Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

Applying Domain Specific Languages and Model-Driven Development to GIS Applications

How the concepts of DSL and MDA fit with GIS application development

vorgelegt von

Christian Bucholdt
GIS_U1490, UNIGIS MSc Jahrgang 2010

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

Plauen, 23.04.2012

Erklärung über die eigenständige Abfassung der Arbeit

Ich versichere, diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit die wörtlich oder sinngemäß übernommen wurden sind entsprechend gekennzeichnet.

Plauen, 23.04.2012 Christian Bucholdt

Abstract

Foundation of this master thesis has been the consideration that Geographic Information Systems (GIS) are a specific kind of information systems, which are significant, different from regular business applications as most people know from their daily experience, This is especially true if looking in detail to the data, their processing and result presentation of those systems. However, the developments in recent years, for example the various map applications on the Internet and virtual globes as well as integration of GIS operations in enterprise information systems show that GIS is moving from a niche IT discipline in the mainstream becoming one of the regular services in information technology. This makes GIS a very good case for the usage of a domain specific language (DSL).

At the same time, the industry thrives to make their IT departments and services more efficient and increase or stabilize quality. Developments such as model-driven software development, agile methods and domain specific modeling languages are just some amongst the concepts to be considered.

So the author of this master thesis is researching the idea that the specifics of GIS make it worthwhile to apply the concept of a domain specific language (DSL) to GIS application development. Furthermore, the author considers the DSL approach only feasible if it is combined with the manufacturing-like approach of model-driven software development.

Looking back in the history of GIS application development and related research shows that there have been various approaches about visual modeling. Looking at these ideas of the beginning or mid 1990’s it becomes clear that these researchers tried to apply concepts, which would be considered under the DSL umbrella today, but at a time when necessary standardization was not nearly as far and ready for practical use as it is today.

Based on a sound explanation of current developments around domain specific languages and model driven software development, the GIS domain is positioned in the IT landscape as a horizontal service. The document presents a GIS meta model and its representation in a domain specific language. The modeling recommendation will be verified in a practical example

Table of Contents

Table of Figures

1. Specifics of Geographic Information Science and Systems

1.1 Introduction

The first chapter of this document is giving an introduction into basics of Geographic Information Science in order to make clear what are the specifics of Geographic Information Systems and Science (GIS) compared to regular business applications as many people know them from their daily work. It is the goal of the author of this document to set the "business" context of this document and to show based on this introductory, yet sometimes basic information, why GI systems are worth looking at them for a particular approach of software development. The level of provided GIS information has been chosen in order to give even not GIS professionals an understanding of the surrounding of this work.

According to [Longley, 2011] geographic information consists of attribute data, time and a particular place on earth. This place is the main concern of Geographic Information Systems (GIS). The characteristics of such a place are hard to determine. Usually such a place is not a natural unit. GIS provide a modeled representation of various real world phenomena. Due to the high complexity and the level of detail of the phenomena such model representations are incomplete and not absolutely exact[1].

It is worthwhile to mention the differences between regular business information systems and GIS system briefly. According to [Yeung/Hall, 2007] the difference can be found in the type of data to be stored and processed and the processing aspects itself[2].

While regular business applications mostly support entering of data, aggregation and calculation of data and their displaying and reporting of it, the GIS systems are very different. Regular business applications deal with relatively simple data formats (e.g. text, numeric) and data processing operations supporting repetitive daily work. The capture of GIS data often can be done by very specific devices such as satellite or aerial cameras only. Data types such as point, line, polygons or network data are completely different from regular numeric and textual data. The

[1] [Longley et. al., 2011], p. 148
[2] [Yeung/Hall, 2007], p. 93

data processing happens for a specific (unique) purpose using specific spatial operators. [Sheklar/Chawla, 2003] mention that this cannot be supported with conventional databases[3]. However, as of today mainstream database systems have powerful extensions to combine their experience in relational data storage and processing with spatial data storage and processing[4].

The following two chapters describe the details of data, processing operations and display of geographic information in more detail.

1.2 Data formats, objects and their relationships

According to [Yeung/Hall, 2007] are spatial data geographically referenced, which means they have coordinates as attributes, which specify a location on or near the surface of the earth. Coordinate attributes are related to geographic space, which is the relation to a particular geographic coordinate system, and to the geographic scale, which refers to the generalization and symbolization of data[5].

Spatial data can be separated in two main categories, vector and raster data. [Yeung/Hall, 2007] differentiate between geographic objects (vector data) and geographic cells forming a grid. [Sheklar/Chawla, 2003][6] call the categories object and field data. While objects (vector) are distinct and identifiable things such as points, lines or polygons, field data represent continuous phenomena in a cell. The cell contains geographic data and the geographic reference. The cell resolution determines how accurate the real world is represented by a grid of cells. The resolution of a cell grid affects the required database space for storage. The higher a resolution, the more details can be captured, but the higher the required data storage size in a database. The resolution depends on the kind of data capture, e.g. the camera resolution. In recent years the importance of raster data has increased, because of the increasing availability of satellite and aerial images[7]. With digital elevation models (DEM) the elevation height of a geographic location can be represented.

---

[3] [Sheklar/Chawla, 2003], p. 22
[4] Opinion of the author of this document which is referring for example to Oracle Spatial for Oracle 11g [Kothuri, 2009]
[5] [Yeung/Hall, 2007], p. 94 ff.; The following explanations are referring to spatial data aspects in general and do not address specific implementations in a particular database system (e.g. Oracle Spatial) in order to stay on the level of general concepts.
[6] [Sheklar/Chawla, 2003], p. 23 ff.
[7] Already mentioned in [Sheklar/Chawla, 2003], p. 228

A very specific and even more complex form of vector data are triangular irregular networks (TIN)[8]. These are data structures for representation of surfaces based on point data of irregular locations and calculated triangles between these points. TINs can be used very effectively for surface analysis such as visibility areas, water flows or 3D perspectives.[9]

Besides the spatial data there are pseudo-spatial data such as street addresses or socio-economic data. They can be used for spatial purposes once they have been related to a geographic coordinate system. This requires significant effort for digitization, transformation and geocoding functionality of GIS tools. In order to stay current with new housing developments or new streets this effort has to be repeated on a regular basis supported by onsite activities.[10]

[Yeung/Hall, 2007] use a functional data classification in order to differentiate geographic data and their usage in information systems.[11]

- Base map data layer: These are topographic base data and a spatial reference frame
- Framework data layer: These data represent geographic reference of human activities such as land development and administration, facility and resource management and addresses.
- Application data layer: These data are datasets for different spatial database applications using base map and framework layers.
- Business solution layer: This layer assembles the previously mentioned layers for a particular operation and decision making.

[Sheklar/Chawla, 2003] are more GIS oriented and separate between data objects (geometries) and relations between geographic objects (topology).[12]

Geometry is an abstraction of points, lines, angles and surfaces in different dimensional spaces.[13] In 1999, the OGC [OGC, 1999] introduced a hierarchy of data types represented as geometric objects. The hierarchy is based on simple geometries such as point, curve, surface and geometry collection. Real world objects are constructions of simple geometries.[14] The simple geometries

[8] For details see [Parmenter, 2007] and [Zeiler, 2010]
[9] [Zeiler, 2010], p. 214 ff.
[10] As an example Google Streetview can be mentioned.
[11] [Yeung/Hall, 2007], p. 96 ff.
[12] [Sheklar/Chawla, 2003], p. 26
[13] For a more detailed definition see the topic „geometry“ in Wikipedia, at: www.wikipedia.com (13.11.2011)
[14] For details see [OGC, 1999], chapter 2.1

need to have a spatially referenced position and orientation related to a particular geographic coordinate system. This is a prerequisite for a topological relationship between geometries. Geometries with the same attributes form a so called feature class. In 2011, the OGC [OGC, 2011] updated the data type hierarchy and added data types for surfaces in order to represent for example TINs.[15]

Besides the actual data objects, in a spatial context their relationships to each other are important. This is where topology comes into play. Spatial topology is the relationship between neighboring real world features, which can be represented as adjacency, connectivity and containment.[16] Topological relationships require special storage capabilities in databases.

A very comprehend model of spatial topologies has been introduced by M. Engenhofer in [Engenhofer, 1993]. He goes a different way than the previous definitions of spatial relationships, such as directions (north, south, west, east) or the image orientation (above, below, left, right).[17] His formalization of spatial relationships is invariant from transformations or other non-topological aspects. For this purpose he introduced the concept of 9-intersection and 4-intersection for binary topology. In order to describe such topology he focuses on the intersections of boundaries and interiors and the exterior of two spatial objects and the content which they enclose.[18] This leads to eight topological relations: Disjoint, Equal, Inside, Contains, CoveredBy, Covers, Overlap.

M. Engenhofer describes the binary topologies in a data model using the concepts of n-dimensional simplical complex and simplex to classify spatial dimension of an intersection. Another invariant is the number of separations of an intersection based on the content specification of empty/non-empty. Any intersection between two objects is then a function of their content, the dimension and separations.

Such a formal definition of spatial topology is the theoretical base for the implementation of spatial topologies in a database. [Yeung/Hall, 2007] follow the assumption that topological

---

[15] For details see [OGC, 2011], chapter 6.1.12
[16] For a more detailed definition see the topic „topology“ in Wikipedia, at: www.wikipedia.com (13.11.2011)
[17] A brief overview of references to these topics are given in [Engenhofer, 1993], p. 261 f.
[18] For details see [Engenhofer, 1993], p. 263 ff.

relations can be defined with three primitives: edge, node and polygons, in a linear graph.[19] Polygons are stored in a spatial database as discrete objects rather than topological structures of points and lines. Such a non-topological data structure brings some limitations in regards to analysis and storage of such data.

The previous mentioned aspects of geographic data have shown the specifics of these data types and therefore the special requirements to information systems to store them effectively and efficiently.

[19] Details regarding the implementation in Oracle Spatial can be found in [Kothuri, 2009], p. 713 ff.

1.3 Data processing and analysis techniques

Geographic data processing includes queries and operations against two- or three-dimensional data. Spatial operators will be used against spatial datasets. The results of such operations are usually represented with complex graphics on geographic maps.

[OGC, 2011] provides a classification of spatial operators into base, topological and analysis operators:[20]

- Basic operators: They provide access to general properties of a geometry object such as Dimension, Geometry type, Spatial Reference system, or basic transformations such as AsText, AsBinary, IsEmpty, etc.
- Topological operators: They evaluate topological relationships between geometries such as Equal, Disjoint, Intersect, Touch, Overlap, etc.
- Analysis operators: They are constructions of analytical spatial queries on one or multiple geometries such as Distance, Buffer, Intersection, Union, Difference, etc.

A different classification of spatial operators can be found in [Yeung/Hall, 2007] and with focus on vector object queries in [Sheklar/Chawla, 2003].

Because spatial datasets are mostly very large, spatial operations will be executed in a two-step approach.[21]

- Filter application: a pre-selection of data based on proximity
- Spatial operator application: The spatial operator will be applied to the pre-selected data set only

Each database system has an internal optimization of this procedure.[22] Spatial SQL extensions are mostly proprietary to a regular database management system, yet not part of the SQL standard. Spatial operators are not completely implemented in database management systems such as

[20] For details see [OGC, 2011], chapter 6.1.2
[21] For details see [Yeung/Hall, 2007], p. 120 or [Sheklar/Chawla, 2003], p. 116 f.
[22] For details on Oracle Spatial 11g see [Kothuri, 2009], p. 255 f.

Oracle, but just a subset most commonly in use such as Disjoin, Meet, Overlay, Cover, Equal, Contain or Inside.[23]

Data processing on raster data has its origins in [Tomlin, 1990], where a high-level computational language for map and image algebra has been defined. Map algebra for raster data and Image algebra for image analysis provide a set of operations such as Aspect, EqualTo, Hillshade, Merge, Slope and Zone Area.[24]

The raster operations can be separated in the categories local, focal, zonal, global and image analysis:

- Local operations: These operations "transform" an input raster in an output raster based on transformation rules of input cell to output cell. An example is Thresholding.
- Focal operations: In such an operation the resulting output cell depends on the input cell and its close neighbors of the input raster. An example is Slope.
- Zonal operations: In this case an output raster cell is the result of a function performed on the input raster cell and values of other cells in a certain zone area around the input raster cell. An example operation is Zonal Sum.
- Global operations: The output raster cell is a function of all cells of the input raster or even a combination with another input raster.

Image operations: Such operations slice an image into smaller pieces. The Trim operation extracts an axis-aligned subset of the input image. The Slice operation creates low-dimensional images from a high-dimensional image.

GIS tools such as ArcGIS enable the user to combine various spatial operations and apply them to input data. With extension mechanisms it is possible to add even more (domain specific) operations to the GIS using API for a common purpose programming language such as Python [Allen, 2010].

[23] [Yeung/Hall, 2007], p. 121 ff.
[24] [Sheklar/Chawla, 2007], p. 228 ff.

1.4 GIS data presentation

The output of GIS data processing and analysis are maps or charts[25] for the purpose of communication and decision support.[26] The maps can be formalized maps following cartographic conventions or map-like visualizations. The formal maps can include just topographic content or a specific thematic aspect such as census statistics for a particular country in Europe.

