specification*.  
[www.opengeospatial.org/standards/gml](http://www.opengeospatial.org/standards/gml).
- OpenGIS (2004) - *Geography Markup Language (GML) Implementation Specification, GML 3.1.1*. Open Geospatial Consortium, Inc.
- RUGG, R. D., EGENHOFER, M. J. and KUHN, W. (1997) - *Formalizing Behavior of Geographic Features, Geographical Systems*, Vol. 4, No. 2, pp. 159-179. [cited 13 December 2003]. Available from [www.spatial.maine.edu/~max/RJ24.html](http://www.spatial.maine.edu/~max/RJ24.html)
- RUMBAUGH, J., BOOCH, G. and JACOBSSON, I. (1999) - *UML Reference Manual*, Addison-Wesley ISBN 0-201-57168-X

Wikimedia Foundation, Inc. Wikipedia GML Article.

[http://en.wikipedia.org/wiki/Geography Markup Language](http://en.wikipedia.org/wiki/Geography_Markup_Language). (visited: August 2007).

VÖGT, J. (2002) - Guidance Document on Implementing the GIS Elements of the Water Framework Directive;

VISTOS

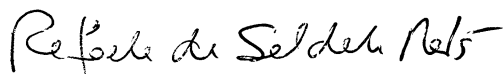


Maria Alzira Santos  
Investigadora Coordenadora  
Chefe do Núcleo de Tecnologias de Informação.

AUTORIA



Nuno Charneca  
Bolsheiro de Doutoramento



Rafaela de Saldanha Matos  
Investigadora Coordenadora  
Directora do Departamento de Hidráulica e Ambiente





**ANNEXES**

---



ANNEX A – Article 3 of the WFD

---



## **Coordination of administrative arrangements within river basin districts**

1. Member States shall identify the individual river basins lying within their national territory and, for the purposes of this Directive, shall assign them to individual river basin districts. Small river basins may be combined with larger river basins or joined with neighbouring small basins to form individual river basin districts where appropriate. Where groundwaters do not fully follow a particular river basin, they shall be identified and assigned to the nearest or most appropriate river basin district. Coastal waters shall be identified and assigned to the nearest or most appropriate river basin district or districts.
2. Member States shall ensure the appropriate administrative arrangements, including the identification of the appropriate competent authority, for the application of the rules of this Directive within each river basin district lying within their territory.
3. Member States shall ensure that a river basin covering the territory of more than one Member State is assigned to an international river basin district. At the request of the Member States involved, the Commission shall act to facilitate the assigning to such international river basin districts. Each Member State shall ensure the appropriate administrative arrangements, including the identification of the appropriate competent authority, for the application of the rules of this Directive within the portion of any international river basin district lying within its territory.
4. Member States shall ensure that the requirements of this Directive for the achievement of the environmental objectives established under Article 4, and in particular all programmes of measures are coordinated for the whole of the river basin district. For international river basin districts the Member States concerned shall together ensure this coordination and may, for this purpose, use existing structures stemming from international agreements. At the request of the Member States involved, the Commission shall act to facilitate the establishment of the programmes of measures.
5. Where a river basin district extends beyond the territory of the Community, the Member State or Member States concerned shall endeavour to establish appropriate coordination with the relevant non-Member States, with the aim of achieving the objectives of this Directive throughout the river basin district. Member States shall ensure the application of the rules of this Directive within their territory.
6. Member States may identify an existing national or international body as competent authority for the purposes of this Directive.
7. Member States shall identify the competent authority by the date mentioned in Article 24.

8. Member States shall provide the Commission with a list of their competent authorities and of the competent authorities of all the international bodies in which they participate at the latest six months after the date mentioned in Article 24. For each competent authority the information set out in Annex I shall be provided.

9. Member States shall inform the Commission of any changes to the information provided according to paragraph 8 within three months of the change coming into effect.

ANNEX B – XML Metadata Interchangeable file (XMI)

---





## **XMI Export Tool: Installation Instructions (Visio 2003, Version 11)**

Install Visio 2003

Install Visio 2003 Professional Edition.

### **Install the ESRI CASE patch**

Copy the file ESRI XMI Export.vsl located under the ArcGIS installation folder ...\\Program Files\\ArcGIS\\CaseTools\\Utilities.

Paste this file to the Visio 2003 installation folder ...\\Program Files\\Microsoft Office\\Visio11\\1033.

Copy ...\\Program Files\\ArcGIS\\CaseTools\\Utilities\\uml.dtd and paste it into the directory to which models will be exported.

### **Install the XMI exporter**

Next, you will download a Microsoft patch for the XMI Export process.

From the Microsoft Downloads page, search using the keyword XMI.

<http://www.microsoft.com/downloads/search.aspx?displaylang=en>

Select the UML to XMI Export download for Visio 2003.

Download the file XMIExpnt.exe and unzip.

Copy and paste the file XMIExpnt.dll to the Visio 2003 installation folder ...\\Program Files\\Microsoft Office\\Visio11\\DLL.

Start Visio 2003 Professional.

Select Tools > Options.

In the Options dialog, select the Advanced Tab and click the File Paths... button.

Select the browse button for Add-Ons field and browse to the Visio 2003 installation folder ...\\Program Files\\Microsoft Office\\Visio11\\1033 and click Select.

Click OK to close the File Paths and Options dialogs.

Select Tools > Macros > Security.

On the Security Level tab, change the level to Low and click OK.

Restart Visio 2003 Professional to have changes take effect.

Install the ESRI Visio 2003 template from the zip file (ArcInfoUMLModel.zip) to ..\\Program Files\\ArcGIS\\CaseTools\\Uml Models

The template is also available from the support site at:

<http://support.esri.com/index.cfm?fa=knowledgebase.documentation.viewDoc&PID=43&MetalD=658>



## ANNEX C – Geodatabase Settings

---



(source: Esri)

## About Setting the X/Y Domain

In order for spatial data to be appropriately stored and referenced to a location on the earth it must have a spatial reference. A spatial reference is comprised of a coordinate system and precision. The coordinate system (geographic or projected) defines the location of the spatial data on the earth. For example, using the GCS\_North\_American\_1983 geographic coordinate system, ESRI is located at -117.195533 longitude and 34.057058 latitude. Precision defines the level of detail that is maintained when data values are stored in the geographic database. For example, if you store the above coordinates and your precision only maintained two decimal places, the values -117.20, 34.06 (rounded) would be stored in the geographic database. If these coordinates are rounded to two decimal places the point would represent an ellipse on the earth's surfaces of 1,109 by 923 meters. Therefore, careful consideration should be given to choosing an appropriate data precision to maintain the precision of your data collection.

For information on choosing an appropriate coordinate system, refer to the ArcGIS Desktop help. Click ArcGIS Desktop Help from the Help pulldown menu in ArcCatalog or ArcMap and click on the index tab. You will find relevant information by typing 'map projections' or 'coordinate systems'. The rest of this topic discusses how to set the geodatabase precision aspect of the spatial reference. The first section discusses the fundamentals of geodatabase precision. The second section discusses different approaches for calculating precision appropriate for your data.

## About Geodatabase Precision

The geodatabase stores coordinates as positive 4-byte integers that have a maximum value of 2,147,483,648. This range of integers is called a spatial domain. It may seem that you are limited to storing one foot or one meter precision with an integer, but that is not the case: you decide what your 4-byte integer units represent. If you need to store meter precision, then you have 2.1 billion meters to work with (approximately 53 times the circumference of the earth). Or you could decide to store centimeters, in which case you would have 2.1 billion centimeters to work with (about half the circumference of the earth). The units that the 4-byte integer represents are called storage units. Storage units are the smallest measurable unit that can be stored for a dataset. The geodatabase converts between storage units and the units of the coordinate system on-the-fly, so you only ever work with decimal numbers even if you are using the lowest level ArcObjects API. The geodatabase uses precision to convert between coordinate system units and storage units using the equation:

Storage Units = Coordinate System Units / Precision.

The following table shows examples of equivalent precision, coordinate system units, and storage units.

Storage units	Coordinate system units	Precision
1 cm	Meters	100
1 mm	Meters	1000
2 cm	Meters	50
1 inch	Feet	12

The geodatabase actually does a little more to convert between storage units and coordinate system units. These coordinates are also shifted during the conversion. You only need to be concerned about this shift if you are manually calculating your spatial domain.

### **Spatial domain extent**

The relationship between the precision and the domain extent (the area you can store) is inversely related. Because you have 2.14 billion integers, there is an outer edge for the spatial domain. As your storage units get smaller (and precision gets larger) the extent of your spatial domain also gets smaller. When you attempt to add features outside of the spatial domain you will get the following error "The coordinates or measures are out of bounds." So it is important that you do not make your storage units so small (and therefore your precision so large) that you will not be able to add features for your entire study area. However, with approximately 2.14 billion storage units to work with, in most cases you can avoid this problem by simply setting the precision appropriately. For example, you can store the entire world with 1 meter storage units but only half the world with 1 cm storage units. Using a decimal degree-based geographic coordinate system like NAD 1983, you could use 1.9 cm storage units for the entire world in a single feature class.

### **Benefits of storing integers**

Performance is the reason the geodatabase uses integer storage instead of floating point storage. Internally, integer coordinates allow the geodatabase to perform spatial operations faster than operations with decimal coordinates. Also, integers can be compressed with relative offsets to consume less storage resources. As the precision becomes larger, the relative offsets between coordinates will get larger, thereby increasing the storage requirements. Only file and ArcSDE geodatabases take advantage of integer compression, so this storage benefit does not apply to personal geodatabases.

### **How to set the spatial domain**

Before you create a spatial domain there are three things to consider.

1. Will the precision of the data maintain the precision of your data collection?
2. Will the spatial domain cover the entire extent of your study area?

3. Is the precision small enough to maximize integer compression (enterprise geodatabases only)?

You don't always need to worry about all of these issues. Many times you can let the default settings generated by the software deal with these issues for you. Below are three different approaches to creating a spatial domain. Choose the one that is most appropriate for your application:

- A. Take the defaults when importing data.
- B. Define a spatial domain by setting the extent and maximizing the precision.
- C. Define the spatial domain by manually calculating your precision and extent.

### **Approach A: Take the defaults when importing data.**

This is the easiest of the approaches. You simply take the default spatial domain generated by the software. Use this approaches if you:

- Have at least one vector dataset or a group of tiled datasets that covers the entire extent of your study area;
- Want the most precision possible within your study area;
- Are creating a raster catalog and all the raster datasets it will contain fall within the extent of your study area.

If you have a dataset that covers the entire study area, import the dataset first and accept the default values for the spatial domain. The defaults will create a domain that encompasses all of the features with a little room to grow. If you have tiled datasets that together cover the entire study area, calculate a spatial domain that encompasses all of the datasets using the Create Spatial Reference tool. Then, create an empty feature class with this spatial domain and load the tiled data into it.

Using this method, the precision will be maximized within the default extent. Because the resulting precision could be very large, this would not be the best approach if you are trying to get the best performance out of an ArcSDE geodatabase. However, this approach will ensure that all of your data will fit inside the spatial domain and that you are using the highest precision possible for your data.

If you are making a raster catalog, and all the raster datasets in your raster catalog are within the default extent, then you can accept the default X/Y domain for the raster catalog. If the raster catalog's contents are outside this extent, the data may not load or display properly.

As you create or import subsequent datasets to the geodatabase use the spatial reference calculated from this original feature class. You can do this by importing the spatial reference from this feature class whenever you create new feature classes or feature datasets. You can also set your geoprocessing settings to use the spatial reference from this feature class.

### **Setting the geoprocessing environment to use a specific spatial reference**

1. In ArcCatalog or ArcMap, from the Tools menu, click Options.
2. Click the Geoprocessing tab.

3. Click the Environments button.
4. Expand General Settings.
5. For Output Spatial Reference, click As Specified Below.
6. Next to the following input box, click the folder icon.
7. On the Coordinate System tab, click Import.
8. Navigate to and select the first feature class that you imported into the geodatabase.
9. Click Add.
10. Click OK to all the open dialogs.

Now all subsequent geoprocessing operations, including importing new data, performed by the current user on this machine, will use this spatial reference.

### **Approach B: Define a spatial domain by setting the extent and maximizing the precision.**

This approach helps you determine an extent for your study area, and then maximizes the precision within that study area. Use this approaches if you:

- Do not have a single vector dataset that covers the extent of your study area but you can define your study area on a map;
- Want the most precision possible within your study area.

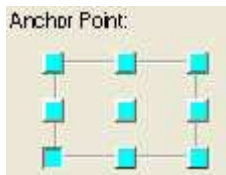
The result of this approach will be exactly the same as Approach A; therefore this approach has the same strengths and weaknesses. Before you can begin you must know the coordinate system that you plan to use. For information on choosing a coordinate system see the Map projections topic in the ArcGIS Desktop Help. If you plan to use the State Plane or UTM coordinate systems, you can find data defining the zone locations at <ArcGIS Installation location>\ArcGIS\Reference Systems in the usstpln83 and utm shapefiles.

### **Determine the extent of your study area**

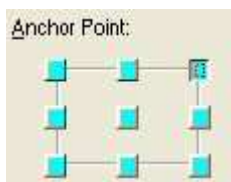
1. Start ArcMap and add reference data for the world or your area of interest. Look for reference data in the following locations.
  - a. ESRI Data and Maps (this is included with ArcGIS)
  - b. <ArcGIS Installation location>\ArcGIS\Metadata\Data
  - c. Geography Network
2. Set the coordinate system of the data frame to the one that you want to use for the new dataset.
  - a. Open the data frame properties;
  - b. Click the Coordinate System tab;



- c. Expand Predefined folder and navigate to the coordinate system that you plan to use;
  - d. Click OK;
3. Zoom in to the part of the world that you plan to use as a study area;
  4. Use the New Rectangle tool on the Draw toolbar to draw on the map a rectangle defining your new study area;
  5. Right-click on that new rectangle and click Properties;
  6. Click the Size and Position tab;
  7. Under Position for Anchor Point, click the lower left box;



8. Copy and paste the coordinates in the X and Y text boxes into a text file. Delete the unit measure at the end of the coordinates. These coordinates are the lower left corner of your study area;
9. Under Position for Anchor Point, click the upper right box;



10. Copy and paste the coordinates in the X and Y text boxes into a text file. Make sure to delete the unit measure at the end of the coordinates. These coordinates are the upper right corner of your study area.

### **Apply the calculated extent when creating a new feature class**

1. In ArcCatalog, navigate to your geodatabase, right-click and click New > Feature Class.
2. For name, type an appropriate name such as StudyArea.
3. Click Next.
4. If necessary, specify a configuration keyword and click Next.
5. In the fields list, click the SHAPE field.
6. In the Field Properties table, click the browse button next to the Spatial Reference property.
7. On the Coordinate System tab, click Select and select your coordinate system.
8. Click the X/Y Domain tab.

9. Copy and paste your coordinates out of the text file into the appropriate text boxes on the X/Y domain tab. Notice that your precision adjusts as you change your extent.
10. Click OK to the Spatial Reference Properties dialog.
11. Click Finish to the New Feature Class wizard.

Now you can import the spatial reference from the StudyArea feature class for all other data that you create in that study area. You can also set your geoprocessing environment so all new data created from geoprocessing operations uses this spatial reference. See Approach A for how to set the geoprocessing environment to use a spatial reference from a feature class.

**Approach C: Define the spatial domain by manually calculating your precision and extent.**

For this approach you calculate the spatial domain parameters manually. Use this approach if you want to optimize performance in an ArcSDE geodatabase.

**Calculate precision**

First you must calculate an appropriate precision by choosing your storage units and calculating your precision accordingly. Set your storage units to be ten times smaller than the best precision of your data collection. This will ensure that the precision of your data collection is maintained in the geodatabase regardless of how you manipulate the data with ArcGIS (geoprocessing, topology cluster tolerance, geometry operations, etc.). Consider the following examples:

<b>Data collection method</b>	<b>Coordinate system units</b>	<b>Equipment precision</b>	<b>Recommended storage unit</b>
Digitize 1:250,000 map	Feet	+/- 416 feet	1 foot
Professional GPS	Meters	+/- 0.5 meter	0.5 meter
Survey with theodolite	Meters	+/- 5 Millimeters	.5 millimeters

Precision is the multiplier that converts coordinate system units into storage units. Mathematically, precision equals one coordinate system unit divided by one storage unit. The following table shows appropriate precision values using the examples above:

<b>Data collection method</b>	<b>Coordinate system units</b>	<b>Recommended storage unit</b>	<b>Precision</b>
Digitize 1:250,000 map	Feet	1 Foot	1
Professional GPS	Meters	.05 meter	20
Survey with theodolite	Meters	.5 millimeter	2000

### **Calculating precision with a geographic coordinate system (GCS)**

Calculating precision based on data that uses a geographic coordinate system is slightly more difficult because angular units (degrees) are not consistent throughout the data. As the latitude changes each degree represents a different amount of area on the ground. If you want to use a linear storage unit with data in a GCS, you will have to perform some calculations. If you calculate an appropriate precision when your angular units are at their largest, you will maintain even more precision in areas where angular units are smaller. For example, if you are maintaining 1m precision where one degree equals one hundred miles on the ground, your geodatabase will maintain 1cm precision where one degree equals one mile on the ground. In a geographic coordinate system, angular units are largest at the equator. Precision will equal the number of storage units in one degree at the equator. As mentioned above, this precision value should be multiplied by ten to account for any ArcGIS processing operations. The following equation can be used:

**Precision = 10 \* GCS Equatorial circumference / 360 / Storage units.**

For example, GCS\_WGS\_1984 has a circumference of 40075016.7 meters. With storage units of 1 meter the equation would look like this:

**Precision = 10 \* 40075016.7 / 360 / 1 meter = 1113195**

Another option is to multiply the semimajor axis of the GCS by the number of radians per angular unit which is the equivalent of the circumference (semimajor axis \* 2) divided by 360.

**Precision = 10 \* Semimajor axis \* radians per unit / Storage units**

You can find this technical information about your GCS by opening its properties in ArcCatalog. If you don't see the Coordinate Systems folder in ArcCatalog, you can make coordinate systems visible from the General tab of ArcCatalog's Tools>Options dialog.

If you are not working with a global dataset, you could calculate your precision based on your lowest latitude. This would allow you to create smaller precision values. However, if your study area ever expanded to lower latitude values, the coordinates stored in those locations would be less precise.

## Check your precision against your study area

To validate that your precision will work given your study area, multiply the greater of the width or height (range) of your study area by the precision. If the result is less than 2,147,483,648, your data can fit inside a spatial domain with your chosen precision.

Even though your data can fit inside a spatial domain, your coordinates may fall outside the coordinate system boundary. Consider the following fictitious dataset with map units of meters.

Given a range of 800,000 (the width) multiplied by a precision of 1,000 equals 800,000,000, which is less than 2.14 billion, therefore, the data will fit. However, the upper right corner of the study area will be 1,000,000,000 x, 4,060,000,000y in the spatial domain (i.e., (1,000,000x) \* 1,000 and (4,060,000y) \* 1,000). Notice that the y value is outside of the 0 to 2.1 billion coordinate system by about 1.9 billion units. To store these coordinates inside the geodatabase, you must shift the spatial domain to surround the data.

## Calculate an appropriate Min x/y

Before you can shift the spatial domain to surround your data, you must identify the center of your spatial domain in map units. The goal is to place your data in the center of spatial domain so your data can expand in all directions if necessary. All of the calculations for shifting the coordinate system are in coordinate system units rather than storage units.

First, find the center of the spatial domain in storage units.

Next, convert the center in storage units to coordinate system units by dividing by the precision. This example uses a precision of 1000.

Now that you have found the center of the spatial domain in coordinate system units, you need to calculate a new minimum x and y of your spatial domain. The formula for calculating the minimum X and Y of your spatial domain is:

**Min X = (DataMinX + DataMaxX)/2 – Domain center in coordinate system units**

**Min Y = (DataMinY + DataMaxY)/2 – Domain center in coordinate system units**

This equation finds the minimum coordinates of your spatial domain to locate the center of your data at the center of the domain. Remember, all of these calculations are in coordinate system units. Examine this equation for the X dimension given the example data:

First find the center of your data.

**(DataMinX + DataMaxX) / 2**

**(200,000 + 1,000,000) / 2 = 600,000**

Next, find the difference between the center of your data and the center of geodatabase space.

**Min X = 600,000 - 1,073,741.824 = -473,741.824**

Because this is a negative number, the spatial domain will shift to the left. Remember, the shift is applied to the spatial domain, not the data. The shift is calculated for both dimensions, so you would need to repeat this process for the Y coordinates.

### **Using your calculated spatial domain in ArcCatalog**

Once you have calculated an appropriate spatial domain, you are ready to create spatial data in the geodatabase. When creating your first dataset, navigate to the X/Y Domain tab of the spatial reference properties and enter the Min X, Min Y, and precision values that you calculated. The maximum X and Y values will be calculated automatically. For all subsequent data that you import or create you can simply import this spatial reference. You can also set your geoprocessing environment so all new data created from geoprocessing operations uses this spatial reference. See Approach A for how to set the geoprocessing environment to use a spatial reference from a feature class.

### **Defining Z and M spatial domains**

The Z and M domains are easier to calculate than X and Y domains. Examine your data, and enter the lowest value for the minimum value and the precision to support its accuracy. You can calculate Z and M precision the same way you calculated precision for the X and Y coordinates. Just like X and Y coordinates, you have 2,147,483,648 storage units to work with. Generally it is not necessary to center the Z and M domains about the data as you can set an absolute minimum based on your data.

When calculating the minimum for a Z domain you could use the lowest point on the earth (-11033 meters in the Mariana trench). Generally M coordinates are positive numbers, so a minimum value of 0 may be appropriate. You may also set the minimum M to have a slight negative offset to account for negative values that could be produced by the extrapolation of measures during operations like Calibrate. These negative values could be corrected later instead of rejected during the extrapolation.