At the times when maps existed just on paper, the map was communication medium and storage medium in one. But today, storage and presentation of data is separated following good practices of software application development. Due to the fact that the creation of a paper map was difficult and expensive, a paper map might contain multiple topics in one map. Today it is relatively easy to produce a map for a particular purpose and a particular technological device from the same data source. Map creation has become much more flexible due to modern GIS tools.[27]

Map design has two, possibly contradicting, goals. On one side the map has to communicate a specific message with highlighting patterns and results, on the other side the map has to be easy to comprehend and therefore has to be a pleasing and interesting picture to the human observer. For these two reasons the design of a map is constrained by the following seven aspects of a map:

- Map purpose
- Reality
- Available data
- Map scale
- Map audience
- Conditions of the future use of the map
- Technical limitations

Between all these aspects the map designer has to find a visual balance.

[25] Maps are for terrestrial areas, charts are for marine areas. They are both what a not specialized user would call a "map" of a particular area of the earth.
[26] [Longley et. al., 2011], p. 302
[27][Longley et. al., 2011], p. 306 f.

The symbols to be used on a map have to be well-defined and follow accepted conventions in their classification. For example, blue lines should be used for rivers and not for streets, green areas should be used for forests rather than for buildings.

The observer of a map will classify map symbols by graphical attributes such as size, value, hue, saturation, orientation, shape, arrangement, texture and focus.

Data to be displayed on a map will be classified using classification schemas such as natural breaks, quantile breaks, equal interval breaks or standard deviation.

According to [Brewer, 2005] the correct use of color in the visualization of spatial content can make it easier for an observer to comprehend the purpose of a map more easily. Perceptions of color and data organization are related to each other and enable the observer to instantly translate color aspects into data organization. The human perception of color is mostly driven by the factors of hue, lightness and saturation. The hue of a color addresses the wave length and mixture of color, lightness addresses the relative perception of an object compared to other objects and saturation means the intensity of a color.

In general it becomes clear that GIS is very much a graphically oriented domain. GIS data and analysis operations are much less worth to a user without the presentation of (result) data on a map. Compared to non-GIS applications the graphical representation is not just about a nice perception by a user, but about exact location and transportation of exact information.

2. Problem Statement

Geographic Information Systems form a very specific part of today's information technology. As explained in chapter one it is significantly different from regular business applications in terms of data types, data processing, and result presentation. Nonetheless, GIS aspects become more and more part of everyday applications such as Google Maps[28], usage of Navigation devices in private cars, etc., where they are normally not used and operated by GIS experts.

This leads to the fact that GIS applications are in one way different to develop than some other applications, but in another way it is required that general software development methods and practices have to be applied.

Today's GIS tools are providing a "one stop shop" for developing and running a GIS application for a particular business purpose. On one side, this follows very much a domain-specific paradigm, but on the other side it leads to a mix up of business domain aspects and technological aspects in one GIS tool, which is the base of the development of a GIS application. Considering the complete lifecycle of the GIS application this can cause problems such as a tool vendor lock-in and capturing of domain knowledge in a particular technology or even in a particular GIS tool. Considering the fact that GIS more and more moves from an area of "specific purpose application" to general usage in all kind of applications in everybody's life, the lock-in with a GIS specific tool causes problems in the broader (re-)use of GIS.

A model driven approach might help to resolve these issues[29] by clearly separating business domain, GIS aspects and technical implementation of an application in different models. The MDA approach has not fulfilled all these hopes as they existed when it was introduced to the software industry. The idea of Domain Specific Languages (DSL), which has come in the industry focus over the last couple of years, can address some shortcomings of the original MDA approach and at the same time consider GIS specifics in the general software development process in a standardized way.

[28] Google Maps is a trademark of Google Inc.
[29] Examples of MDA application, although described on a very high level, can be found at [Guttmann, 2007]

3. Master Thesis objectives

This document is addressing the possibility of using domain specific languages (DSLs) and model-driven architecture (MDA) and development (MDD) in the development of Geographic Information Systems.

The Master Thesis is researching whether:

- GIS is a horizontal domain, which is specific enough to gain significant benefit from using DSLs for software development.
- Previous and current research around this topic provides a foundation for this document. The author is looking for required extension to the current research state in the industry and science in order to address all “GIS domain” aspects.
- A GIS domain meta model and a GIS domain specific language can be applied to a practical example in order to show how GIS application development could be supported.

The document is structured in two parts, a more theoretical and a more practical part.

The theoretical part of the document addresses the state of the art in regards to visual modeling of geographic data, model-driven software development and the current use of Domain Specific Languages (DSL) in the software industry. The author will research the topic of different modeling approaches that have been introduced in the GIS community and what is the state of these attempts today. This will address topics such as visual modeling and object-oriented GIS modeling as introduced in the mid-1990’s.

Then this document will give an overview of different approaches of positioning, developing and using of DSLs in the software industry. This part of the document will explain the basics of DSL development and different viewpoints on the topic. Furthermore, this part of the document is looking at the current state of the industry on the DSL topic and model-driven software development as it has been introduced at the beginning of this millennium.

The connection between the first and second part of this document is the positioning of GIS as an enabler service for various business domain solutions and their relations to different implementation technology.

The second, more practical part of the document addresses a possible usage of Domain Specific Languages and Model Driven software development for GIS enabled software applications. For this purpose the author will develop a modeling approach for the GIS domain, which goes beyond the current research by incorporating behavioral and presentational aspects in the modeling process. This results in a GIS meta model and a UML-based DSL.

The GIS domain model and the UML based DSL will finally be applied to a prototype focusing on a sample of the INSPIRE infrastructure for spatial information of the European Commission.[30] A sample model will be developed applying the GIS DSL, and then model transformations will be shown for data structure and a GIS operation. With this sample the author shows how the problem of mixing up GIS domain specific knowledge and specific technology or vendor products could be resolved. This can be taken as a recommendation of how a higher reuse of GIS functionality can be achieved.

[30] For details see [Inspire, 2010]

4. Literature, research overview and state of the art in the software industry

4.1 Modeling of GIS aspects

4.1.1 Introduction to GIS modeling

As a first modeling approach for GIS topics can be considered *map algebra* as introduced in [Tomlin, 1990]. D. Tomlin introduced a scientific general purpose GIS "language" for GIS scientists, not for GIS users.

In the mid-1990's, Geographic Information Systems became more powerful, but more complex too. So at this time, a strong movement started to address this complexity with different modeling approaches in order to make GIS more easily to use by (non-scientist) users.

One approach was the idea of visual modeling of GIS data and processes with J. Albrecht as one of the main contributors. Another approach went along the upcoming object-oriented software development and the related UML modeling path using object-oriented modeling for GIS specific purposes. G. Kösters was one contributor in this area

Looking back, both developments came already pretty close to what today could be considered a DSL for GIS. This will become clearer after a more detailed explanation of both initiatives and the details about DSLs in the next chapter.

4.1.2 Visual modeling of spatial problems

The idea behind visual modeling as introduced by [Albrecht, 1995] was to provide an abstraction of geoprocessing to the ordinary user. Software and hardware complexity was requiring more and more diverse skills to master GIS. J. Albrecht mentions in [Albrecht, 1997] previous research such as about data-driven GIS and process-driven environmental modeling, data integration issues and the absence of user friendly interfaces of GIS products in.[31] At this time the usage of GIS outside the narrow GIS domain was very difficult. The goal of J. Albrecht was to hide

[31] For details see [Albrecht, 1997], p. 155 ff.

technical details and let the user focus on solving a particular spatial problem. In [Albrecht, 1997] seven criteria are mentioned, which should be fulfilled with a user friendly modeling approach:

- Visual: Easy to use graphical interface for the user
- Interaction: Users can develop interactive scenarios
- System dynamic modeling: User can model in a conceptual and flexible way rather than driven by technical constraints and details
- Spatial: Specific analysis and presentation functionality for spatial topics
- Model database: The history of development and analysis have to be stored in a versioned manner for models and results
- Integrated: User have to be able to easily link spatial and non-spatial data for analysis purpose
- Generic: The spatial tool box was supposed to be independent from a particular domain

With visual modeling GIS was supposed to be used in a more interdisciplinary way. The idea was that this could be achieved by having each user interaction take place through a graphical user interface.[32] At this point, only VGIS and a parallel work in New Zealand by Samuel Mann [Mann, 1996] (SPMS) have been close to fulfill the seven criteria mentioned above.

Both use the concept of flow charts on top of an actual GIS. While VGIS was a research study, which has invested effort in independence from a particular system and data structure, the SPMS approach was targeting a specific application purpose for environmental decision making.

Virtual GIS (VGIS) was a prototype, developed as part of a research initiative for semantic modeling and extraction of spatial objects from images and maps. Image and map interpretation was supposed to be automated and modeling for object capture, description and usage combined in a common user interface. Semantics have been considered the sum of all relations between the modeled world and its representation in a computer.[33]

VGIS consisted of the following components:

[32] [Albrecht, 1995], p. 141
[33] [Albrecht, 1995], p. 142

- Visual language interface editor
- Interpreter to translate visual language in commands of an underlying GIS
- GIS with a documented interface, acting in a black-box manner for the user interface and the interpreter
- Processing plan builder

The visual editor was providing generic GIS functions to the user and the interpreter was translating these generic functions and their usage to a specific GIS product with specific GIS functions. Albrecht was using the open-source GIS GRASS for its studies.

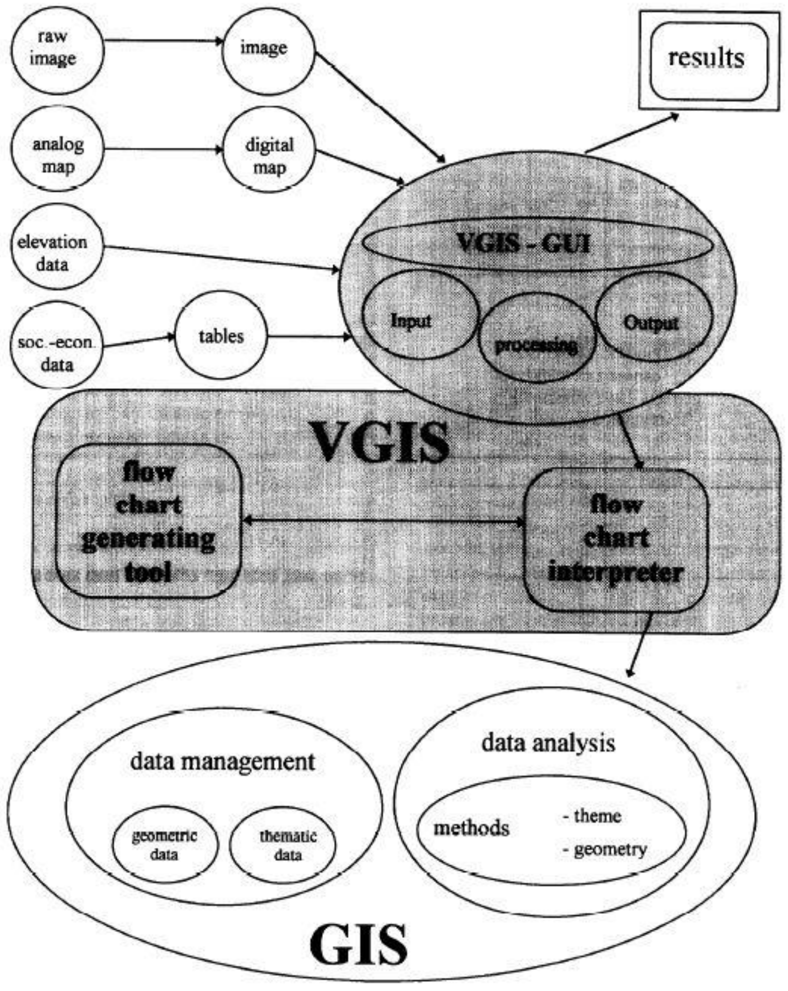


Figure 1: VGIS conceptual design by Albrecht (Source: Albrecht, 1995)

Albrecht developed VGIS based on a set of 20 generic GIS functions, which he considered enough for resolving most GIS problems, and which are generic enough to be used independent from a specific GIS product.

For these 20 GIS functions the scope was broadened from the original narrow image analysis scope, considering location, temporal and thematic aspects[34].

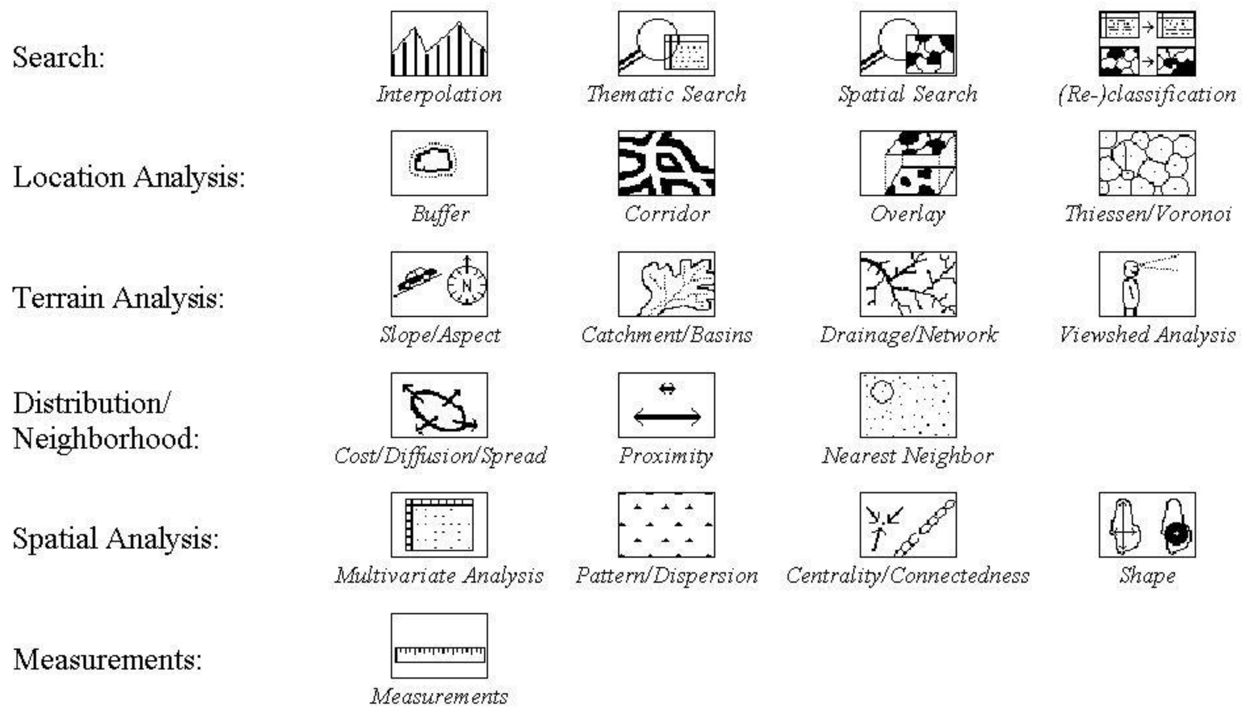


Figure 2: Generic GIS functions by Albrecht (Source: Albrecht, 1997)

According to J. Albrecht, GIS work was so becoming more user task oriented. The visual system was supposed to provide a meta model, which supported what a user wanted to achieve rather than a specific GIS implementation.

An issue which could not be addressed with this prototype was the implementation of dynamic flows, where input parameter change at runtime and where iterations and conditional behavior could be modeled.

[34] See also [Albrecht, 1997], p. 158 ff.

An overview and a comparison of Albrecht's VGIS approach, the first version of the ArcGIS Model Builder[35] and the Spatial Modeling Environment (SME3)[36] for collaborative global climate modeling of scientists can be found in [Crow, 2000]. Interestingly the research studies VGIS and SME3 do not play an important role today, but the ArcGIS Model Builder has evolved and is a powerful part of ArcGIS 10 more than 10 years later.

4.1.3 Object-oriented GIS modeling

A different approach was taken by G. Kösters [Kösters, 1996], but around the same time as Albrecht's work. He considered GIS as a specific domain, but integrated in multiple business contexts. From his perspective, development of GIS applications using regular software development and modeling methodologies such as Entity Relationship models[37] or different object-oriented analysis (OOA) notations[38] have not been sufficient. It was not business user friendly and missing sufficient graphical representation. So far he had similar thoughts as J. Albrecht as explained in the previous chapter.

Kösters approach was the development of a modeling extension to the OOA notation for GIS elements, element topology, network and temporal aspects. This idea could be considered similar to today's UML profiles as an extension mechanism in the UML standard [UML241I][39] or a domain specific language (DSL) for the GIS domain. The extension to OOA was called GeoOOA, which provided primitive data structures for geographic topics. The GeoOOA extension did not contain model elements for processes and activities.

Kösters made a clear distinction between spatial and non-spatial classes when representing real-world objects. He introduced special spatial classes for point, line, region and raster besides conventional classes for non-spatial data structures. Each spatial class type came with geometric standard services (operations) and was represented by a specific icon. This could be compared to the current UML concept of profiles with stereotypes [UML241S].[40]

[35] For details see [Murray et. al., 2000]
[36] For details see at: http://www.uvm.edu/giee/SME3/ (30.12.2011)
[37] For details see [Chen, 1976]
[38] For details see [Coad/Yourdon, 1991]
[39] For details see [UML241I], chapter 12
[40] For details see [UML241S], chapter 18

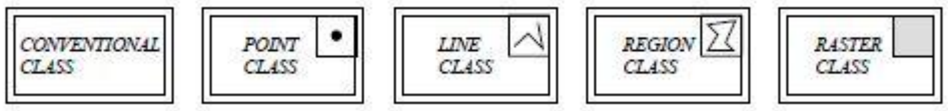


Figure 3: Spatial class types by Kösters (Source: Kösters, 1996)

In addition, a clear separation between topological and conventional relations has been made. So it was possible to make topological relations explicit and not keep them hidden in textual description. With a consequent versioning a reuse of classes and topological relations has been made possible. Kösters introduced a whole-parts relation for the specified spatial classes again with different icons in a triangular symbol. There have been actually three whole-parts relation types: covering structure, containment structure and partition structure.

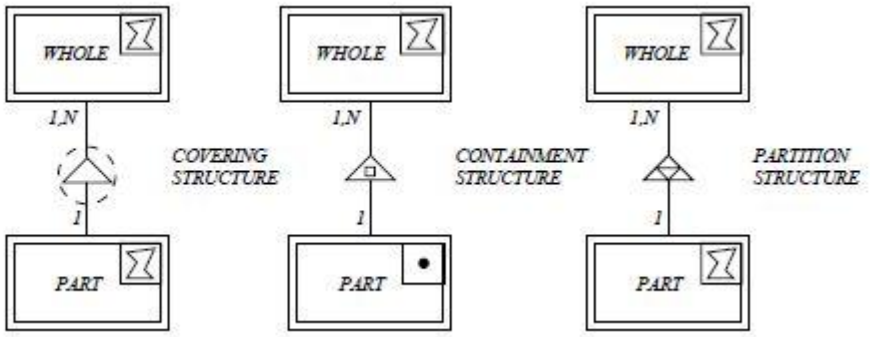


Figure 4: Topological relation types by Kösters (Source: Kösters, 1996)

Each relation type has again standard services (operations) such as accessing, connecting, disconnecting or querying for neighbors, as the spatial class types.

The concept of spatial networks has been constructed of nodes and links by Kösters. Both are spatial objects, which are *spatially embedded* in a network.[41] A network class has at least one node. A node of a network has no direct connection to another node of another network. But node and link classes can occur in multiple networks. Node classes are either point or region classes of the previously mentioned spatial class types, while link classes are represented by line classes of the spatial class types. The relations between classes are typed and represented by a symbol, telling whether a class is a node or a link class.

[41] Kösters, 1996, chapter 3

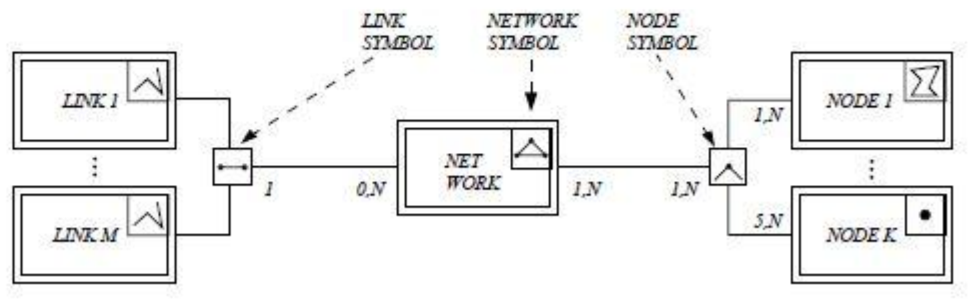


Figure 5: Network structure by Kösters (Source: Kösters, 1996)

Temporal classes and Contract classes are more advanced structures of Kösters' GeoOOA notation. Both can be attributes of spatial and non-spatial classes.

G. Kösters was considering appropriate tool support essential for broader usage of GeoOOA.[42] A lack of such tool support and a lack of standardization might be the reason, why GeoOOA has not been used more widely.

A similar approach has been taken by N. Tryfona in [Tryfona, 1996]. He is mentioning the lack of a concept for GIS application development with a missing software methodology and process considering GIS specifics. For this purpose, Tryfona was applying object-oriented methodology to GIS application development. In difference to Kösters [Kösters, 1996], he was building an extension for the Object Modeling Technique (OMT) modeling notation by [Rumbaugh, 1991]. Tryfona was trying to integrate object data (vector) and field data (raster) in a single object model.[43] This object model was supposed to represent the basic characteristics of spatial information. He added the two model constructions Space and Position, while a Position is always related to a Space by an *is_in* relation. A Space represents geometry with attributes which describe the Space. A Position represents a geographic class position in space using a relation *is_located_at*. Geographic classes are subsets of geometric figures such as point, line and region. They are defined by shape, size, location and orientation. Shape could be zero, one or two dimensional.

[42] See [Kösters, 1995]
[43] In 2001 a similar, but more concrete approach using OMT in regards to geographic data modeling was taken by [Borges et. al., 2001]. There a set of conceptual modeling constructs including constructs for relationships have been introduced.

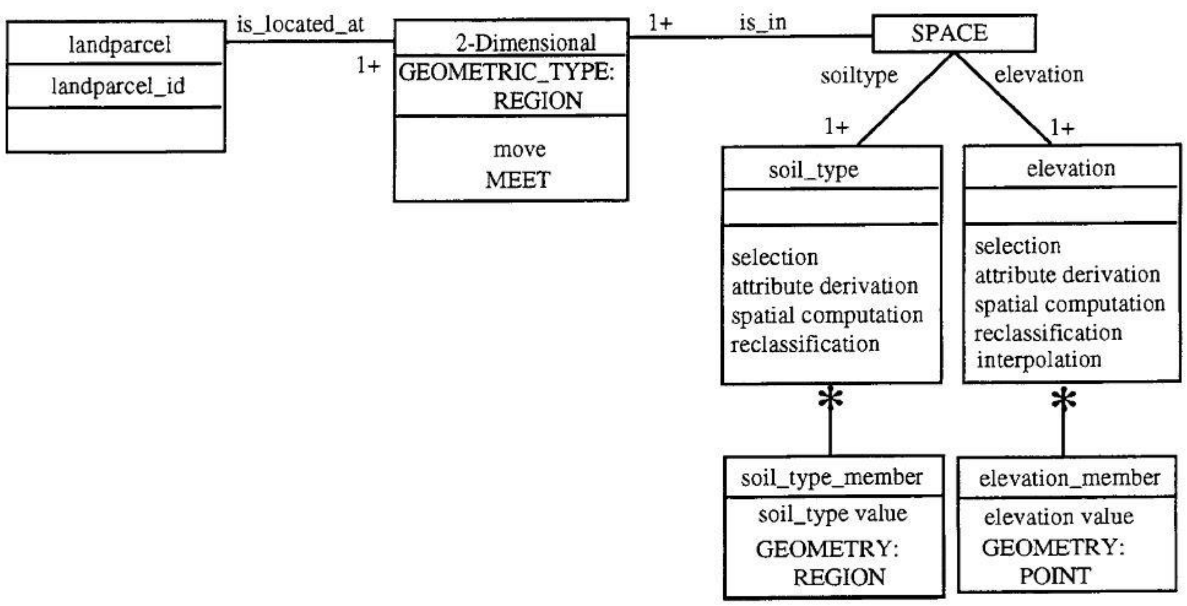


Figure 6: Sample spatial modeling by Tryfona (Source: Tryfona, 1996)

Both, Kösters and Tryfona have a focus on data structure elements rather than processes and activities. This is a difference to Albrecht, who was using data flowcharts as base for its visual modeling. As in Albrechts work, dynamic process modeling is missing. Both protagonists of the object-oriented focus area had user friendliness as a main reason for their work. While Kösters put the focus on a pragmatic usefulness of modeling for GIS development Tryfona is presenting a mathematically correct construction of its modeling approach. Köster is not explicitly introducing a meta model in his work, while in Tryfonas work the user seems to be overwhelmed with meta model information, which reduces the user friendliness compared to Kösters focus on graphical representation with different icons.

Both use a particular modeling notation with a (non-standard) extension approach, but none of them has a focus on a particular software development process.

But both approaches also failed to gain a wider audience or even to become a standard approach. From today's perspective the later consolidation of three modeling notations in the 1990's in the UML standard and the development of this standard up to UML 2.x has provided an environment, which was missing then to make their work more popular and used by a wider audience. Exactly this base could make a similar approach as GeoOOA etc. more successful when using UML standard and standard extension mechanisms together with the current knowledge about DSL development.

Another pragmatic approach of using object-oriented methodology can be found in [Shanan/Gorani, 2007] with a focus of using Use Case modeling and the Unified Process as software development process methodology. The approach is using purely UML standard notation without any GIS specific notation elements. They are describing the use of UML for modeling a business solution rather than the GIS specifics of the system. For that reason, this work cannot be compared to the work of Albrecht, Kösters or Tryfona.

The latest work which could be considered similar to the modeling work of Kösters and Tryfona is the use of UML for description of data transformation rules from national or proprietary GIS data formats in the INSPIRE data format for GIS data [Inspire/GMES, 2011]. An UML extension, called UMLT, enables the transformation of data via a XMI interface and the tool FME from Safe Software. The goal of this research project is similar as already mentioned for the other research to provide a more user friendly interface and to work on a higher abstraction from a particular GIS technology and GIS expert knowledge. The current findings show that conceptual models are often not exact enough to drive transformation rules. The modeling results are often drawings and not fully correct models, which causes syntactical errors. At the time of publishing of the results the encoding rules for the transformations have not been fully operable yet.