### **About Setting Tolerance**

The **XY tolerance** is an extremely small distance used to resolve inexact intersection locations of coordinates during clustering operations. The XY tolerance is the minimum distance allowed between XY coordinates before they are considered equal. It is used in clustering operations such as topology validation, buffer generation, polygon overlay, and for some editing operations. Another way to think of XY tolerance is that it is the maximum distance a coordinate can move during clustering operations.

The default XY tolerance is the equivalent of 1mm (0.001 meters) in the linear unit of the data's XY (horizontal) coordinate system on the earth surface at the center of the coordinate system. For example, if your coordinate system is recorded in feet, the default value is 0.003281 feet (0.03937 inches). If coordinates are in latitude-longitude, the default XY tolerance is 0.000000556 degrees. If the XY coordinate system is unknown, the XY tolerance value is 0.001 unknown units.

You can normally just accept the default tolerance. This gives good results in most scenarios. There are some situations where you may choose to specify the XY tolerance yourself. You may set a larger tolerance value for data that has less coordinate accuracy, or you may set a smaller value for data that requires extremely high accuracy, such as a survey control network.

When you create a feature class inside a feature dataset, the feature class always uses the XY tolerance of the feature dataset.

You cannot change the XY tolerance of any feature class or a feature dataset after you have created it.

When you create a topology, the cluster tolerance of the topology defaults to the XY tolerance of the feature dataset in which the topology is being created. You can specify a cluster tolerance for a topology that is larger than the XY tolerance, but not one that is smaller than the XY tolerance.

### **How it works**

When processing feature classes using geometry operations such as topology validation, polygon overlay, Buffer, Clip, etc., coordinates whose X distance and Y distance are within the XY tolerance of each other are considered equal (i.e., moved to the same location). Thus, some coordinates are moved to the location of other coordinates or a new location is computed as an average distance between the coordinates in the cluster. In many cases, a weighted average distance is calculated based on the clustered coordinates and their accuracy ranks.

The Clustering process works by moving across the map, identifying clusters of coordinates that fall within the XY tolerance of one another. ArcGIS uses this algorithm to discover, clean up, and manage "shared geometry" between features. This means that the coordinates of the shared geometric elements are co-located (snapped to the same location). This is fundamental to many GIS operations and concepts. The maximum distance a coordinate could move to its new location during this operation is the square root of twice the XY tolerance. The clustering algorithm is iterative. So it is possible in some cases that points can shift more than this distance.

It is important to note that the XY tolerance is not intended to be used to generalize geometry shapes. Instead, it is intended to integrate line work and boundaries during topological operations. That means integrating coordinates that fall within very small distances of one another. Because coordinates can move in both X and in Y by as much as the XY tolerance, many potential problems can be resolved by processing datasets with commands that use the XY tolerance. These include handling of extremely small overshoots or undershoots, automatic sliver removal of duplicate segments, and coordinate thinning along boundary lines.

### **XY tolerance and topology**

When you create a topology in 9.2, the cluster tolerance of the topology defaults to the XY tolerance of the feature dataset in which the topology is being created. You can still specify a cluster tolerance for a topology at 9.2. This allows you to override the XY tolerance of the feature dataset to use a larger tolerance. But you can't specify a cluster tolerance for a topology that is smaller than the XY tolerance specified for the feature dataset.

### **XY tolerance and geoprocessing**

When you perform geoprocessing in 9.2, the geoprocessing functions all default to using the XY tolerance property stored for the data being processed:

- If the output is a new standalone feature class, a new raster catalog or a new feature dataset, the XY tolerance of the input will be used and will be set as the XY tolerance of the output. For tools that have multiple inputs (such as Intersect) the tolerance of the first input specified in the parameter list (i.e. the top most input in a geoprocessing tool dialog) will be used as the default.
- If the output is a feature class inside an existing feature dataset, the XY tolerance of that feature dataset will be used.

You can override the XY tolerance stored in the data so that the geoprocessing functions use an XY tolerance you specify instead. You can specify an XY tolerance in the Environment Settings dialog launched from ArcToolbox so that it applies to all your geoprocessing or in the Environments Settings dialog launched from a tool dialog so it applies to that one tool. When geoprocessing functions they look at the XY tolerance environment and if it has not been set they use the tolerance properties of the data.

### **Working with pre 9.2 geodatabases**

Data in any geodatabase you work with at 9.2 has the new XY tolerance property even if you have not upgraded the geodatabase to 9.2. When you look at the properties of a standalone feature class or feature dataset in a geodatabase that has not been upgraded you'll see a Tolerance tab showing the XY tolerance calculated by the software. However if you create new feature data in a geodatabase that has not been upgraded, the wizard does not give you the tolerance panel and you are not able to specify the XY tolerance manually. If you want to be able to specify the XY tolerance for new data you create in your existing geodatabases, you have to upgrade them to 9.2.

### **Tips**

- To keep coordinate movement small, keep the XY tolerance small. However, an XY tolerance that is too small (such as  $2 * XY$  Resolution or less) may not properly integrate the line work of shared boundaries.
- Conversely, if your XY tolerance is too large, feature coordinates may collapse on one another. This can compromise the accuracy of feature boundary representations.
- Your XY tolerance should never approach your data capture resolution. For example, at a map scale of 1:12,000, one inch equals 1,000 feet, and 1/50 of an inch still equals 20 feet - a data capture accuracy that would be hard to meet during digitizing and scan conversion. You'll want to keep the coordinate movement using the XY tolerance well under these numbers. Remember, the default XY tolerance in this case would be 0.003281 feet.
- In topologies, you can set the coordinate rank of each feature class. You'll want to set the coordinate rank of your most accurate features (e.g., surveyed features) to 1 and of less accurate features to 2, 3, and so on in descending levels of accuracy. This will cause other feature coordinates with a higher accuracy rank number (and therefore, a lower coordinate accuracy) to be adjusted to the more accurate features with a lower rank number.

## **Z tolerance and M tolerance**

ArcGIS 9.2 introduces new properties for specifying and working with M and Z values. M and Z values now have tolerance and resolution properties. These function in the same way as the new XY tolerance and XY resolution properties.

M tolerance and Z tolerance are extremely small values that define the minimum distance between M values and Z values before they are considered equal ('clustered') by spatial processing operations. Coordinates containing M or Z values whose distance from each other are within the M or Z tolerance of each other will be given the same M or Z value.

The default M tolerance 0.001 units. The default Z tolerance is the equivalent of 1 mm in the linear units of the vertical coordinate system used by the data, so if your vertical coordinate system units are meters, the Z tolerance defaults to be 0.001 meters. If no vertical coordinate system is specified, the Z tolerance value is 0.001 Unknown units.

You can normally accept the default M and Z tolerances when you create data contain M and Z values. This gives good results in most scenarios. When you create a new standalone feature class and you specify that the coordinates will contain M or Z values, you'll get prompted to specify the corresponding M or Z tolerance.

When you create a new feature dataset, you are always get prompted to specify the M and Z tolerances that will be used by any feature classes created in that feature dataset for which M or Z values will be stored. (Specifying M or Z tolerances for a feature dataset doesn't mean that the coordinates of the feature classes it contains will always have M or Z values).

For more information about tolerances, resolutions, and other feature class basics, see the ArcGIS Desktop Help topic called 'Feature class basics'. To find that topic, type its name into the Search tab in the Desktop Help.



# ANNEX D – Geodatabase Diagrammer

---



The geodatabase diagrammer is added to ArcCatalog as a command. It invokes Visio to produce diagrams while it inspects the geodatabase. These are not finished schema diagrams, but the building blocks that you can use in Visio to publish your geodatabase data model.

This tool was developed internally for ESRI documentation uses, but it was shared with users to document their geodatabases in a presentation style compatible with ESRI documentation (Michael Zeiler, ESRI, June 4, 2002, updated for ArcScripts release October, 2002).

### **Installation steps:**

1. You must have Visio 5 or better installed. Go to the PutInVisioStencilsFolder and copy GeodatabaseDiagrammer\*.vss and .vst files to your Visio stencils folder. For Visio 5 and Visio 2000, this is likely the c:\Program Files\Visio\Solutions folder. For Visio 2002, it could be *Program Files\Microsoft Office\Visio10\1033\Solutions\Visio Extras* (These are Visio stencil and template files that the command uses for primitives and layout). One set is for TrueType fonts (for general use) and one for PostScript fonts (for publication).
2. Install and register the GeodatabaseDiagrammer.dll.
3. Start ArcCatalog. Using Customize..., add this custom command to a toolbar.
4. Then, select a geodatabase. When a geodatabase is selected, the command will be active. Click it.
5. A dialog appears. If you are publishing documentation, select PostScript fonts. Otherwise, select TrueType fonts. Click Generate Diagram.
6. Watch Visio get launched and create graphics of your feature classes and other geodatabase elements automatically. When done, you'll find a diagram in the same folder as the geodatabase.



## ANNEX E – UML Semantics Checker in ArcCatalog

---



(source: <http://support.esri.com/index.cfm?fa=knowledgebase.techarticles.articleShow&d=26342>, visited at 2 August 2007)

Software: **ArcGIS – ArcEditor:** 8.3, 9.0, 9.1, 9.2 **ArcGIS – ArcInfo:** 8.3, 9.0, 9.1, 9.2

## Summary

Instructions provided describe how to set up ArcCatalog to run the Semantics Checker so an existing geodatabase model may be checked without Microsoft Visio.

The ESRI UML Semantics Checker is a tool used to check the general integrity of XML Metadata Interchange (XMI) files and Microsoft Repositories. This tool is most commonly run inside of Microsoft Visio as part of a geodatabase template macro.

## Procedure

1. Start ArcCatalog.
2. Open Tools > Customize > Commands tab.
3. Select UIControls under Categories.
4. Click New UIControl.
5. Select UI Button Control.
6. Click Create and Edit.
7. When the Visual Basic Editor window opens, under 'Private Sub UIButtonControl1\_Click()', add:  

```
umlsemcheck.SemChecker.StartChecker
```

*UIButtonControl1\_Click* may have a different number than 1.
8. In VB Editor Click Tools > References, and check ESRI UML Semantics Checker.
9. Click Save.
10. Close the Visual Basic Editor.
11. Open Tools > Customize > Commands tab.
12. Select UIControls under Categories.

From Commands, drag the new UIControl onto a toolbar. This button now starts the UML Semantics Checker from ArcCatalog.





## ANNEX F – UML Tools

---



## Non-proprietary UML tools

- Acceleo: Eclipse and EMF template-based system for source-code generation from UML models.
- ArgoUML: a Java-based UML engineering tool, closely follows the UML standard, BSD license.
- Astade: a platform-independent UML-tool based on wxWidgets.
- ATL - a QVT-tool which can transform UML models into other models. Available from the Eclipse GMT project (Generative Modeling Tools).
- BOUML: multi-platform UML 2.0 toolbox, reverse/generates C++/Java/IDL. Very high performances (written in C++, on Qt). Licensed under the GNU GPL.
- Dia: a GTK+/GNOME diagramming tool that also supports UML (licensed under the GNU GPL)
- Eclipse: with Eclipse Modeling Framework (EMF) and UML 2.0 (meta model without GUI) projects.
- Fujaba: Acronym for "From UML to Java and back". Allows modeling behaviour using story diagrams.
- Gaphor: a GTK+/GNOME UML 2.0 modeling environment written in Python
- Kivio: part of the KOffice project
- MetricView Evolution: a tool for metrics-based quality-analysis and better comprehension of UML models
- MonoUML: based on the latest Mono, GTK+ and ExpertCoder.
- NetBeans: with NetBeans IDE 5.5 Enterprise Pack
- Omondo: Eclipse 3.2 plugin. Implements UML2.1, uses JDK 5.
- Papyrus: an open-source UML2 tool based on Eclipse and licensed under the EPL
- StarUML: a UML/MDA platform for Microsoft Windows, licensed under a modified version of GNU GPL, mostly written in Delphi
- Taylor: model-driven architecture "on rails" (licensed under the GNU LGPL)
- Topcased: open source model editors, transformation and formal verification tools, modelling languages ([www.topcased.org](http://www.topcased.org))
- Umbrello UML Modeller: part of KDE
- UML Pad: a UML modeller written in C++/wxWidgets (licensed under the GNU GPL)
- UML Pad (PalmOS): a UML tool for PalmOS

- UMLet: a Java-based UML tool (licensed under the GNU GPL)
- Use Case Maker: a use cases management tool (licensed under the GNU LGPL)
- Violet UML Editor: an easy-to-use Java-based UML Editor; fully integrated into Eclipse; licensed under the GNU GPL
- Xholon: an open-source tool that transforms, simulates and executes models developed using third-party UML 2.0 modelers

### **Proprietary UML tools**

Potential users can freely download versions of most of the following tools; such versions usually impose limits in capability and/or by a time-period.

- AgileJ StructureViews: custom reverse-engineered class-diagrams — Java/Eclipse/XP. (Formerly marketed as "Modelistic".)
- Altova UModel: GUI UML editor, supports UML 2.1, able to export as XML
- Apollo for Eclipse:supports UML 2.0 and Java 5. Integrates with the Eclipse IDE
- ARTiSAN Studio: supports UML 2.0 and SysML
- Blueprint Software Modeler: An integrated software-modeling environment with UML 2.1 modeling, OCL 2.0, meta-modeling and profiles; based on Eclipse
- Borland Together: UML modelling tool, integrated with Eclipse and with MS VS.NET 2005. Supports UML 2.0 and MDA, OCL, MOF.
- Cadifra UML Editor: UML diagram editor for Windows
- ConceptDraw 7: diagramming tool for Windows and Mac, supports UML 2.0
- eRequirements: free web-based requirements-management tool
- GatherSpace: Online/On-Demand Use Case and Requirements Management
- Gliffy: Desktop application feel in a web-based UML diagramming solution
- JUDE: object-oriented analysis and design with UML and Mindmap. JUDE/Community, though free to use, does not provide open source.
- Konesa: Canyon Blue's collaborative modelling tool
- MacA&D: UML and requirements-management for Mac OS X
- MagicDraw UML: UML 2.0 tool with forward- and reverse-engineering and support for many plugin products for MDA. Integrates with many IDEs, including Eclipse and NetBeans. Supports SysML

- MasterCraft (software): a suite of tools from Tata Consultancy Services Limited which support object-oriented analysis and design using UML for development of MDA-based application-software. The tool-suite consists of IDE-based modelers which allow for UML-based modeling.
- Metamill: a round-trip UML tool for C++, C# and Java.
- Microsoft Visio:– a diagramming tool that also supports UML
- Objecteering: provides complete coverage of model-driven development (UML 2.0, MDA)
- OmniGraffle: for Mac OS X.
- OptimalJ: a model-driven development environment for Java.
- Poseidon for UML: commercial version of ArgoUML - supports UML 2.0
- PowerDesigner: by Sybase; supports UML 2.0, data-modeling, business-process modeling - round trip engineering
- Rational Rose: by Rational Software (sold to IBM in 2003); supports UML 1.x.
- Rational Rose XDE: an "eXtended Development Environment" in the tradition of Rational Rose; supports UML 1.x
- Rational Software Architect: Eclipse-based UML 2.0 tool by the Rational Division of IBM
- SDMetrics: a UML-design quality-measurement and design-rule-checking tool
- Select Architect: a UML/MDA platform for Microsoft Windows, running on a scalable repository it integrates with Eclipse and VS.NET
- SmartDraw: UML-diagram tool for Microsoft Windows
- Sparx Enterprise Architect: supports UML 2.1 and SysML
- Telelogic Rhapsody: supports UML 2.0 and SysML for embedded and real-time systems markets
- Telelogic TAU: supports UML 2.0 and SysML
- Use Case Studio: a use-case authoring tool by Rewritten Software. Free for educational use.
- Visustin: reverse-engineers UML activity-diagrams and flow-charts
- Visual Paradigm for UML: supports UML 2.1, data modeling and business modeling WinA&D: UML and requirements management for Microsoft Windows.



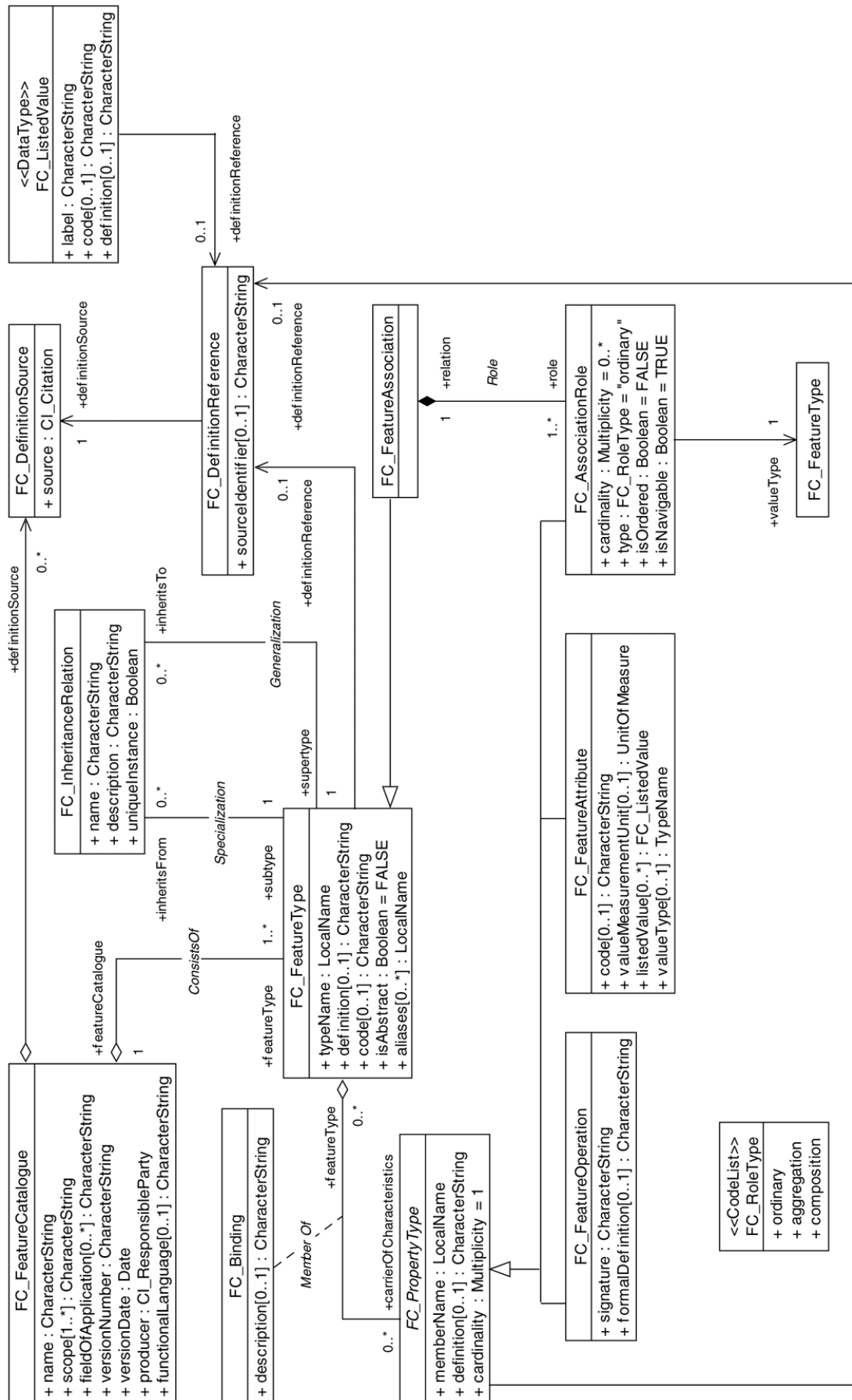
# ANNEX G – Feature Catalogue

---





(Source: ISO 19110:2005(E))



## WFD – Article 3

### Feature Catalogue Example

#### Feature Catalogue

Class FC_FeatureCatalogue (identity = 1)			
Attribute FC_FeatureCatalogue.name	"Digital Geographic Information Exchange Standard (DIGEST) Feature and Attribute Coding Catalogue (FACC)"		
Attribute FC_FeatureCatalogue.scope	"Surface, costal, transitional and groundwater bodies"		
	"Hydrography networks"		
Attribute FC_FeatureCatalogue.fieldOfApplication	"European Legislation application"		
	"River Basin Management Plans"		
Attribute FC_FeatureCatalogue.versionNumber	"1.0"		
Attribute FC_FeatureCatalogue.versionDate	2007-09-14		
Attribute FC_FeatureCatalogue.producer	<b>Class ISO 19115 Metadata:: CI_ResponsibleParty</b>		
	individualName	"Nuno Charneca"	
	organisationName	"Joint Research Centre"	
	contactInfo	<b>Class ISO 19115 Metadata:: CI_Contact</b>	
	phone	<b>Class ISO 19115 Metadata:: CI_Telephone</b>	
	voice	"+39 *** **"	
	address	<b>Class ISO 19115 Metadata:: CI_Address</b>	
	City	Ispra	
	postalCode	21021	
	Country	Italy	

Attribute FC_FeatureCatalogue.functionalLanguage	<b>"English"</b>
Role FC_FeatureCatalogue.featureType	<b>FC_FeatureType</b> (identity = 3)
Role FC_FeatureCatalogue.featureType	<b>FC_FeatureType</b> (identity = 5)
Role FC_FeatureCatalogue.featureType	<b>FC_FeatureType</b> (identity = 7)
Role FC_FeatureCatalogue.definitionSource	<b>FC_DefinitionSource</b> (identity = 2) ( <i>unspecified in this example</i> )

## Definition Source

<b>Class FC_DefinitionSource</b> (identity = 2)				
Attribute FC_DefinitionSource.source	<b>Class ISO 19115 Metadata::CI_Citation</b>			
	Title	"Water Framework Directive – Article 3"		
	Date	<b>Class ISO 19115 Metadata::CI_Date</b>		
		Date	2007	
		dateType	02(publication)	
	<b>Edition</b>	"First"		
	<b>citedResponsibleParty</b>	<b>Class ISO 19115 Metadata::CI_ResponsibleParty</b>		
		organisationName	"Joint Research Centre"	
		Role	11(publisher)	
	<b>otherCitationDetails</b>	"GIS Guidance Document"		

## Feature types and feature attributes

Feature types are classes of real world phenomena with common properties. The example feature catalogue contains many feature types, represented using `FC_FeatureType` (Table B.2). Table C.3 illustrates a 'mine' feature type; it includes a definition, code, and is not an abstract feature type. It also has an alias.