4.1.4 Conceptual data modeling

A newer approach of GIS modeling starting in the first decade of the 2000's is about conceptual modeling with a strong focus on geographic data and database modeling. In this sense, a conceptual model is a logical transformation of real world phenomena in model objects in a formalized way, but without consideration of technical implementation aspects. In difference to [Albrecht, 1996], the conceptual modeling approach is not addressing spatial operations. A good overview about the most current discussions of this topic can be found in [Shekhar/Xiong, 2008].

[Parent et. al., 2008] address the problem of multiple representations of geographic data. For example, a city might be represented as a point in one context and as an area in another. The MADS (Modeling Application Data with Spatio-temporal features) framework uses a non-standard extension to the Entity-Relationship Model notation with the dimensions of space and time representation in order to enable a domain specific data perception. The geographic phenomena data basically have to be stored multiple times in a “multi perception” database.

[Miralles et. al., 2008] put the focus on the usage of MDA for the development of spatial databases. A so called Enriched MDA framework for spatio-temporal database design has been introduced, which through model transformation will finally generate SQL code for the actual database creation. The framework does not use standard UML, but uses the pictogram language as described in [Bedard/Larrivée, 2008].

[Bedard/Larrivée, 2008] introduce a visual modeling language of GIS data aspects using a set of standardized pictograms and pictogram placements.[44] These pictograms could be used as extension to UML. Since the pictogram concept is similar to the stereotype icons, but goes beyond a specific CASE tool is required for modeling and potential translation of conceptual models into ISO or OGC data specifications.

[Lisboh/Iochpe, 1999] have introduced a geographic framework (GeoFrame) where they are proposing geographic data modeling based on the usage of UML and the UML extension mechanism of stereotypes. In [Lisboa et. al., 2008], the usage of UML class modeling in a framework of three levels is explained. The planning level, as the most abstract level, maintains the modeling of various themes and its relation to a geographic location. The meta model level separates between conventional (non-spatial) objects and dedicated geographic phenomena. The geographic phenomena can be separated in field (raster data) and object (vector data) phenomena. On the spatial representation level, the framework introduces the representation of point, line, polygon, cell, TIN and other geographic data representation. The GeoFrame concept uses temporal modeling elements similar to [Kösters, 1996] and has been extended by [Sempliuc, 2009] with network modeling elements as introduced by [Kösters, 1996]. This framework follows a top-down modeling approach using hierarchical themes constructions.

Based on the GeoFrame concept, the usage of a specific UML based meta model and UML profile has been introduced in [Lisboa et. al., 2010]. A meta model similar to what has been

[44] The research regarding this topic already has taken place earlier as can be seen in: Bédard, Y., Larrivée, S., Proulx, M., Nadeau, M. (2004) Modeling geospatial databases with plug-ins for visual languages: a pragmatic approach and the impacts of 16 years of research and experimentations on Perceptory. in: CoMoGIS 2004. LNCS. vol. 3289, pp. 1148-1158 and in: Bédard, Y. (1999) Visual modeling of spatial databases: towards spatial PVL and UML. Geomatica, 53(2), pp. 169-186

described by A. Kleppe in [Kleppe, 2009] for language development has been chosen as base for a UML profile, called GeoProfile, for conceptual geographic modeling. As in GeoFrame the "theme" topic is the base type of the meta model and the profile has been developed applying the following principles:

- Separation between geographic and non-geographic (conventional objects) themes
- Separation of field, object and network elements as geographic themes
- Separate stereotypes for temporal aspects
- Separate stereotypes for topological relationships:[45] touch, in, cross, overlap, disjoint

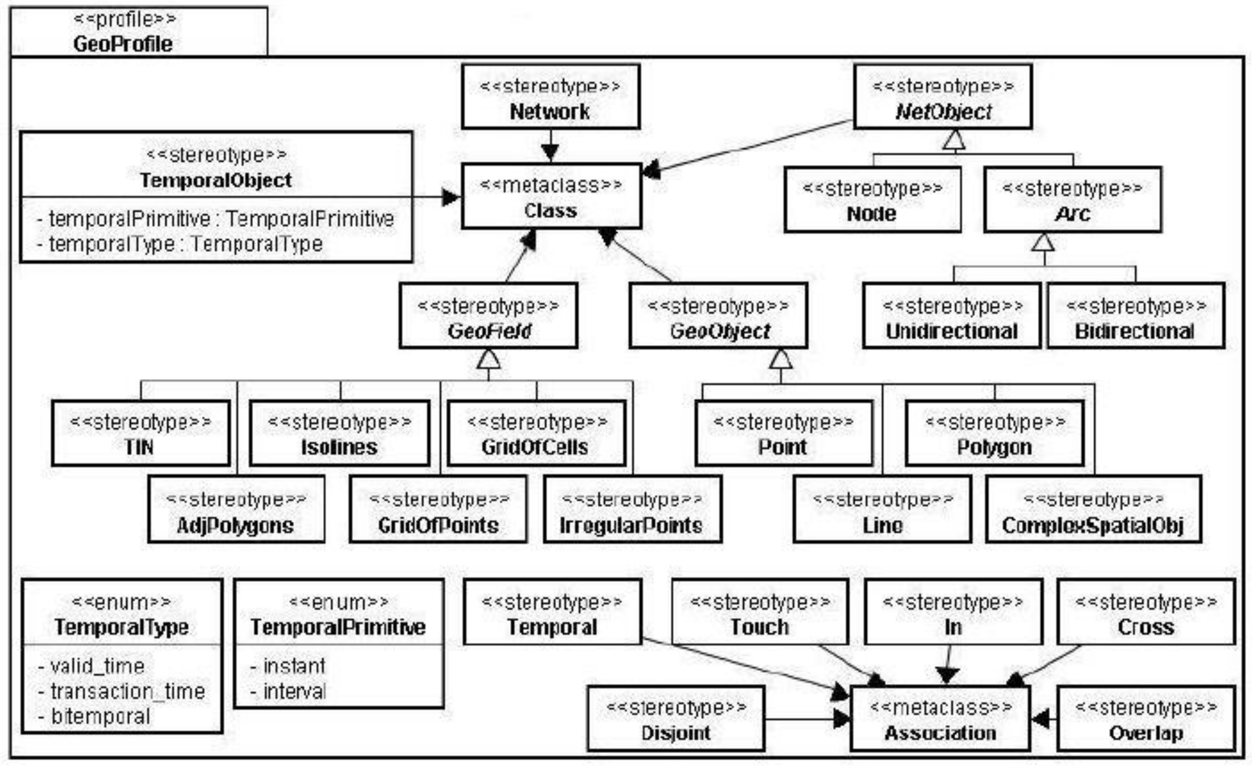


Figure 7: UML profile GeoProfile (Source: [Lisboa et. al., 2010], p. 26)

The introduced stereotypes extend the UML meta classes Class and Association. No stereotypes for behavioral modeling such as for activities or state transitions and for GIS presentation aspects have been introduced. As the modeling focus has been on data modeling, this is no surprise.

[45] [Lisboa et. al., 2010] uses the base topologies as introduced by [Clementini et. al., 1993], which can be used to construct any other topology. This is a reduction in the formalized model of binary topologies as introduced by [Engenhofer, 1993] due to its focus of usage by end users rather than an exact scientific model.

Although the management of geographic data is an important topic of GIS applications, missing GIS operations and presentation aspects prevents the full usage of a model-driven software development process for GIS applications.

4.1.5 Conclusions

This research overview shows that modeling has been a significant topic for research and discussion in the GIS community in the mid-1990's. This was at a time, when software development has not been supported by modeling standards such as UML, but many different notations have been just under development. It seems that the GIS community was very affine to the modeling approach because of its general focus on graphical representation of output of computational output such as maps.

Apparently since that time, general developments in the software industry in regards to modeling and standardization have not revived the research in this area. Approaches such Kösters' GeoOOA [Kösters, 1996] or Albrechts VGIS [Albrecht, 1997] might have had much more success in being applied by a broader audience when using UML or the concept of DSL/DSML.

The developments since 2000 take these considerations in account, but with a dedicated focus on geographic data modeling [Parmenter, 2007][Parent et. al. 2008][Selic, 2011]. This development seems to be driven a lot by the MDA approach in software development in general and its standardization by the OMG. The various attempts bring the development of "GIS domain specific languages" and general model-driven approaches closer together, although the research and prototyping have not taken place in the explicit context of DSL development. In order to apply the domain-specific modeling approach and model-driven software development to GIS applications, the two other aspects which separate GIS from regular business applications, operations for geographic data processing and presentation of geographic data have to be taken into account.

4.2 Domain Specific Languages

4.2.1 Introduction to DSL

According to [Fowler, 2010] Domain Specific Languages (DSL) are (abstract) computing programming instructions, have general language characteristics, limited expressiveness and focus on a particular domain.[46]

Computing programming instructions in this sense mean that humans can instruct a computer to do certain things in an easy to do manner. The instructions are therefore executable by a computer.

The language characteristics mean that a DSL has a defined set of expressions and that these expressions and composed combinations of such expressions form a specific semantic.

The limited expressiveness addresses the fact that a DSL has a feature minimum with focus on a particular purpose rather than a general purpose to express. This has a strong relation to the domain focus of a DSL. A DSL is supposed to help resolving issues with a clear, but rather narrow focus.

Fowler talks about three categories of DSLs: external DSL, internal DSL and language workbenches.

An external DSL is a language which is different from a main programming language of the system that is to be developed. It has a custom syntax or a general purpose syntax such as XML configuration files for declarative programming.

An internal DSL is using a general purpose language. This language is the same language as the one of the system that is to be developed. But the DSL is using just a subset of the actual language features in a particular style. A sample for such a language is Ruby on Rails.

Language workbenches are special Integrated Development Environments (IDE) for DSL creation. Such workbenches also provide an environment for DSL editing. Known products of

[46] Fowler, 2010, p. 27

this category are MetaEdit from the company MetaCase in the United States or Hypersenses of Delta Software Technologies in Germany.

From the perspective of the author of this document the language workbenches are just a specific type of external DSL.

Another DSL characterization can be found in [Kleppe, 2009]. From her perspective, the term and the boundaries of a domain are not clearly defined, which leads to a wide variety in the interpretation of what DSLs are.[47] Fowler sees the domain not as a good boundary for the definition of a DSL.[48] For this reason, A. Kleppe provides different opposite aspects in order to define the DSL boundaries.

The first opposite is between domain experts and computer experts. Should a DSL be used by a domain expert or even developed by a domain expert?[49] It seems to be clear, that a DSL is used to improve communication between domain and IT experts, but not to bring programming to the domain experts.[50]

Another opposite is between large and small user groups. Is it correct, that fewer users mean a more specific domain? Are general purpose languages used by a wider audience than a DSL?

An interesting point, introduced by A. Kleppe is the difference between horizontal and vertical DSL. Technical oriented DSLs have horizontal orientation, while business oriented DSLs have a vertical focus. But from her perspective business oriented DSLs have their foundation on technical DSLs. This might enlighten the discussion about domain and computer experts as mentioned above.

The comparison of DSLs and Frameworks or APIs is another opposite to consider. Usually Frameworks or APIs are structured in an existing language, but not a separate language. However, they could be seen as internal DSLs as defined by [Fowler, 2010]. The difference seems to be in the existence of a grammar.

[47] [Kleppe, 2009], p. 33 ff.
[48] [Fowler, 2010], p. 29 ff.
[49] See also [Fowler, 2010], p. 27
[50] The communication gap and need for a common language between developers and domain experts has been described very well by E. Evans in [Evans, 2004]

Another consideration of what a DSL is, are reasons for their use and possible constraints:[51]

- Improving development productivity by a more clear communication
- Improving collaboration between IT and domain experts by a more clear and precise communication
- Change of execution context by a more abstract programming approach
- Alternative computational model using more declarative programming rather than imperative programming

Issues of DSLs are:

- Cost of learning another language (for IT and domain experts)
- Cost of building and maintaining another language
- Ghetto language, which might be too specific
- The fit between the increased abstraction and the fit to the real world might be problematic

Fowler has a very IT focused perspective on DSLs. All his descriptions and details about DSLs are from a programmer’s perspective. This, from the perspective of this documents’ author, reduces exactly the advantage of DSLs, which is supposed to enlighten the communication between IT and domain experts. In order to show two different approaches to DSLs and language development in general, this document gives more details of other research perspectives in the following two chapters.

4.2.2 DSLs and language development patterns

A slightly different approach is taken by T. Parr in [Parr, 2009]. His work is not focusing on DSL development directly, but on development and usage of computing languages in general. From his perspective this process follows certain patterns. Different patterns can be used for different kinds of languages.

Following Parr, a language application starts with an input language stream[52] following a certain grammar:[53]

[51] [Fowler, 2010], p. 33 ff.