<b>Class <code>FC_FeatureType</code></b> (identity = 3)	
Attribute <code>FC_FeatureType.typeName</code>	"BASINS"
Attribute <code>FC_FeatureType.definition</code>	"The area of land from which all surface run-off flows through a sequence of streams, rivers and, possibly, lakes into the sea at a single river mouth, estuary or delta, in large systems also sub-basins are found, the area of land from which all surface run-off flows through a series of streams, rivers and, possibly, lakes to a particular point in a water course (normally a lake or a river confluence)"
Attribute <code>FC_FeatureType.code</code>	"AA001"
Attribute <code>FC_FeatureType.isAbstract</code>	FALSE
Attribute <code>FC_FeatureType.aliases</code>	"River basin"
Role <code>FC_FeatureType.featureCatalogue</code>	<b><code>FC_FeatureCatalogue</code></b> (identity = 1)
Role <code>FC_FeatureType.carrierOfCharacteristics</code>	<b><code>FC_FeatureAttribute</code></b> (identity = 4)
	<b><code>FC_Binding</code></b> (identity = 6)

The example feature catalogue includes the feature attribute 'pfafstetter'. The following table illustrates its representation using `FC_FeatureAttribute`.

<b>Class <code>FC_FeatureAttribute</code></b> (identity = 4)	
Attribute <code>FC_PropertyType.memberName</code>	"Pfafstetter"
Attribute <code>FC_PropertyType.definition</code>	This system is based upon the topology of the drainage network and the size of the surface area drained. Its numbering scheme is self-replicating, making it possible to provide identification numbers to the level of the smallest sub basins. For a given location it is possible to automatically identify all upstream sub basins, all upstream river reaches, or all downstream reaches.
Attribute <code>FC_PropertyType.cardinality</code>	1
Role <code>FC_PropertyType.featureType</code>	<b><code>FC_FeatureType</code></b> (identity = 3)

	<b>FC_Binding</b> ( <i>unspecified in this example</i> )
Attribute FC_FeatureAttribute.code	"Pfaf"
Attribute FC_FeatureAttribute.valueType	Integer

<b>Class FC_FeatureType</b> (identity = 5)	
Attribute FC_FeatureType.typeName	"RIVERS"
Attribute FC_FeatureType.definition	"A selection from the dataset with surface water bodies, used for general overview purposes"
Attribute FC_FeatureType.code	"BB999"
Attribute FC_FeatureType.isAbstract	FALSE
Attribute FC_FeatureType.aliases	"River"
Role FC_FeatureType.featureCatalogue	<b>FC_FeatureCatalogue</b> (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	<b>FC_FeatureAttribute</b> (identity = 6)
	<b>FC_FeatureAssociationRole</b> (identity = 13)
	<b>FC_Binding</b> ( <i>unspecified in this example</i> )

<b>Class FC_FeatureAttribute</b> (identity = 6)	
Attribute FC_PropertyType.memberName	"Bsn_Id"
Attribute FC_PropertyType.definition	Identification of the hydrographic basin where the river belongs.
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	<b>FC_FeatureType</b> (identity = 5)
	<b>FC_Binding</b> ( <i>unspecified in this example</i> )
Attribute FC_FeatureAttribute.code	"BSN_ID"
Attribute FC_FeatureAttribute.valueType	Integer

<b>Class FC_FeatureType</b> (identity = 7)	
Attribute FC_FeatureType.typeName	"COUNTRIES"
Attribute FC_FeatureType.definition	"Countries of Europe"
Attribute FC_FeatureType.code	"CC001"
Attribute FC_FeatureType.isAbstract	FALSE
Attribute FC_FeatureType.aliases	"Country"
Role FC_FeatureType.featureCatalogue	<b>FC_FeatureCatalogue</b> (identity = 1)
Role FC_FeatureType.carrierOfCharacteristics	<b>FC_FeatureAttribute</b> (identity = 7)
	<b>FC_Binding</b> ( <i>unspecified in this example</i> )

<b>Class FC_FeatureAttribute</b> (identity = 8)	
Attribute FC_PropertyType.memberName	"Member"
Attribute FC_PropertyType.definition	"Classification of European countries concerning it status about the European Union"
Attribute FC_PropertyType.cardinality	1
Role FC_PropertyType.featureType	<b>FC_FeatureType</b> (identity = 5)
	<b>FC_Binding</b> ( <i>unspecified in this example</i> )
Attribute FC_FeatureAttribute.code	"CTY_ID"
Attribute FC_FeatureAttribute.valueType	Integer
Attribute FC_FeatureAttribute.listedValue	FC_ListedValue (identity = 9)
	FC_ListedValue (identity = 10)
	FC_ListedValue (identity = 11)

### Feature attributes listed values

<b>Class FC_ListedValue</b> (identity = 9)	
Attribute FC_ListedValue.label	"Yes"
Attribute FC_ListedValue.code	"1"
Attribute FC_ListedValue.definition	"The country belongs to the European Union"

<b>Class FC_ListedValue</b> (identity = 10)	
Attribute FC_ListedValue.label	"No"
Attribute FC_ListedValue.code	"-1"
Attribute FC_ListedValue.definition	"The country do not belong to the European Union"

<b>Class FC_ListedValue</b> (identity = 11)	
Attribute FC_ListedValue.label	"Z"
Attribute FC_ListedValue.code	"0"
Attribute FC_ListedValue.definition	"The attribute value is missing"

## Feature associations

Feature associations are relationships that link instances of a feature type with instances of the same or of a different feature type. The feature types have specific roles in the association. The example feature catalogue contains the "belongs to" association, relating the two feature types 'Basins' and 'Rivers'. These feature types are illustrated in the following tables.

### Example Feature association 'Belong to'

<b>Class FC_FeatureAssociation</b> (identity = 12)	
Attribute FC_FeatureType.typeName	"Belong to"
Attribute FC_FeatureType.definition	"An object is completely contained by another object."
Attribute FC_FeatureType.code	"101"
Attribute FC_FeatureType.isAbstract	FALSE
Role FC_FeatureType.featureCatalogue	<b>FC_FeatureCatalogue</b> (identity = 1)
Role FC_FeatureAssociation.role	<b>FC_AssociationRole</b> (identity = 13)

The 'belong to' feature association has one association role, represented using FC\_AssociationRole. The 'flow through' role pertains to the 'rivers' feature type. The role cardinality is one or more, and any rivers within the river basin are ordered.

### Example association role 'flow through'

<b>Class FC_AssociationRole</b> (identity = 13)	
Attribute FC_PropertyType.memberName	"flow through"
Attribute FC_PropertyType.definition	"Rivers that flow through the territory of a hydrographic basin"
Role FC_PropertyType.featureType	FC_FeatureType (identity = 3)
Attribute FC_AssociationRole.cardinality	"1..*"
Attribute FC_AssociationRole.type	FC_RoleType.ordinary
Attribute FC_AssociationRole.isOrdered	TRUE
Attribute FC_AssociationRole.isNavigable	TRUE
Role FC_AssociationRole.relation	<b>FC_FeatureAssociation</b> (identity = 12)
Role FC_AssociationRole.valueType	<b>FC_FeatureType</b> (identity = 5)



# ANNEX H – Geodatabase elements

---



## Overview

Generally, feature classes are thematic collections of points, lines, or polygons, but there are seven feature class types:

1. **Points**
2. **Lines.**
3. **Polygons.**
4. **Annotation**— Map text including properties for how the text is rendered..
5. **Dimensions**— A special kind of annotation that shows specific lengths or distances, for example, to indicate the length of a river, or the distance between two features.
6. **Multipoints**— Features that are composed of more than one point. Multipoints are often used to manage arrays of very large point collections such as LiDAR point clusters which can contain literally billions of points. Using a single row for such point geometry is not feasible. Clustering these into multipoint rows enables the geodatabase to handle massive point sets.
7. **Multipatches**— A 3D geometry used to represent the outer surface, or shell, of features that occupy a discrete area or volume in three-dimensional space. Multipatches comprise planar 3D rings and triangles that are used in combination to model a three-dimensional shell. Multipatches can be used to represent anything from simple objects, such as spheres and cubes, or complex objects, such as iso-surfaces and buildings.

Feature classes can be extended to achieve a number of objectives. Here are some of the ways that users extend feature classes using the geodatabase and why.

**Feature Dataset:** Hold a collection of spatially related feature classes or build topologies, networks, and digital elevation models.

**Subtypes:** Manage a set of feature subclasses in a single feature class.

This is often used on feature class tables to manage different behaviors on subsets of the same feature types.

**Attribute Domains:** Specify a list of valid values (coded values) or a range of valid values for attribute columns. The use of domains helps to ensure the integrity of attribute values. Domains are often used to enforce data classifications (such as road class, zoning codes, and land-use classifications).

**Relationship Classes:** Build relationships between feature classes and other tables using a common key. For example, find the related rows in a second table based on rows selected in the feature class and so on.

**Topology:** Model how features share geometry. For example, adjacent counties share a common boundary. Also, county polygons nest within and completely cover states.

**Network dataset:** Model flow connectivity

**Geometric Network:** Model utilities networks and tracing.

**Terrains:** Model triangulated irregular networks (TINs) and manage large lidar and sonar point collections.

**Address Locator:** Geocode addresses.

**Linear Referencing:** Locate events along linear features with measurements.

**Versioning:** Manage a number of key GIS workflows for data management, for example, support long update transactions, historical archives, and multiuser editing.

### **Attribute data types in the geodatabase**

There are a number of supported column types used to hold and manage attributes in the geodatabase. The available column types include a variety of number types, text, date, binary large objects (BLOBs), and globally unique identifiers (GUIDs). The supported attribute column types in the geodatabase include:

- **Numbers.** In one of four numeric data types: short integers, long integers, single-precision floating-point numbers (often referred to as floats), and double-precision floating-point numbers (commonly called doubles)
- **Text:** Any set of alphanumeric characters of a certain length
- **Date:** Holds date and time data
- **BLOBs:** Binary large objects used to store and manage binary information such as symbols and CAD geometries.
- **Global Identifiers:** GlobalID and GUID data types store registry style strings consisting of 36 characters enclosed in curly brackets. These strings uniquely identify a feature or table row within a geodatabase and across geodatabases. These are heavily used to manage relationships especially for data management, versioning, change-only updates, and replication.

Tables provide descriptive information for features, rasters, and traditional attribute tables in the geodatabase. In the geodatabase, there is a focused set of capabilities that are optionally used to extend the capabilities of tables.

# ANNEX I – Data Types

---



## Overview

When creating a table or adding a column to a table in the database, columns are created as a specific data type. Data types are classifications that identify possible values for and operations that can be done on the data as well as the way the data in that column is stored in the database.

When you import data of one type into a column of another data type, you need to understand what the equivalent data types are between ArcSDE and your database management system (DBMS) because it can impact data content. Also, when creating new datasets in ArcGIS, it is helpful to know the equivalent data types between ArcGIS and your DBMS. For example, if you add a floating point (float) column to an existing feature class, that equates to a numeric data type column in a SQL Server database.

NOTE: Moving data from one database to another can cause data types to remap.

Each DBMS product can use different data types. The matrix below compares the data types used by each DBMS for each ArcSDE data type. For example, for the ArcSDE date type, DB2 uses a timestamp data type, Informix uses a datetime data type, Oracle uses a date type, and SQL Server uses datetime or smalldatetime. That means if you move a dataset containing a date field from a DB2 database to an Oracle database. The subsequent sections show DBMS versus ArcGIS data types.

ArcSDE type	Oracle	SQL Server
SE_STRING_TYPE	VARCHAR2(size)	CHAR, VARCHAR
SE_NSTRING_TYPE	NVARCHAR2(size)	NCHAR, NVARCHAR
SE_NCLOB_TYPE	NCLOB	NTEXT
SE_INT16_TYPE (SE_SMALLINT_TYPE)	NUMBER(n) n can be in the range of 1 to 5.	Smallint, tinyint, numeric (precision < = 4, scale = 0)
SE_INT32_TYPE (SE_INTEGER_TYPE)	NUMBER(n) n can be in the range of 5 to 10; however, if created with the sdetable -o create operation o, int32 results in NUMBER(38).	Int, numeric (precision < = 9, scale = 0)
SE_INT64_TYPE	NUMBER(n) n can be in the range of 10 to 38. Note: The server configuration parameter INT64TYPES must be TRUE to create columns with this data type.	Bigint, numeric (precision < 19, scale = 0) Disabled by default; not supported by ArcGIS; enabled with sdeconfig; unsupported with SQL Server 7.0
SE_FLOAT32_TYPE (SE_FLOAT_TYPE)	NUMBER(n,m) n can be in the range of 1 to 7. m is 127 or less.	Float, numeric (precision < 7, scale > 0)
SE_FLOAT64_TYPE (SE_DOUBLE_TYPE)	NUMBER(n,m) n can be in the range of 7 to 38. m is 127 or less.	Real, money, smallmoney, numeric (precision > = 7, scale > 0)
SE_DATE_TYPE	DATE	Datetime, smalldatetime
SE_UUID_TYPE*	NCHAR(38)	Uniqueidentifier
SE_BLOB_TYPE	LONG RAW BLOB	Image, binary, varbinary
SE_SHAPE_TYPE	NUMBER(38) SDO_GEOMETRY ST_GEOMETRY Oracle data type depends on the geometry storage chosen for the layer: compressed binary or well-known binary = NUMBER(38) Oracle Spatial = SDO_GEOMETRY Spatial Type = ST_GEOMETRY	

\*The unique identifier field will be created as NCHAR in Oracle and SQL Server if the configuration keyword with which you specified the tables creation had the parameter UNICODE\_STRING set to TRUE.

## Oracle data types

### ArcGIS to Oracle data type mapping

When creating a feature class or table in ArcGIS, there are 11 different data types available for each column. These types are mapped to Oracle types in the table below.

ArcGIS data type	Oracle data type	Notes
OBJECTID	NUMBER(38)	NOT NULL
SHORT INTEGER	NUMBER(4)	
LONG INTEGER	NUMBER(38)	
FLOAT	NUMBER(38,8)	
DOUBLE	NUMBER(38,8)	
TEXT	VARCHAR2(50)	
DATE	DATE	
BLOB	BLOB	
GUID	CHAR(38)	
GEOMETRY	ST_GEOMETRY* NUMBER(38) SDO_GEOMETRY	Oracle data type depends on the geometry storage specified for the layer.
RASTER	BLOB LONG_RAW SDO_GEORASTER	Oracle data type depends on the raster storage specified in the DBTUNE table.



# ANNEX J – Geodatabase UML Static Structure Objects

---





Package shape

A package is a grouping of model elements. You can imagine an entire system description as a package with all the system elements in it, including other packages, models, diagrams, and elements.

A package is the basic organizing element of a UML model. Each element belongs to only one package, and one package can be nested in another. You can create new packages or add elements or views (diagrams) to packages by right-clicking icons in the tree view.

Dropping a Package shape onto a drawing page automatically creates a new static structure diagram on a new page. (Double-click the new static structure element in the tree view to go to that page.) To create a view, you can drag some of the elements that belong to the package onto the new page.



Class shape

In a static structure diagram, a class describes a set of objects with similar structure, behavior, and relationships. Classes are declared in class (static structure) diagrams and represent concepts in the systems being modeled. The name of a class must be unique within its package.



Generalization shape

In a static structure diagram, a generalization is a relationship between a specific element and a general element, such that the specific element is fully consistent with the general element and includes additional information (such as attributes and associations). For example, the classes Polygon, Ellipse, and Spline can all be specific elements of a more general abstract class element named Shape.

To indicate a generalization, use a solid line with a hollow arrow at the end pointing toward the more general element. You can add a discriminator text label to a generalization path.

Generalization is most often used with classes, use cases, and packages, but can also be used with other UML elements.



Binary Association shape

In a static structure diagram, a binary association is a relationship between exactly two classes. You can add name and stereotype properties to a binary association. The point where a binary association connects to a class is called an association end or role. Properties related to a specific role, such as end name, multiplicity, aggregation, and navigability are attached to an association end.



Composition shape

A composition is a form of aggregation that indicates that a part may belong to only one whole and that the lifetime of the whole determines the lifetime of the part.

A composition is indicated with a solid filled diamond adornment on one of the association ends, and can be thought of as a collaboration in which all of the participants are part of a single composite object.

## ANNEX L – Tagged Values

---



## Overview

In the logical data model developed were used tagged values to set additional properties of UML elements, such as classes, attributes, and associations (Perencsik *et al.*, 2004b), for example, to set the length (in characters), of a string field or an origin class of a relationship. Table 5 shows the tagged values which were used in the WFD Article3 logical data model. The next table presents the tagged values used in the WFD Article3 Logical Data Model

Table 5| Tags to geodatabase UML classes

Category	Tagged value name	Values	Meaning
<b>Feature Class</b>	GeometryType	esriGeometryPoint esriGeometryPolygon esriGeometryPolyline	Sets the geometry type for feature classes
	HasM	True/False	Indicates whether a feature class contains M-values
	HasZ	True/False	Indicates whether a feature class contains Z-values
<b>Fields</b>	Precision	Integer	Sets the number of digits for integer or double type fields
	Scale	Integer	Sets the number of decimal places in single and double type fields
	Length	Integer	Sets the width of string type fields
	AllowNulls	True/False	Indicates whether null values are allowed for a field
<b>Relationship Class</b>	OriginClass	Class name	Defines the origin class for a relationship class
	OriginPrimaryKey	Field name	Sets the primary key field of the origin class
	OriginForeignKey	Field name	Sets the foreign key field of the origin class
	DestinationPrimaryKey	Field name	Sets the primary key field of the destination class
	DestinationForeignKey	Field name	Sets the foreign key field of the destination class

## Domains

The data types of the CDM attributes were replaced by the corresponding ESRI field types. The data types defining specific value domains were replaced by classes, stereotyped *CodedValueDomain* for coded value domains and stereotyped *RangeDomain* for range domains, using the UML class properties.

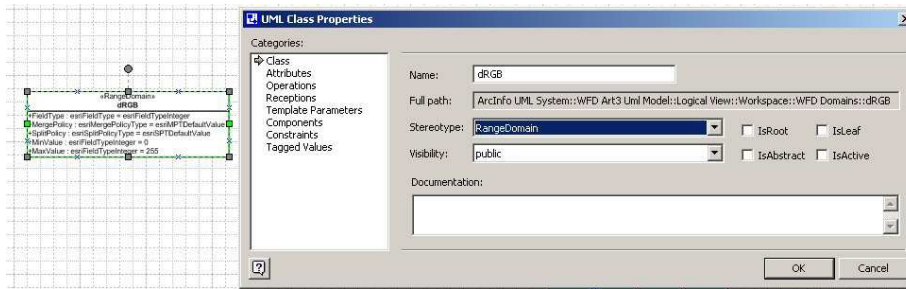


Figure 41| Range Domain definition

Domains can be used as the type of a field to define the type and valid values for the field. The first three attributes in the class define:

- Field type;
- Merge policy;
- Split policy.

The type for any of these attributes is not important and can be left unspecified. The initial value, however, is the actual setting. For example, the following is a valid *MergePolicy*:

- • Attribute name: MergePolicy;
- • Attribute type: <Blank>;
- • Attribute Initial Value: *esriMPTDefaultValue*.

The valid initial values for *FieldType*, *MergePolicy*, and *SplitPolicy* are taken from the *esriFieldType*, *esriMergePolicy*, and *esriSplitPolicy* enumerations, respectively. In addition to the standard attributes of domains (*FieldType*, *MergePolicy*, and *SplitPolicy*), range domains have *MinValue* and *MaxValue*. The initial values in *MinValue* and *MaxValue* define the actual range. The type of these attributes is not important and can be left unspecified.

**Coded value domains** can have any number of UML attributes representing the set of permissible values. The Figure 42 shows the editing of an attribute table with four coded values applied to the field *AreaType*.



OBJECTID *	Shape *	ID	AreaType	LGE	DTT_ID	CTY_ID	Shape_Length	Shape_Area
1	Polygon	<Null>	Agricultural	<Null>	400	100000	422.588581	10825.276424
2	Polygon	<Null>	Agricultural	<Null>	400	100000	499.032184	15900.046197
3	Polygon	<Null>	Industrial	<Null>	400	100000	303.782574	5161.00524
4	Polygon	<Null>	Commercial	<Null>	400	100000	264.339311	3604.046514
5	Polygon	<Null>	Industrial	<Null>	400	10000090	316.675896	6104.54553
6	Polygon	<Null>	Residential	<Null>	400	100000	446.578362	11990.595995
8	Polygon	<Null>	Residential	<Null>	400	100000	0	0
9	Polygon	<Null>	Residential	<Null>	400	100000	0	0
7	Polygon	<Null>	Residential	<Null>	400	100000	<Null>	<Null>

Record: 9 Selected Records (1 out of 9 Selected) Options

Figure 42| The edit of an attribute table with coded values

The initial value defines the valid code, and the name of the attribute is the name of the code. The type of these attributes is not important and can be left unspecified. By convention, the name of the classes that define the domains of the attributes is composed by “d” plus the name of the attribute. For example, Figure 43 shows the definition of the coded value domain for the attribute *MEMBER* of the feature class *COUNTRIES*.

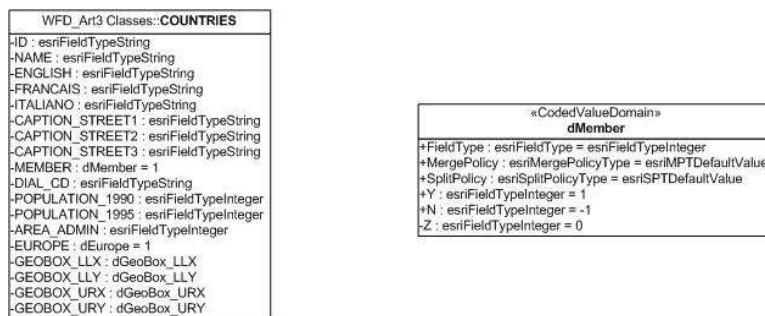


Figure 43| Definition of a coded values domain

The attribute *MEMBER* should be defined applying the domain to its data type, like Figure 44 shows.