1) A reader is recognizing the input constructs of the language stream
2) From this an intermediate representation (IR) of the input stream will be build
3) Then a semantic analyzer is analyzing the semantics of the intermediate representation and adds additional information as kind of annotations to the IR
4) Then a generator creates output from the enriched IR such as source code or byte code

The so called intermediate representation (IR) at T. Parr can be seen as very similar to A. Kleppe's Abstract Syntax Model[54] and Fowlers Abstract Syntax Tree.[55] Parr's Semantic analyzer can be very much compared to Kleppe's Semantic Meta Model[56] and Fowler's semantic model.[57] However, Fowler sees its semantic model as optional although he recommends to always having one. Parr thinks that the semantic analysis is required except of very simple parser patterns. For Kleppe, the Semantic Model is an essential meta model.

From this general process Parr develops different language application categories such as:

- Reader/Parser: These applications build the data structure of the IR from the input language stream
- Generator: These applications "walk" the IR data structure and create output
- Translator/Rewriter: These applications are a combination of Reader and Generator
- Interpreter: These applications are reading, decoding and executing instructions from an input stream

Complete language applications have one or multiple of the mentioned application categories and they are using different patterns for language processing, which he calls Reader/Parser patterns, Tree construction patterns, Tree walking patterns, Semantic analysis patterns, Interpreter patterns and Translation patterns.

Reader and parser patterns can be basic or advanced because different languages might require a different degree in their parsing complexity and therefore stronger parsing patterns.[58] The more

[52] This is similar to Kleppe's Concrete Syntax Model in [Kleppe, 2009], p. 93 ff.
[53] [Parr, 2009], p. 4
[54] [Kleppe, 2009], p. 75 ff.
[55] [Fowler, 2010], p. 43
[56] [Kleppe, 2009], p. 131 ff.
[57] [Fowler, 2010], p. 159 ff.
[58] For details see [Parr, 2009], p. 21 ff.

complex a language is, the stronger has to be the parsing pattern and therefore the slower is the parsing procedure. Advanced parsers can handle a broader input scope of a language and can use technology such as backtracking in order to recognize more complex language constructs and change the parsing flow dynamically.

The Parser recognizes valid language constructs and can distinguish between separate language constructs such as construct and sub-constructs. Language grammars are the specifications of parsers. They are basically a DSL for parser generators, which build the parser basically automatically. The parser applies grammar rules against the input language stream in order to check the validity of the input.

The Tree Construction patterns are responsible for creation of an intermediate representation (IR) of the input language stream in the computer memory.[59] Mostly this is done in form of a parsing tree or more complex an abstract syntax tree (AST) of the recognized valid input elements, which again can store additional information. The tree contains the essential set of operations and operands and their relations from the input stream. An AST can help to require a full understanding of a language phrase since it enables multiple tree walking iterations (see next paragraph) and can represent ordered and nested data structures. This again helps to identify patterns in the input stream. Recognized information can then be added to the AST elements and reused in additional tree walking operations.

Tree Walking patterns are using the built abstract syntax tree as mentioned before. The tree walkers can be embedded in the AST or external to the tree using a tree grammar. The tree walkers extract pattern information from the AST, can restructure the tree or translate tree nodes. Tree walking consists of a visitation order of nodes and the actual visitation of a node, which is performing an action on a particular node each time a node is "passed" during the walk in a certain order.

The visitation order is implemented in an algorithm (e.g. *depth – first search* as in sample above), which implements a fix sequence for node discovery in an abstract syntax tree.

[59] [Parr, 2009], p. 73 ff.

T. Parr recommends that tree walking and visitation actions are separated in order to organize the visitation actions separately and being able to change the visitation order independently from the visitation actions or node changes.

Semantic Analysis patterns are responsible for “explaining” what a certain language symbol actually means.[60] For this purpose a language application is using one or multiple symbol tables for tracking and recognizing symbols of the input language stream. A symbol will be discovered initially and stored in the symbol table with name, category and type. When the same symbol appears again in the tree, the semantic analysis has to recognize the symbol again. In order to achieve this, the AST has to be walked at least twice and symbols and forward references have to be mapped to each other.[61] For this purpose it is important to recognize not just the symbol, but the symbol’s scope, which is its visibility, too. The symbol scope could be global, local or nested. This can be called a scope tree within the AST. In addition, correct semantics can be ensured by correct typing of symbols, which is the enforcement of semantic grammar rules in yet another tree walk. This can be done dynamically at runtime in languages such as Python or statically at compile time as in C++.

Interpreter patterns take care of executing the information recognized and stored in the intermediate representation (IR) such as in the abstract syntax tree. This means they are one way of creating output from the language analysis.[62] Interpreters can be syntax directed, tree based or stack based.

Parr separates between high-level interpreters with direct source code execution (e.g. Python) and low-level interpreters for execution of byte code (e.g. Java[63] Virtual Machine). He considers high-level interpreter best for DSLs because of their simplicity, low-cost implementation and direct AST execution. On the negative side he sees the runtime efficiency.[64] Low-level interpreters are better for general purpose languages because of runtime efficiency and strength of possible operations. On the down side to mention are high complexity and the required preprocessing for byte code creation as for example the *javac* compiler.

[60] [Parr, 2009], p. 131 ff.
[61] [Parr, 2009], p. 156 ff.
[62] [Parr, 2009], p. 217 ff.
[63] Java is a trademark of Oracle Corporation
[64] [Parr, 2009], p. 219 ff.

The last pattern category to mention is Translation patterns. They are an alternative to the direct execution as in the Interpreter patterns. Instead of executing the AST these patterns translate the input language in another (output) language, which again has all characteristics of a language.[65] The translation can take place syntax directed, model based or template based.

While syntax directed translation basically occurs while walking the AST once, the model-driven approach occurs over multiple tree walks. The usage of templates for output generation enables the separation of logic (code) and output format. The logic basically fills in values in the output template. With this approach a reuse of the templates for different problems is possible.

4.2.3 DSLs and meta models

In [Kleppe, 2009], A. Kleppe puts an emphasis on the usage and importance of meta models for development of a domain specific language. Here definition of DSLs has been mentioned in the previous chapter although her work has a focus on definition of languages in general, and not so much the focus on DSLs as Fowler in [Fowler, 2010]. Selic has a similar take on language development as Kleppe has, but with a strong focus on the OMG standards for modeling and model transformation [Selic, 2009 and Selic, 2011].

To support her ideas about language creation she provides a definition of a meta model such as that a meta model models valid language expressions of a (domain) specific language. From her perspective a definition of a meta model is more powerful than the Backus Naur Form (BNF) as it has been mostly used by Fowler [Fowler, 2010]. Kleppe considers three necessary meta models for a language definition: abstract syntax model, concrete syntax model and semantic domain model.[66] Kleppe provides a formalized description of the base of the meta model concept.[67]

Graphs are the mathematical foundation of graphical models, which consist of nodes and edges (links) between the nodes. She mentions two types of graphs: directed, labeled graph and type graph. A (directed) labeled graph is an edge with direction between a source and a target node. Every node and every edge have a label. In addition, every edge has an index number. The combination of source node, target node, edge index and label is unique.

[65] [Parr, 2009], p. 279 ff.
[66] Her understanding of the semantic model is different from Fowlers semantic model in [Fowler, 2010], p. 159 ff.
[67] [Kleppe, 2009], p. 58 ff.

Directed graphs between nodes, which do not represent objects, but types are type graphs. Edges in such a type graph are relations between nodes and represent inheritance between nodes. A model is a type graph as defined above, constraint by nodes of various types. A model instance is a labeled graph (as explained above) with nodes and edges of types as defined in the corresponding type graph.

The validity of the nodes and edges and their relations in a model instance are defined by constraints, most important invariants. Invariants are logical expressions over a typed graph, which have to be fulfilled by a model instance. Other constraint types are: multiplities, bidirectionality, ordering, uniqueness, acyclic, unshared, redefinition, subset and union.

With such a formalized description of models, A. Kleppe explains the three meta models of a (domain) specific language.

The Abstract Syntax Model links together concrete syntaxes and semantic models.[68] It describes the underlying structure of a language with its concepts and relationships. This model is hidden, because it exists in the computer only. Any representation of the Abstract Syntax Model would be in a concrete syntax.[69] A sample for an Abstract Syntax Model would be the common ground between UML model elements and its XMI representation, but not the concrete mapping in transformation rules. B. Selic basically adds details to this definition by showing the concrete aspects of OMG's Meta Object Facility (MOF) standard highlighting concepts for syntax definition such as generalization relationship, composition relationship, association relationship, the class concept, packages for modularization and constraints.[70] The base elements of the important modeling constraints are propositions, which are either true or false. Predicates are propositions with variables in a particular range in their values. Constraints of a model are then predicates, which have to be true for a particular model within the context of a class. OMG's Object Constraint Language (OCL) is the relevant standard for expressing such constraint in the context of UML and MOF models in a declarative form with first order logic.[71]

[68] [Kleppe, 2009], p. 75 ff.
[69] See [Frankel, 2003], p. 104 for „representation of an abstract syntax tree" using UML concrete syntax
[70] For details see [Selic, 2011], p. 95 ff.
[71] [Selic, 2011], p. 107 ff.

An abstract syntax can address business problems (vertical orientation) or can be pattern-oriented (horizontal orientation). Vertical languages address a particular business topic such as environmental modeling for a particular environment. Horizontal languages address a common pattern over multiple business topics. An example can be a language for GIS patterns in business topics such as Geomarketing or Disaster Recovery support. In this case an abstraction will be taken from a particular problem to a common understanding, yet a pattern.

The second meta model of a language, the concrete syntax model represents a language to a human being, either textual or graphical.[72] A Concrete Syntax Model defines a set of symbols like an alphabet and parsing rules of the language. The parsing rules more specifically contain rules for parsing of symbols for valid combinations of symbols, abstraction rules from concrete syntax to abstract syntax and binding rules for separation of concrete syntax constructs, which are represented by the same abstract symbol. A DSL user would see the concrete syntax in the diagrams of a modeling language such as UML. The corresponding standard of the OMG for such data definition of graphical syntax and diagram interchange is currently not perfectly implemented in the different modeling tools.[73]

The Semantic Model provides the meaning of a language.[74] It is the base for communicating an understanding of a language between human beings in order to come from a subjective understanding to a common understanding. According to Kleppe the Semantic Model kicks in at runtime of a computer or more general a language communication when linguistic symbols are mapped to the underlying concept of a topic.

The Semantic Model can be described in a concrete syntax (a language, which is already defined) by annotations using mathematic objects, a reference implementation, translation into a known language or operational state transition systems, which describe the states of coming from a situation without semantic knowledge to the actual semantic meaning. The semantics of a domain define the computational model at runtime and are represented in a domain model. OMG's UML Foundation would again be the relevant standard.[75] For B. Selic, the semantic model needs to be

[72] [Kleppe, 2009], p. 93 ff.
[73] [Selic, 2011], p. 147 ff.
[74] [Kleppe, 2009], p. 131 ff.
[75] [Selic, 2011], p. 155 ff.

defined upfront and the abstract and concrete syntax models have to be based on this domain model.[76]

Kleppe leads the idea further by discussing the combination of different languages.[77] Considering the fact that domain specific languages have a rather narrow purpose different languages are needed to enable interaction between domains. If we consider horizontal languages, there could be particular languages in software development for user interfaces, business logic and data access. Well defined language interfaces are required to define which language constructs of a particular language are available to other languages. The Abstract Syntax Model or a subset of it would be the interface between different concrete syntax models of different languages.

4.2.4 Graphical Language Development

Kleppe's work has shifted the focus towards visual DSLs, so called Domain-Specific Modeling Languages (DSML). In a GIS context, DSMLs could be considered a further, more scientific language development of the visual GIS modeling attempts of Albrecht [Albrecht, 1996] in the mid-1990's. The focus of the visual GIS modeling was clearly on the enhancement of the user experience and not so much on the correct specification of a language. The DSML approach is often inspired by standard general purpose modeling languages such as UML or MOF.[78] This can be related closely to Kösters' approach of defining GIS specific language constructs in GeoOOA [Kösters, 1996] based on the OOA notation at a time before UML became a common modeling standard.

B. Selic has provided an introduction to this topic [Selic, 2007] and has given an overview on DSML [Selic, 2009][Selic, 2011]. From his perspective he considers (graphical) modeling languages as more powerful than textual languages. They can express complex aspects in a way that can be easier understood by an audience. Besides the communication aspect, DSMLs are supposed to improve software development and therefore are often used for code generation or transformations in other, less abstract models.

76[Selic, 2009], p. 315
77 [Kleppe, 2009], p. 176 ff.
78 UML and MOF are trademarks of the Object Management Group (OMG)

But DSMLs come with some challenges. Creation and usage of the languages are usually not possible without using a particular language workbench which supports language definition and language usage with an editor and then with transformation and generation capabilities.[79] For that reason the learning curve for such language workbenches is quite high, for both, DSL developers and users. In addition, there are difficulties in the definition of concrete syntaxes.[80] Different language workbenches support different graphical elements and without an additional textual language not all required expressiveness can be reached.

Selic sees three methods of defining DSML's:[81]

- Refinement of an existing modeling language by specializing general constructs to present domain specific concepts.[82]
- Extension of an existing modeling language by supplementing the language with domain specific constructs. So, parts of the DSML are within the existing language and parts are outside.
- Definition of a new modeling language. Such a language has the highest expressiveness, but also the highest costs.[83]

Because of the high costs of a complete separate language, Selic is supporting the method of refinement. He very much supports OMG's UML standard modeling notation as a meta model for a DSML and the usage of UML profiles as the refinement mechanism using stereotypes and modeling libraries for DSML definition and definition of domain specific viewpoints.[84]

B. Trask and A. Roman base their take on Model Driven Development (MDD) and Domain Specific Languages (DSL) on the premise that the current programming languages are not powerful enough to manage the complexity of software product lines [Trask/Roman, 2011]. Their tutorial about DSL on the MODELS 2011 conference goes along the words "Model", "Driven" and "Development". *Models* are simplified representations of a system, basically the abstraction of reality at a particular level. Such abstractions create new vocabulary, relations and

[79] On language workbenches see [Fowler, 2005] and [Fowler, 2010], p. 129 ff.
[80] For more details see [Selic, 2011], p. 147 ff.
[81] [Selic, 2009], p. 299
[82] This can be very much related to Fowlers concept of an Internal DSL in [Fowler, 2010], p. 67 ff.
[83] Issues and experiences from a practical example can be found in [Wienands/Golm, 2009]
[84] [Selic, 2009], p. 308

combinations and therefore are language development. The word *Driven* is used in a sense of urgency of causing or guiding something, while *Development* is actually the creation of something that can be used or even executed. Model Driven Development (MDD) for them is programming with models using a particular language, graphical editor and a generator. From their perspective, MDD and DSLs are equivalent. A DSL is just the model concept of a specific domain. As practical tools they are referring to the Eclipse Modeling Project with its various sub projects for model creation, transformation and generation. Unfortunately, although they seem to bring significant practical experience to the table, the tutorial has much of a marketing event, rather than serious science, especially in comparison of different approaches [Selic, 2011].

4.2.5 Conclusions

M. Fowlers approach on (domain specific) languages is clearly the enhancement of programming using the Backus-Naur-Form of language grammar definition. Especially the concept of an Internal DSL is not compatible with the model-driven approach of separating business domain aspects and technological aspects of an implementation. The focus of visual (domain specific) languages is using the meta model concept and much more towards improved communication with domain experts and usage of model-driven engineering approaches for the software development.

His take on DSLs in [Fowler, 2010] is very similar to T. Parr in [Parr, 2009]. But Fowler has a clear focus on DSLS and software building, while Parr has a focus on language creation and patterns in language applications in a more general way. Both have a strong focus on textual languages, while A. Kleppe in [Kleppe, 2009] and B. Selic in [Selic, 2009] see the concept of meta models and domain specific modeling languages as leading their research.

There seem to be various shortcomings and less depth in Kleppe’s explanations about the meta model approach compared to Selic, who adds much more depth to this perspective, although the practical application of the OMG concepts in the software industry are not so far as a reader of his documents might think. It is interesting to compare these approaches against the opinion in [Czarnecki/Eisenecker, 2000][85] where DSLs are seen in the domain specific implementation only.

[85] [Czarnecki/Eisenecker, 2000], p. 21ff.

4.3 Model Driven Architecture and Development

4.3.1 Introduction to MDA

Continuous challenges in the software industry are quality and longevity of software products in line with costs of development and maintenance.[86] In order to cope with these challenges the software industry is discussing the option of increasing the abstraction level in software development by using standardized modeling for specification, architecture documentation and communication as well as for directly developing artifacts from models.[87] This approach has different naming, which address a different focus of the same issue:

- MDA – Model Driven Architecture: A standard of the OMG[88] which is UML based and addresses model interoperability between tools and model standardization for various application domains. It provides the base terminology for MDSD
- MDSD[89] – Model-driven Software Development: Describes a process of software development with a focus on generation of software artifacts from models. The goals of this approach are increasing development speed and quality through automation, separation of concerns by reduction of redundancies and managing of complexity through abstraction and formalization
- MDD – Model Driven Development: Same as MDSD, but without the narrow focus on software development
- MDE – Model Driven Engineering: Does not so much describe the process, but more the methodology of using a sound engineering approach supported by models.

In the definition by [Stahl/Völter, 2005][90] a model is an abstract representation of the structure of a system, its function and behavior.

A formal definition of a model's semantics in a formal syntax is required for a mapping of a model to a particular platform. A platform is in general just a more specific layer of a system

[86] See already [Frankel, 2003], p. 4
[87][Frankel, 2003], p. 32
[88] For details see the relevant webpage of the Object Management Group (OMG): http://www.omg.org/mda and the related specification http://www.omg.org/mda/specs.htm
[89] [Stahl/Völter, 2005], p. 4 and p. 13
[90] [Stahl/Völter, 2005], p. 18 ff.

compared to a platform independent model. Platform independence and a platform specific view can be considered on different abstraction layers. A technology such as JEE can be considered a (technical) platform, while the pure business object model of a domain is platform independent from this particular implementation technology platform. The most specific platform model would be the source code or the executable of a particular system.

The MDA specification describes the separation of platform independent (PIM) and platform specific models (PSM). A platform definition model (PDM) contains the meta model of what an actual platform constitutes. So implicitly this describes the separation between platform specific and platform independent. Based on the two terms model and platform, the MDA concept contains the transformation between PIM and PSM using automated execution of mapping or refactoring rules.

The core building blocks of MDA according to [Stahl/Völter, 2005][91] are UML as defacto standard modeling notation, MOF[92], which describes the meta model of UML, and XMI[93] as interoperability standard for model exchange.

UML brings advantages such as separation of concrete and abstract syntax, a built-in extension mechanism,[94] platform independence and the design by contract pattern through definition of formal semantics using OCL.[95] But even in the version UML 2.x important disadvantages such as a missing viewpoint concept, limited expressiveness of the extension mechanism of profiles and a vague interoperability standard for diagrams remain.

[91] [Stahl/Völter, 2005], p. 240 ff.
[92] MOF = Meta Object Facility: Meta model of UML, which is using UML as concrete syntax. Standard of the OMG. A partial MOF implementation is the Eclipse Modeling Framework (EMF).
[93] XMI = XML Metadata Interchange: Specification of the mapping of MOF to XML. Standard of the OMG
[94] The PIM can use UML profiles as concrete syntax for modeling of specific semantics [Frankel, 2003], p. 203 ff. This leads to a nice bridge to the DSL concept. More details will follow in subsequent chapters.
[95] OCL = Object Constraint Language: Standard of the OMG

4.3.2 Meta model levels

The concept of meta modeling is the most basic and most important concept of model-driven architecture and development and their concept of automated model transformation. A meta model contains statements about models and modeling defining their abstract foundation, possible modeling constructs and their relations and the borders of a particular kind of model [Stahl/Völter, 2005]. For that purpose a meta modeling language is used as described in a *meta meta model*. Models are instances of a meta model using a concrete syntax for modeling as user interface between abstract syntax and model and respective model users.[96]

[Frankel, 2003] gives a stringent explanation of the meta model levels in the context of MDA and UML.

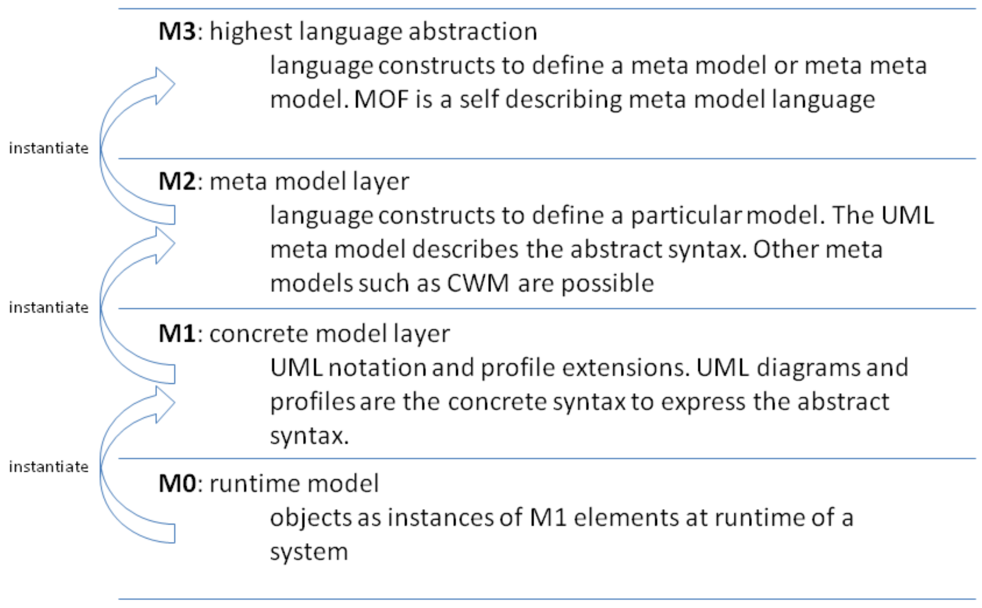

M3: highest language abstraction
language constructs to define a meta model or meta meta model. MOF is a self describing meta model language
M2: meta model layer
language constructs to define a particular model. The UML meta model describes the abstract syntax. Other meta models such as CWM are possible
M1: concrete model layer
UML notation and profile extensions. UML diagrams and profiles are the concrete syntax to express the abstract syntax.
M0: runtime model
objects as instances of M1 elements at runtime of a system
instantiate
instantiate
instantiate

Figure 8: OMG concept of meta model levels (Source: Frankel, 2003, p. 105 ff.)

The concept of meta modeling as formally described in the OMG's 4 meta model levels can be found in [UML241I].[97]

[96] See also [Kleppe, 2009]
[97] [UML241I], chapter 7.10

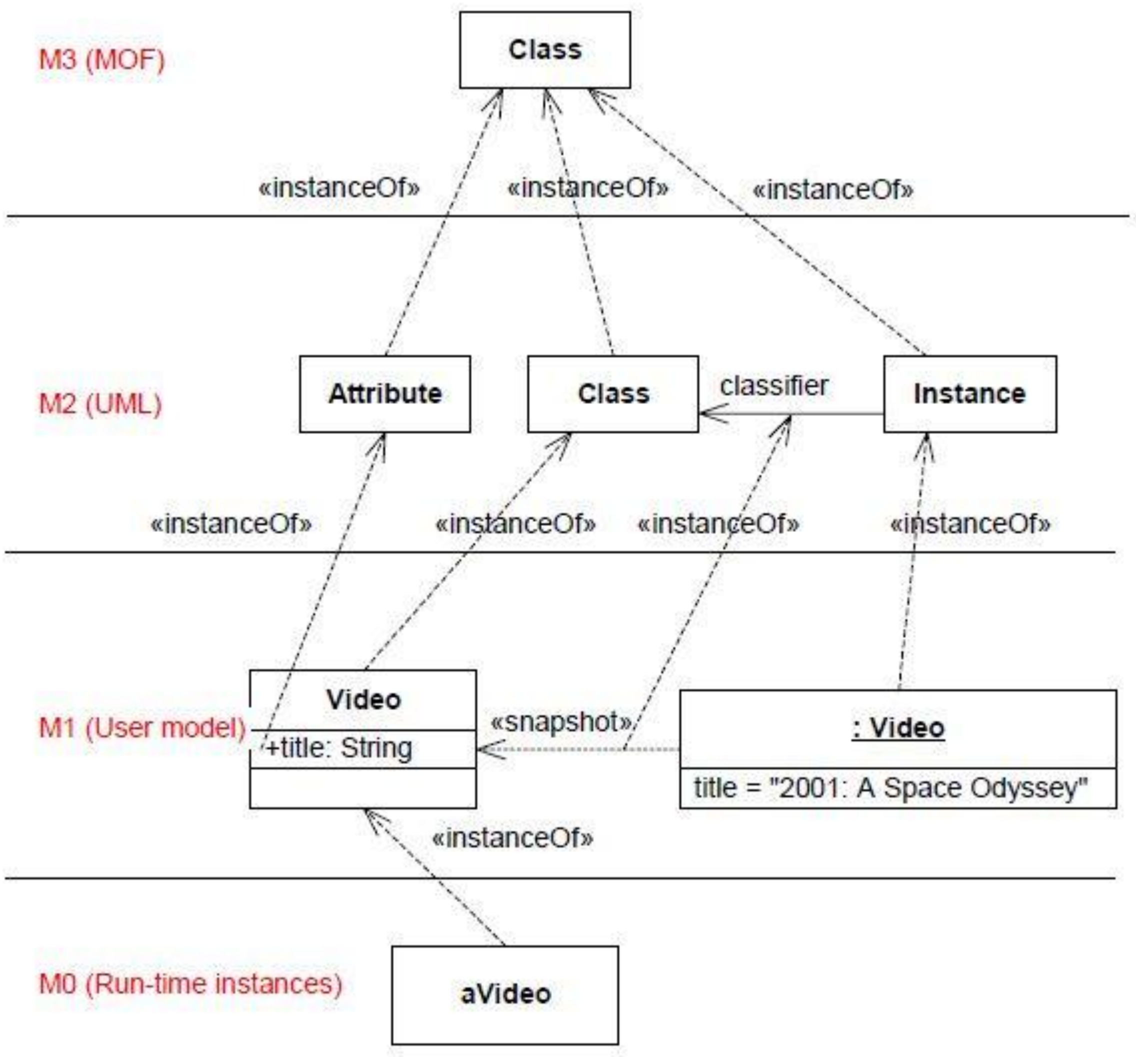


Figure 9: Sample of OMG meta model levels (Source: [UML241I], chap. 7.10, fig. 7.8)

The sample shows nicely the instantiation between the different model levels describing with one concept runtime behavior up to the abstract syntax definition.