Attribute	Type	Visibility	Multiplicity	Init. Value
ITALIANO	ESRI Types::esriFieldTypeString	private	1	
CAPTION_STREET1	ESRI Types::esriFieldTypeString	private	1	
CAPTION_STREET2	ESRI Types::esriFieldTypeString	private	1	
CAPTION_STREET3	ESRI Types::esriFieldTypeString	private	1	
MEMBER	WFD Domains:dMember	private	1	1
DIAL_CD	ESRI Types::esriFieldTypeString	private	1	
POPULATION_1990	ESRI Types::esriFieldTypeInteger	private	1	
POPULATION_1995	ESRI Types::esriFieldTypeInteger	private	1	
AREA_ADMIN	ESRI Types::esriFieldTypeInteger	private	1	

Categories: Class, Attributes, Operations, Receptions, Template Parameters, Components, Constraints, Tagged Values

Buttons: New, Duplicate, Delete, Move Up, Move Down, Properties...

OK Cancel

Figure 44| Data type definition with a domain

The Figure 45 shows some examples of range and coded values domains declared for the WFD-Art3 data model.

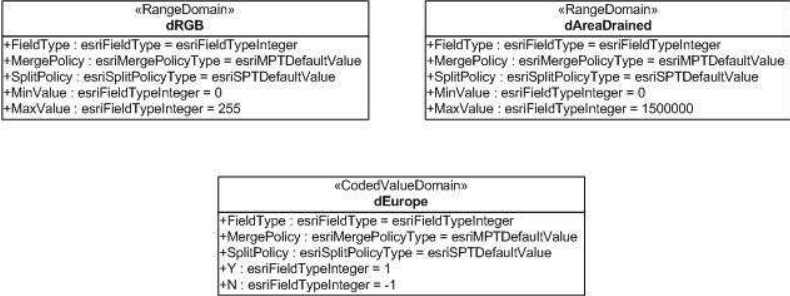


Figure 45| Samples of coded values and range domains

To facilitate the creation of domains, the ArcInfo UML Model has template domains that can be cloned and modified:

- TemplateCodedValueDomain;
- TemplateRange Domains.

**Domains in an Oracle DBMS**

In the Oracle database, domains are stored in the system tables GDB\_DOMAINS. Coded domains are tracked in the GDB\_CODEDDOMAINS system table, and range domains are tracked in the GDB\_RANGEDOMAINS system table.

The GDB\_FIELDINFO system table stores one record for every field in every user-defined table in the geodatabase. This table also stores the name of the domain (if any) that is associated with each field.

The GDB\_ATTRRULES table stores the name of a field, the name of the domain associated with that field, the subtype code for the feature class, and the rule ID.

The Figure 46 illustrates the domain system tables with their implicit relationships.

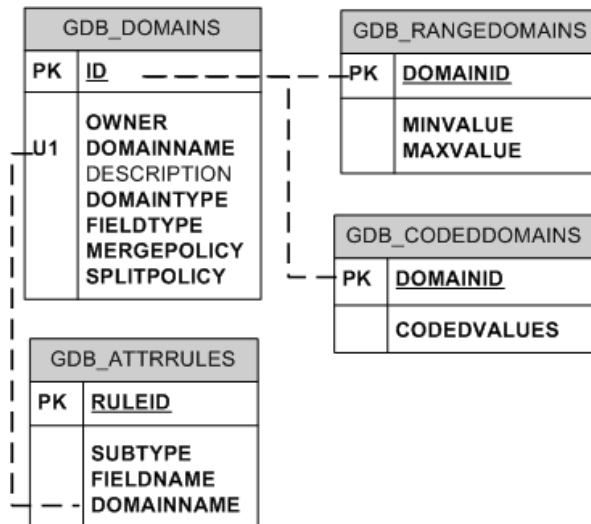


Figure 46| Domains system tables in Oracle DBMS (Source: Esri)

### IsAbstract Class properties

An abstract class in a UML model does not have a direct representation in the geodatabase. It is a modelling artifact, used to model fields that are common to many tables or feature classes in a single place. When the model is used to create a geodatabase, the fields in parent or abstract classes are made part of the derived tables or feature classes.

For abstract types the *IsAbstract* property was checked to prevent from being implemented later on as tables or feature classes in the database.

In the WFD-Art3 data model a class, checked as *IsAbstract*, was produced for the fields “GEOBOX\_” and “UPDATED\_”. The properties of that class are illustrated in the Figure 47.

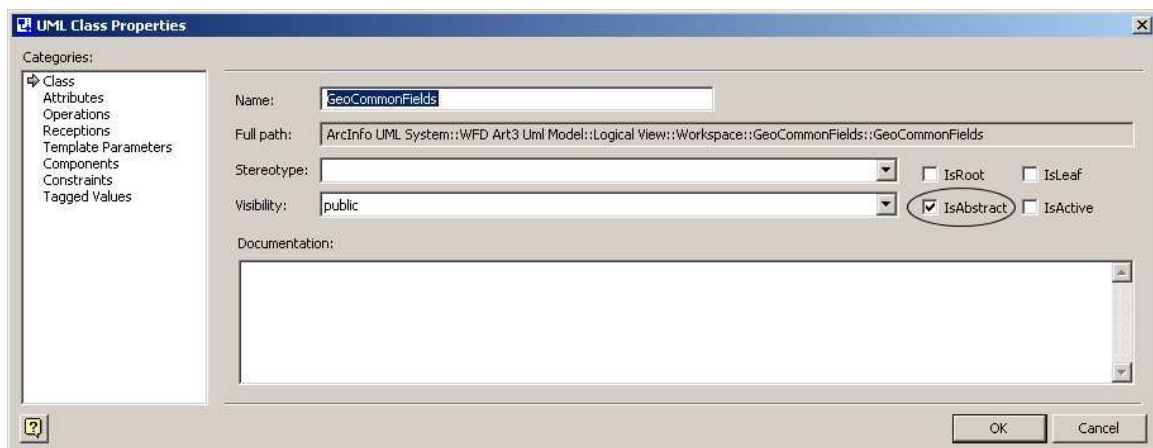


Figure 47| UML Class properties of an abstract class

Almost all feature classes have four fields with the same data type. The generalization relationship was used to model this inheritance, as showed in Figure 48. With the application of this technique, the modeler doesn't need to declare the same attribute in all classes where it appears, because it can be simply inherited from an abstract class.

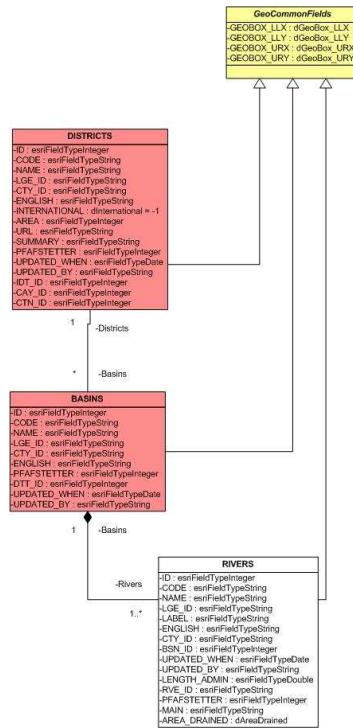


Figure 48| The use of IsAbstract Class

## ANNEX M – CD-ROM Contents

---



The CD-Rom attached to this document contains the following directories:

1. Data Model (MS Visio files and XMI files);
2. ArcInfo UML Model Template;
3. Change XML Spatial Reference;
4. GeoDB Diagram Designer;
5. UML Semantic Checker;
6. Visio XMI Exporter;
7. Esri documentation;
8. Presentation.

The **Data Model** directory contains tree files in MS Visio format (.vsd). The file "**WFD\_Art3\_ClassDiagram.vsd**" has the class diagram of the geodatabase model, where is possible to see in detail the database structure. The file "**WFD\_Art3\_Geodatabase\_Model.vsd**" it's the Esri geodatabase model, where all the classes were declared and from where the XMI file is exported. The "**WFD\_Art3\_XMI.xml**" it's the XMI file that encapsulates the geodatabase schema.

The **ArcInfo UML Model Template** directory has the objects collection to design geodatabase models with MS Visio and Rational Rose.

The **Change XML Spatial Reference** directory groups the files to change the geographic spatial reference of a XMI file.

The **GeoDB Diagram Designer** directory contains all the files needed to add to ArcGIS interface the functionality to automatically construct geodatabase model diagrams.

The **UML Semantic Checker** directory contains all the files needed to install the tool to analyze XMI files that result from a MS Visio export.

The **ESRI Docs** directory contains all files that explain the geodatabase model construction.

The **Presentation** directory contains a slide presentation that summarizes the methodology applied in the geospatial model construction.





## ANNEX N – Terms, definitions and abbreviations

---



For the purposes of this document, the following terms and definitions were applied.

### **International Standardization Organization (ISO)/Technical Specification Terms**

**Application:** manipulation and processing of data in support of user requirements [ISO 19101]

**Application Schema:** conceptual schema for data required by one or more applications [ISO 19101]

**Conceptual Model:** model that defines concepts of a universe of discourse [ISO 19101]

**Conceptual Schema:** formal description of a conceptual model [ISO 19101]

**Data Type:** specification of a value domain with operations allowed on values in this domain.

EXAMPLE: Integer, Real, Boolean, String, Date and SG Point (conversion of data into a series of codes).

NOTE Data types include primitive predefined types and user-definable types.

**Domain:** well-defined set. NOTE Domains are used to define the domain set and range set of attributes, operators and functions.

**Feature:** abstraction of real world phenomena [ISO 19101]

NOTE: A feature may occur as a type or an instance. Feature type or feature instance should be used when only one is meant. NOTE 2 In UML, a feature is a property, such as operation or attribute, which is encapsulated as part of a list within a classifier, such as interface, class or data type (Rugg *et al.*, 1997).

**Feature Association:** relationship that links instances of one feature type with instances of the same or a different feature type [ISO 19109]

NOTE: A feature association may occur as a type or an instance. Feature association type or feature association instance is used when only one is meant. NOTE 2 Feature associations include aggregation of features.

**Feature Attribute:** characteristic of a feature [ISO 19101].

NOTE: A feature attribute has a name, a data type, and a value domain associated to it. A feature attribute for a feature instance also has an attribute value taken from the value domain. NOTE 2 A feature attribute may occur as a type or an instance. Feature attribute type or feature attribute instance should be used when only one is meant.

**Feature Catalogue:** catalogue containing definitions and descriptions of the **feature** types, **feature attributes**, and **feature associations** occurring in one or more sets of geographic data, together with any feature operations that may be applied

**Feature Operation:** operation that every instance of a feature type may perform [ISO 19110]

EXAMPLE 1: An operation upon a “dam” is to raise the dam. The result of this operation is to raise the level of water in a reservoir. EXAMPLE 2 An operation by a “dam” might be to block vessels from navigating along a “watercourse”.