As explained in a previous chapter, A. Kleppe [Kleppe, 2009] brings this concept together with the development and application of domain specific languages.

4.3.3 UML and its meta model

The UML (Unified Modeling Language) is a general purpose modeling language defined via a MOF-based meta model. This meta model describes the abstract UML syntax, specifies a notation (concrete syntax) and rules for the combination of notation elements. In addition, using an XML-based interchange format description, the compliance levels of tools to the abstract syntax can be specified on different levels. So the primary goal is interoperability between visual modeling tools via an agreed notation and semantics.

With version 2.x of UML, a set of design principles has been applied more consequently than before (version 1.x) such as a modular structure, which provides partitioning of the language, layering of the specification, a clearly defined extension mechanism of profiles and the reuse of specification elements. The Infrastructure package [UML241I] and the Superstructure [UML241S] represent the UML specification. The concept of language units has been introduced, so that not all constructs have to be used for each domain or application. Language units are tightly coupled modeling concepts within UML.

The infrastructure package contains the core specification and extension mechanisms along with a couple predefined primitive types such as Boolean or Integer for potential reuse. The superstructure package contains the user-level language constructs based on the infrastructure package. The user-level constructs are partitioned in packages as listed below:

- Common behaviors
- Classes
- Use Cases
- State Machines
- Interactions
- Activities
- Actions
- Composite Structures
- Components
- Auxiliary Constructs

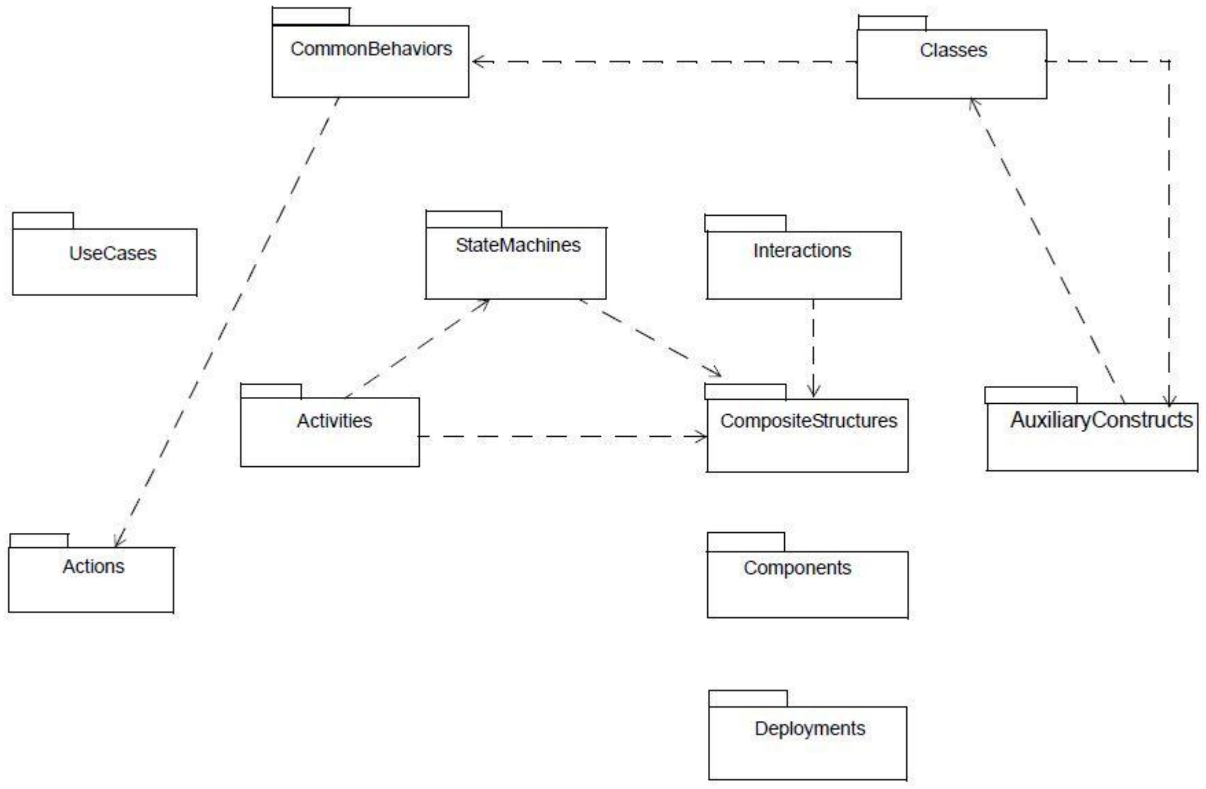


Figure 10: Top-level packages of UML Superstructure (Source: [UML241I], chap. 7.6, Fig. 7.5)

The package structure shows that there are dependencies between some packages. How far tools support or enforce these dependencies is up to the tool vendors and might be "measured" by the compliance levels of the tools.[98]

[98] For details see [UML241S], chapter 2.2

4.3.4 UML Profiles

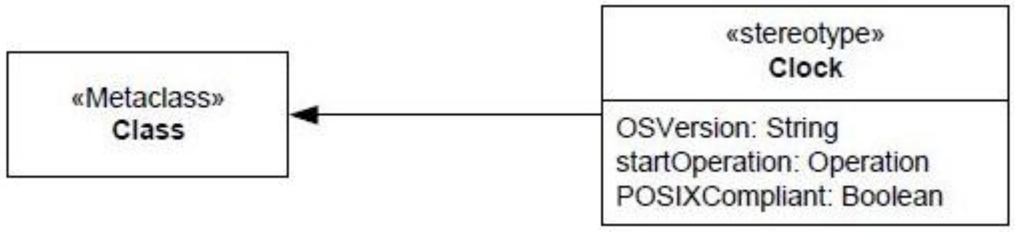
Since UML profiles are the built-in extension mechanism of UML and therefore the first choice for development of a DSL in a visual modeling context around UML the profile concept will be explained in more detail at this point.[99]

UML profiles provide the ability to adapt meta classes of the existing meta model in order to extend them beyond the semantics of the general purpose language notation.[100] With this mechanism it is possible to tailor a model to a particular platform, a business domain or specific methods. The UML profile mechanism does not allow the modification of existing meta models and cannot remove existing meta model constraints. New constraints can be added.

UML profile is a specialization of the package construct of the core constructs of the Infrastructure part of the UML specification. Best known is the Stereotype construct, which is a class that extends a meta class through extension rather than through inheritance/specialization. Stereotypes can be applied in a model to defined model elements through a keyword or through an icon. Stereotypes can have their own:

- properties using tag definitions
- constraints (OCL)
- Image/icon

Stereotypes support the concept of inheritance, but cannot be used by other stereotypes. They are always available as concrete syntax in combination with the meta class they extend.



[99] For details see [UML241I], chapter 12

[100] A more advanced, sometimes called heavyweight extension is the definition of a new meta class using MOF.

Figure 11: How to apply a stereotype (Source: [UML241I], chap. 12.1.9, Fig. 12.12)

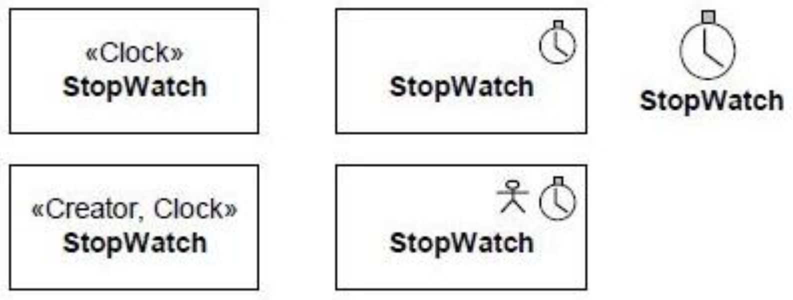


Figure 12: Sample of stereotype application (Source: [UML241I], chap. 12.1.9, Fig. 12.14)

The picture above shows the stereotype definition and its application to a particular model element. The Stereotype "Clock" has been defined with a particular keyword (Clock), an icon and three tags. The Stereotype can extend the meta class "Class". This means that the same stereotype cannot be applied to a different meta class as for example to a "Package" meta class. The stereotype tags are then available as properties to the extended class. An element can be extended by more than just one stereotype.

4.3.5 Model usage

After explaining about models, meta model standards and the ideas behind them, this chapter addresses the usage of models in the software development process.

The basic model usage according to MDA is the transformation between the abstraction layers of PIM and PSM with reproducible quality and automated validation against the meta model as last step before the actual creation of platform specific source code .101 So for example a business object class could be transformed in a relational model first before the generation of the actual DDL code or possibly Java code for the data access layer.

Other usages of transformation are according to [Gronback, 2009][102] model refactoring, model migration, model merge or transformation between different DSLs. Although this concept is part of the MDA standard, the specification does not describe the "Howto". The transformation topic is supposed to be supported by the QVT standard.[103] An alternative is the Atlas Transformation Language (ATL). Unfortunately the standardization in this area has not progressed as it would be required and the tool support is still very limited. For example the QVT standard is actually a combination of three different languages.

Due to these practical limitations and the fact that mostly software development is focusing on software creation, the model to model transformation is much less applied in practice than the actual source code generation from models. The goals of code generations are increasing efficiency of source code creation, potentially for multiple target platforms (e.g. mobile applications for iOS, Android and Windows) and reliable quality of the source code creation independent from the performance of a particular developer by better analyzability, earlier validation and better modularity on a higher abstraction level than the source code itself.

For code generation different techniques can be applied:[104]

- Templates and Filtering: First, a filter identifies the relevant part of the model and then defined variables will be replaced with values and output will be generated. In this

[101] [Stahl/Völter, 2005], p. 203 ff.
[102] [Stahl/Völter, 2005], p. 231 ff.
[103] QVT = Query/View/Transformation: Standard of the OMG since 2005
[104] [Stahl/Völter, 2005], p. 186 ff.

approach the abstraction level is low and it becomes complicated if the models are complex.

- Templates and meta models: In this case an input model will be parsed and the meta model will be instantiated as an abstract syntax tree. The abstract syntax tree will then be walked several times to determine information, which finally will be produced as output using a template.[105] The abstraction layer of this approach is much higher and can be applied to more complex environments than for the previously explained template usage
- Frames and slots: In this approach again an abstract syntax tree will be created after parsing the input model. Frames are code elements, which can be instantiated with its own attributes one or multiple times. The frames contain so called slots, which are kind of variables. These slots will be filled with concrete values determined from the abstract syntax tree or again other frames. The frame instance with be transformed into source code using templates.
- API-based generators: In this case a generator API will be called with parameters from an input model. This generator is usually bound to a particular programming language, which reduces the flexibility of generation for multiple target platforms.
- Inline generation: In this case the generated code part are embedded in the regular source code. A preprocessor as for example in C++ replaces the template elements in the code with actual source code. This comes very close to the concept of "Internal DSLs" as explained in [Fowler, 2010].[106]
- Code attributes: These are placeholders in manually written code and are programming language specific. They are used for interface generation (JEE) or documentation generation (e.g. JavaDoc).
- Code weaving: This is the (automated) combination of independent code snippets using combination rules via hooks.

The different code generation techniques support the goals of the generation approach fairly different. Code generation, which generates one source code class from one UML class are rather primitive and will certainly not support increased efficiency or better quality. The approach of

[105] For details see the previous chapter about DSL
[106] See previous chapter about DSL

using templates and a meta model is much more promising, but requires a rather complicated setup.

Another important usage of models in general and UML in particular is the documentation of architecture [Clements, 2010] and design of an application or system. The usage of UML or models in the design leads to the further usage in code generation as explained before. The goals of the architecture documentation are:[107]

- Education: for example for ramp-up of new people in an organization or team
- Communication: In order to get decisions by stakeholders
- Impact analysis: Architecture documentation is an important source of the impact analysis for an application in the future
- Construction: It explains the overall picture to the developers in order to show them where their individual work fits in the system

UML can be used for documenting an application structure using different views such as:

- Module view: Units of functionality (classes, packages, components) with module styles (decomposition, use, layers)
- Component and connector view: This explains how functional unit's work together using events, call-return and data flows. UML has just limited support for this topic.
- Allocation view: This view addresses the context of an application, its interfaces, deployment and installation

The behavior of an application can be documented using UML elements such as Use Cases, Use Case diagrams, Sequence diagrams, Activity diagrams, Communication diagrams and State Machine diagrams.

This architecture documentation is not just a one-off attempt of an implementation project, but has to be maintained and stay current over the complete lifecycle of an application/product.

[107] [Clements et. al., 2010], p. 9 ff.

4.3.6 Conclusions and further research topics

The standards of OMG about modeling provide a powerful set of specifications in order to address challenges in the software development process as of today. Even though UML has its limitations, with version 2.x it provides a standard which is well supported by various tools in order to put the OMG specifications and standards to practical use.

While reviewing the current developments and research in regards to domain specific languages and the concepts of MDA and MDSD it seems that both development strings complement each other and are already moving closer to each other. When reviewing research in recent years it becomes obvious that the MDA/MDSD approach cannot be thought without the DSL concept in mind and vice versa, especially when it comes to model transformation and modeling for specific business domains.