NOTE Feature operations provide a basis for feature type definition.

**Metadata:** data about data [ISO 19115]

**Metadata Element:** discrete unit of metadata [ISO 19115]

NOTE 1: Metadata elements are unique within a metadata entity. NOTE 2 Equivalent to an attribute in UML terminology.

**Schema:** formal description of a model [ISO 19101]

**Service:** distinct part of the functionality that is provided by an entity through interfaces [ISO/IEC TR 14252]

**Value Domain:** set of accepted values

EXAMPLE: The range 3-28, all integers, any ASCII character, enumeration of all accepted values (green, blue, white).

## Uml Terms

The following are UML terms that are adapted from ISO/IEC 19501.

**Actor:** coherent set of roles that users of use cases play when interacting with these use cases

NOTE: An actor may be considered to play a separate role with regard to each use case with which it communicates.

**Aggregation:** special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part

NOTE See composition.

**Association:** semantic relationship between two or more classifiers that specifies connections among their instances

NOTE: A binary association is an association among exactly two classifiers (including the possibility of an association from a classifier to itself).

**Attribute:** feature within a classifier that describes a range of values that instances of the classifier may hold

NOTE: An attribute is semantically equivalent to a composition association; however, the intent and usage is normally different. NOTE 2 "Feature" used in this definition is the UML meaning of the term and is not meant as defined in 4.1 of this Technical Specification.

**Behaviour:** observable effects of an operation or event, including its results

**Cardinality:** number of elements in a set

NOTE: Contrast: multiplicity.

**Class:** description of a set of objects that share the same attributes, operations, methods, relationships and semantics.

**NOTE:** A class may use a set of interfaces to specify collections of operations it provides to its environment. See: interface.

**Classifier:** mechanism that describes behavioural and structural features

**NOTE:** Classifiers include interfaces, classes, data types, and components.

**Component:** modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces.

**NOTE:** A component represents a physical piece of implementation of a system, including software code (source, binary or executable) or equivalents such as scripts or command files.

**Composition:** form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts.

**NOTE:** Parts with non-fixed multiplicity may be created after the composite itself, but once created they live and die with it (i.e. they share lifetimes). Such parts can also be explicitly removed before the death of the composite. Composition may be recursive. Synonym: composite aggregation.

**Constraint:** semantic condition or restriction

**NOTE:** Certain constraints are predefined in the UML, others may be user defined. Constraints are one of three extensibility mechanisms in UML. See: tagged value, stereotype.

**Dependency:** relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element)

**Generalization:** taxonomic relationship between a more general element and a more specific element that is fully consistent with the more general element and contains additional information

**NOTE:** An instance of the more specific element may be used where the more general element is allowed. See: inheritance.

**Inheritance:** mechanism by which more specific elements incorporate structure and behaviour of more general elements related by behaviour

**NOTE:** See generalization.

**Instance:** entity that has unique identity, a set of operations can be applied to it, and state that stores the effects of the operations

**NOTE:** See object.

**Interface:** named set of operations that characterize the behaviour of an element

**Metamodel:** model that defines the language for expressing a model

**Method:** implementation of an operation

NOTE: It specifies the algorithm or procedure associated with an operation.

**Multiplicity:** specification of the range of allowable cardinalities that a set may assume

NOTE: Multiplicity specifications may be given for roles within associations, parts within composites, repetitions and other purposes. Essentially a multiplicity is a (possibly infinite) subset of the non-negative integers. Contrast: cardinality.

**Object:** entity with a well-defined boundary and identity that encapsulates state and behaviour.

NOTE: State is represented by attributes and relationships, behaviour is represented by operations, methods and state machines. An object is an instance of a class. See: class, instance.

**Operation:** service that can be requested from an object to affect behaviour

NOTE 1: An operation has a signature, which may restrict the actual parameters that are possible.

NOTE 2: Definition from UML Reference Manual: A specification of a transformation or query that an object may be called to execute. NOTE 3 An operation has a name and a list of parameters. A method is a procedure that implements an operation. It has an algorithm or procedure description.

**Package:** general purpose mechanism for organizing elements into groups

NOTE: Packages may be nested within other packages. Both model elements and diagrams may appear in a package.

**Refinement:** relationship that represents a fuller specification of something that has already been specified at a certain level of detail NOTE For example, a design class is a refinement of an analysis class

**Relationship:** semantic connection among model elements

NOTE: Kinds of relationships include association, generalization, metarelationship, flow and several kinds grouped under dependency.

**Specification:** declarative description of what something is or does

NOTE: Contrast: implementation.

**Stereotype:** new type of modeling element that extends the semantics of the metamodel

NOTE: Stereotypes must be based on certain existing types or classes in the metamodel. Stereotypes may extend the semantics, but not the structure of pre-existing types and classes. Certain stereotypes are predefined in the UML, others may be user defined. Stereotypes are one of three extensibility mechanisms in UML. The others are constraint and tagged value.

**Tagged Value:** explicit definition of a property as a name-value pair

NOTE: In a tagged value, the name is referred as the tag. Certain tags are predefined in the UML; others may be user defined. Tagged values are one of three extensibility mechanisms in UML. The others are constraint and stereotype.

**Type:** stereotyped class that specifies a domain of objects together with the operations applicable to the objects, without defining the physical implementation of those objects

NOTE: A type may have attributes and associations.

**Value:** element of a type domain

NOTE: A value may consider a possible state of an object within a class or type (domain). NOTE 2 A data value is an instance of a data type, a value without identity.

## Abbreviations

API	Application Programming Interface
CASE	Computer Aided Software Engineering
CORBA	Common Object Request Broker Architecture
CSL	Conceptual schema language
CSMF	Conceptual Schema Modeling Facility
DCOM/OLE	Distributed Compound Object Model/Object Linking and Embedding
GFM	General Feature Model
OCL	Object Constraint Language
ODMG	Object Database Management Group
OMG	Object Management Group
ODP	Open Distributed Processing
ODBC	Open Database Connection
SRS	Spatial Reference System
UML	Unified Modeling Language
url	Uniform Resource Locator
XML	Extended Markup Language
XMI	XML Metamodel Interchange
GML	Geography Markup Language
CDM	Conceptual Data Model
LDM	Logical Data Model

The UML notation was used to declare the data models schemas, to then create an XML file that encapsulates the data structure to be applied in several applications. In the scope of the work here described, this XML file was produced using a proprietary solution.

The data model schemas can also be declared in a standard language, defined by the Open Geospatial Consortium, called OpenGIS® Geography Markup Language (GML). By the importance and relevance of this language it was decided to summarize its main characteristics in the next chapter.



## ANNEX O– Traineeship work plan

---



## **Introduction**

The main goal of the thesis is to assist in the implementation, in Portugal, of the GIS and the geospatial database model as proposed by the WFD GIS Working Group. We therefore expect to be able to propose eventual transformations to the Portuguese reality, and we will describe strategies to integrate the amendments with relevant existing environmental information systems.

The thesis will focus its activity in the design and implementation of an information system prototype dedicated to report the present status of water quantity and quality in water bodies through a web application providing an interactive visualisation based on open-source and free technologies.

The work to be executed shall start with a contribution of the analysis of end-users requirements. These requirements will be reflected in the design of the system architecture and in the proposed geospatial data model. A geospatial data model defines, in addition to key dependencies, of a relational data model, also the geometrical dependencies between features in various tables.

Geospatial information related to hydrological features is proposed to be structured by defining various rules, both relational and geographic, and using a definition and drawing technique as UML to present users and developers in a platform independent manner the structure lay-out. Such database models, coupled with GIS applications, will facilitate integrated water resources management and information exchange between different actors from all water related domains at the different levels of administration (European, national and regional).

The Water Framework submissions for Article 3, 5 and 6 are already mostly being submitted by the Member States. Analysis of the submitted data can yield additional insight in the usability of the datasets and the data models thus far implemented. Further more the datasets being submitted can be analysed for their comparability between Member States and sometimes even within Member States

The PhD student believes the integration in the JRC Rural, Water and Ecosystem Resources Unit could help to clarify, not only the complexity of the WFD implementation but also all the technical aspects of its GIS development and realization.

## **Main Objectives**

The work programme will be based in 3 principal objectives, each objective corresponding to a specific task:

1. To explore the status of implementation of the GIS Elements of the WFD, namely concerning the geospatial and relational database design and the technologies applied to support the access and interoperability between information systems.
2. To collaborate in the research activities ongoing at the JRC, specially the research action 11603 - Water Quality Information System - in the Rural, Water and Ecosystems Resources Unit.
3. To test and explore the WFD data by comparing the results of WISE submissions with the results of other reporting obligations on the environment.

## **Tasks to be performed and calendar**

### **Task 1 (Objective 1)**

During this task the implementation status of the GIS Elements of the WFD will be evaluated by the trainee with special emphasis to the geospatial database model. Other specification elements will also be considered: geodetic framework, GIS Layer validation, feature coding, GIS datasets, data exchange and access, metadata, co-operation with the recently approved INSPIRE directive. The progress of the Common Implementation Strategy and its future work programme for the WFD will also be studied.

### **Task 2 (Objective 2)**

It is proposed to develop indices allowing assessment of the progress and quality of submission of various datasets being recently submitted under Article 5. The collaboration in the research activities ongoing at JRC will help the trainee to understand the interdisciplinary fields that are relevant to the WFD common implementation strategy.

### **Task 3 (Objective 3)**

By geometrically relating datasets created in the reporting of water related Directives and treaties, consistencies, inconsistencies and overlaps will be analysed by using for example datasets from:

- Urban Waste Water Treatment Directive;
- Natura 2000 Directive
- Ramsar Treaty
- Flood Directive;
- Priority Substances Directive;
- Marine Strategy Directive;
- Urban Waste Water Directive;
- Bathing Water Directive;
- Nitrates Directive.

### **Task 4**

A report containing the results of the previous mentioned tasks will be made as finalization of the work at the JRC.

## Tasks timeline

	1 <sup>st</sup> Month		2 <sup>nd</sup> Month		3 <sup>rd</sup> Month	
	1 <sup>st</sup> Week	3 <sup>rd</sup> Week	1 <sup>st</sup> Week	3 <sup>rd</sup> Week	1 <sup>st</sup> Week	3 <sup>rd</sup> Week
<b>Task 1</b>	X	X	X	X		
<b>Task 2</b>			X	X	X	X
<b>Task 3</b>				X	X	X
<b>Task 4 Reporting</b>					X	X

### Dissemination of results

A final report covering the items addressed during the three tasks mentioned and a presentation to JRC colleagues will be given.

### Repercussions in doctorate thesis

This training post aims to be a contribution to transfer knowledge, technology and experiences about good-practices on water resources information management, considering the complex new legal framework at European level, and the WISE instrument to upload, access, and disseminate data and information.

Since the implementation of the WFD and other water related Directives require the handling of large amounts of spatial data, both for the preparation of the River Basin Management Plans and for reporting to the Commission, it is important that technical norms and specifications, that are internationally accepted, could be known and efficiently applied.