[Guerra, 2010] recommends considering model transformations as a development process of its own including analysis, design, implementation and validation using a transformation language which obviously is a DSL for the model transformation domain. According to [Kleppe, 2009] it could be considered a horizontal DSL. Another DSL is used in [Wagelaar, 2011] to provide a concept of modularizing model transformations using a virtual transformation machine and a particular transformation language.

[Lucio, 2010] addresses the automated validation of model transformation results using a DSL-based check of properties in the input and result model. [Ehrig, 2009] is building correctness and completeness checks in the transformation process itself using a triple graph grammar.

[Orejas, 2009] is applying the pattern concept to model to model transformations in order to apply transformation constraints for bi-directional transformations. This is a similar approach as in [Parr, 2010] about language development.

[Bagheri, 2010] is addressing the architecture documentation with models. He is researching the mapping of domain specific requirements and knowledge to architecture styles. The concept of architecture styles sounds very similar to the platform-specific concept of MDA. He is recommending a DSL for architecture modeling based on a meta model.

It almost seems like a Babylonian language issue. But looking at the research papers the development of languages always goes along with a lack in tool support. The transformation infrastructure is in almost all cases proprietary using research prototypes, in-house developed infrastructure or commercial products with proprietary concepts. An often used infrastructure is the Eclipse modeling Project and related subprojects. Although this comes close to a standard based tool infrastructure for DSL and model-driven development, it is based on Ecore and EMF, which is just a partial implementation of the MOF and UML standard. Due to the open source nature the further development is not always clear. When researching details at the Eclipse site, it becomes obvious that some projects have started and then stopped again or there are multiple competitive ideas. Although this dynamism is positive, a company which is looking to apply the DSL and model-driven concepts to a product line looking for a product lifecycle of multiple years might has difficulties relying on such an infrastructure.

5. Positioning business domain, GIS services and implementation technology

5.1 Introduction

Building software applications serves the purpose of supporting particular processes of a business or public domain. Such domains can be supported using different technology. Since domain knowledge usually is relatively stable over the long term and technology changes rather quickly, a separation of both is strongly recommended.

On the other side, technology and software vendors offer highly integrated and specific purpose applications. Such highly integrated solutions promise a quick return on investment for a particular implementation project.

But these tools often lock domain knowledge within the technological product and make it very difficult to extract it later in order to reuse it for different purpose or to replace just the technological solution for the same domain with a better product. Specialized software functionality serves very well one purpose, but not necessarily future demands in a particular domain.

Therefore, it is recommended to separate both, domain knowledge and technology in order to keep flexibility in adjusting technology to changing or new domain demands or to be able to support domain processes with better technology with reasonable change effort.

For this purpose it is important to consider the concepts of standardization and open extension mechanisms as explained in the chapters before, in this case to the GIS domain and services. This chapter positions and describes the “GIS domain” and its service character in the information technology.

5.2 Business domain definition

What is a *business domain*? Wikipedia provides a rather software oriented definition.[108] The description is relating to object oriented programming and objects in a business model. For the work in this document this definition cannot be used since it is too narrow to software development and IT aspects, especially since it does not fit with the explanation of *business model* in Wikipedia.[109] In this explanation the business model focuses on a solely economic definition of how an organization creates, delivers and captures value in the market place addressing issues such as business purpose, offerings, strategies, markets, etc.

[Maxted, 2008] considers business modeling from a business analyst perspective with IT focus. A business model describes the business and its intentions as a simplified model of the real world complexity. According to her, the business model helps to understand the business domain and to capture relevant aspects of its meaning and focus. This helps to communicate efficiently with IT project stakeholders and can lead to a correct implementation in related IT systems.

In order to set a common context for this document, the following definition of *business domain* as it can be found in [Czarnecki/Eisenecker, 2000][110] has been applied in the context of information technology, software development and software development methodology, but independent from a particular technical platform.

A vertical view of a business domain captures a narrow aspect of the real world, but explains it in detail. It basically describes the whole value chain of a business. A sample for such a vertical view would be the environmental modeling of the tuna population in relation to environmental programs to protect the tuna population in a certain area.

A horizontal domain view addresses an aspect of the real world, which is relevant in multiple vertical views such as a particular methodology. For example, the methodology of spatial analysis can be applied to environmental modeling of soil nutrient concentration and to modeling of spatial socio-economic factors in a Geomarketing analysis of a city neighborhood.

[108] http://en.wikipedia.org/wiki/Business_domain (14.01.2012)
[109] http://en.wikipedia.org/wiki/Business_model (14.01.2012)
[110] [Czarnecki/Eisenecker, 2000], p. 20

5.3 The "GIS domain" as a horizontal service

Geographic Information Science and Systems (GIS) is about modeling spatial aspects of the real world. GIS has not a value for itself or is not about creation of a "new thing", but in the support of decision making and problem solving in various domains. The topic where GIS can provide a supporting function has to exist somewhere in space with spatial data, which can be analyzed by mapping, measuring, monitoring, modeling and managing [Longely et. al., 2011].[111]

Samples where GIS provides a supporting function are support of:

- Disaster relief activities such as analysis of degree of disaster in neighborhoods in Haiti after the latest Earthquake.
- Selection of retail branch locations for banks or supermarkets in neighborhoods by analyzing the socio-economic distribution in the area.
- Optimizing routing information for cargo drivers.

In general GIS provides supporting services in different areas such as:

- Government and Public Service
- Business and Service Planning
- Logistics and Transportation
- Environment

The usage of GIS started back in the 1960's when only national and federal organizations could afford GIS systems. Today GIS applications and functions are used in communities, by local and regional government, too.[112] Although the GIS usage is much broader today than back in its beginnings, the various government organizations are still the biggest user group of GIS. It supports many aspects of top-down decision making in government, but more and more supports participation of citizens. GIS is then used for inventory of resources and infrastructure, planning of transportation, public service delivery planning, management of land development and decision making for public health safety, and welfare by monitoring health risks, managing public

[111] [Longely et. al., 2011], p. 45
[112] [Longely et. al., 2011], p. 46 f.

housing stock and distribution, crime tracking and geodemographics for issue prevention activities.

In business and service planning spatial information are used at different decision levels:[113] operational, tactical and strategic. Operational decisions are about "Where to apply a short-term service" such as firefighters in the event of a flooding. Tactical decisions include the "Where to allocate resources" such as stock flow management of retailers or waste collection tours in a city. Strategic decisions are of the kind of "Where to locate a new service" or where to build a new factory. In all these decisions, the spatial information supplements other non-spatial information such as financials or available resources.

Logistics and Transportation are about shipping and transportation of goods between places using an infrastructure.[114] While the infrastructure changes only slowly over a longer period of time the dynamic elements in transportation are the amount of goods/items/people and available resources in relation to the available (static) infrastructure. With GIS it is possible to optimize routes and schedules and to track resources in order to reduce time and cost of transportation services. While a company such as UPS can optimize delivery routes, the UPS customer can check the whereabouts of a particular package from its computer at home using GIS technology. Although originally GIS was used for static infrastructure aspects such as planning of new streets or train routes, the monitoring of dynamic aspects becomes more important due to improvement and wider availability of technology such as GPS.

The usage of GIS in environmental questions is a mature discipline, especially if related to government activities.[115] One aspect is the land management of a particular region. Since the land resource is limited and cannot be easily extended, GIS helps to make best use of it for agriculture, recreational areas and urban development. Monitoring of land use change is another topic, for example how green or woodland changes to industry land use. Technology like remote sensing is of great use for that purpose. A rather new application is the usage of GIS for simulation of scenarios on a global scale such as the impact of urbanization on the environment or trends such

[113][Longely et. al., 2011], p. 51 ff.
[114] [Longely et. al., 2011], p. 60 ff.
[115] [Longely et. al., 2011], p. 66

as climate change. The information determined from such analysis can be provided again for decision making regarding the identified issue.

The development of GIS in recent years goes more and more in the direction of “location intelligence” in the background rather than plain map presentation. The usage of geographic information becomes more and more driven by business workflows rather than GIS technology.[116]

This categorization of GIS in the “daily business” shows that GIS provides supporting service and can be considered a horizontal domain. GIS provides technology and methodology, which can be applied for various purposes.

[116] For details see [Strobel, 2005], p. 41

5.4 Modeling a domain

After the definition of the *business domain* term and clarification that GIS can be considered a horizontal business domain this chapter provides the concept of how to apply modeling to a business domain best.

As defined earlier, a business domain is a certain problem space to be addressed by an IT system. According to [Fowler, 2010][117] a business domain consists of a web of interconnected objects, each representing an individual "thing" of a problem domain.

P. Oldfield in [Oldfield, 2002] applies domain modeling to the enterprise level. For each enterprise in a particular business domain the business model is the same.[118] A domain model according to him contains objects and responsibilities that do not change too frequently. So for him a domain model is independent of a particular use, but rather supports reuse of the domain aspects in different usage scenarios.[119]

The core concept of a domain model is abstraction. The UML "class" concept represents the concept of abstraction of individual entities by reduction of individual entities to the essence of a concept. An abstraction has assigned responsibilities: knowledge and behavior and abstractions might have relations to each other. Since knowledge can be considered to be data and properties and behavior can be considered as operations the two concepts can be represented as UML class attributes and operations. But some behavior is not in the responsibility of just one abstraction class, but occurs only in relation of more abstraction classes. Rules and constraints of an abstraction could be modeled using informal textual description or formal description via OCL, which is more precise than simple text. But it might not be easily readable for all model users. This could affect the communication of domain expert, domain modeler and model user about the model.

Interesting is Oldfield's discussion about "pure" and "non pure" domain modeling. Pure domain modeling relies solely on the input of a domain expert, while in the non-pure form, domain facts can be determined by simulation by non-experts.

[117] [Fowler, 2010], p. 116
[118] [Oldfield, 2002], p. 3
[119] [Oldfield, 2002], p. 11

Although Oldfield considers UML not perfect for domain modeling due to the lack of alternatives, he is applying the notation to its domain concept.[120] Considering the previous chapter about models, profiles with stereotypes might come handy to help with the UML notation for domain modeling purpose. [Evans, 2004] recommends a common language for all stakeholders when modeling and communicating about a domain.[121]

While Oldfield recommends a clear separation of domain model and design model in order to support their independence from a particular usage, Evans recommends to keep them together since otherwise the domain model will "age" and over time differences between domain and design model will occur.[122] Evans' approach for separation of concerns is the layering of the model.[123]

The author considers the approach of Oldfield more valid. Applying concepts of MDA and automated model transformations would help to keep different models aligned. Therefore, the author will separate platform independent (domain) model and platform specific (design) model in the following practical oriented chapters.

[120] [Oldfield, 2002], p. 7 ff.
[121] [Evans, 2004], p. 24 ff.
[122] [Evans, 2004], p. 47 ff.
[123] [Evans, 2004], p. 67 ff.

6. A GIS domain model with meta model and DSL

6.1 Introduction

The following GIS meta model for GIS application modeling is based on the UML general purpose meta model and based on the ISO General Feature Model.[124] The approach of this document is to use standards as much as possible even if they do not fit perfectly for every aspect of the GIS domain. Due to the fact that standards in general provide more advantages, such as support with standard UML tools, the author is rather looking for a workaround of a shortcoming of a standard than developing a proprietary alternative which would be costly in tool support, future maintenance and general perception by potential users. This approach differs from other attempts which use a GIS specific tool for modeling their data and operations.

This follows the principle of separating domain knowledge from technology or even a particular vendor product as introduced by the MDA standard. The GIS domain is seen as a computational independent or at least platform independent model. It could be transformed and put to use in different application platforms such as ArcGIS[125] data catalogue, Oracle Spatial[126] or ArcGIS model builder for spatial operation modeling.

While the ISO General Feature Model covers the aspects of geographic data and their structure and topologies, an extension is required to address the following GIS aspects as identified in chapter one of this document:

- GIS operations: Modeling of behavioral aspects and operations for processing geographic data. This goes beyond the object oriented concept of behavior of an object, but addresses the usage and manipulation of GIS data.
- GIS presentations: Modeling of view aspects for representation of results of GIS operations such as in maps or object attribute lists. For this purpose, aspects such as geographic reference system, zoom or map scale or target audience and GIS operator are required.

[124] For details see ISO 19109, Clause 7
[125] ArcGIS is a product and trademarked by ESRI Inc.
[126] Oracle Spatial is a product and trademarked by Oracle Corporation.

As introduced by [Lisboa et. al., 2010], conventional data elements which do not directly have special aspects and network elements are considered in the meta model too.

Similar to the GeoFrame framework the meta model can be separated in three levels which are here called meta level, abstract level and representation level.

- Meta level: This level contains the meta classes ConventionalType, GIS_FeatureType, GIS_Operation and GIS_Presentation and secondary meta information about their properties.
- Abstract level: This level contains abstract model elements, which specify the meta classes in more detail, but do not exist in actual instances in the real world. They are required for categorization purposes.
- Representation level: This level contains the model elements used for modeling real world objects and behavior.

6.2 GIS meta model

The core of the GIS meta model are the GIS meta classes:

- GIS_FeatureType (based on GF_FeatureType in ISO 19109)
- GIS_Operation: Spatial operations using the GIS_FeatureTypes and
- GIS_Presentation: Presentation of GIS_FeatureTypes and results of GIS_Operations

and their relations to each other, especially GIS_SpatialTopology.

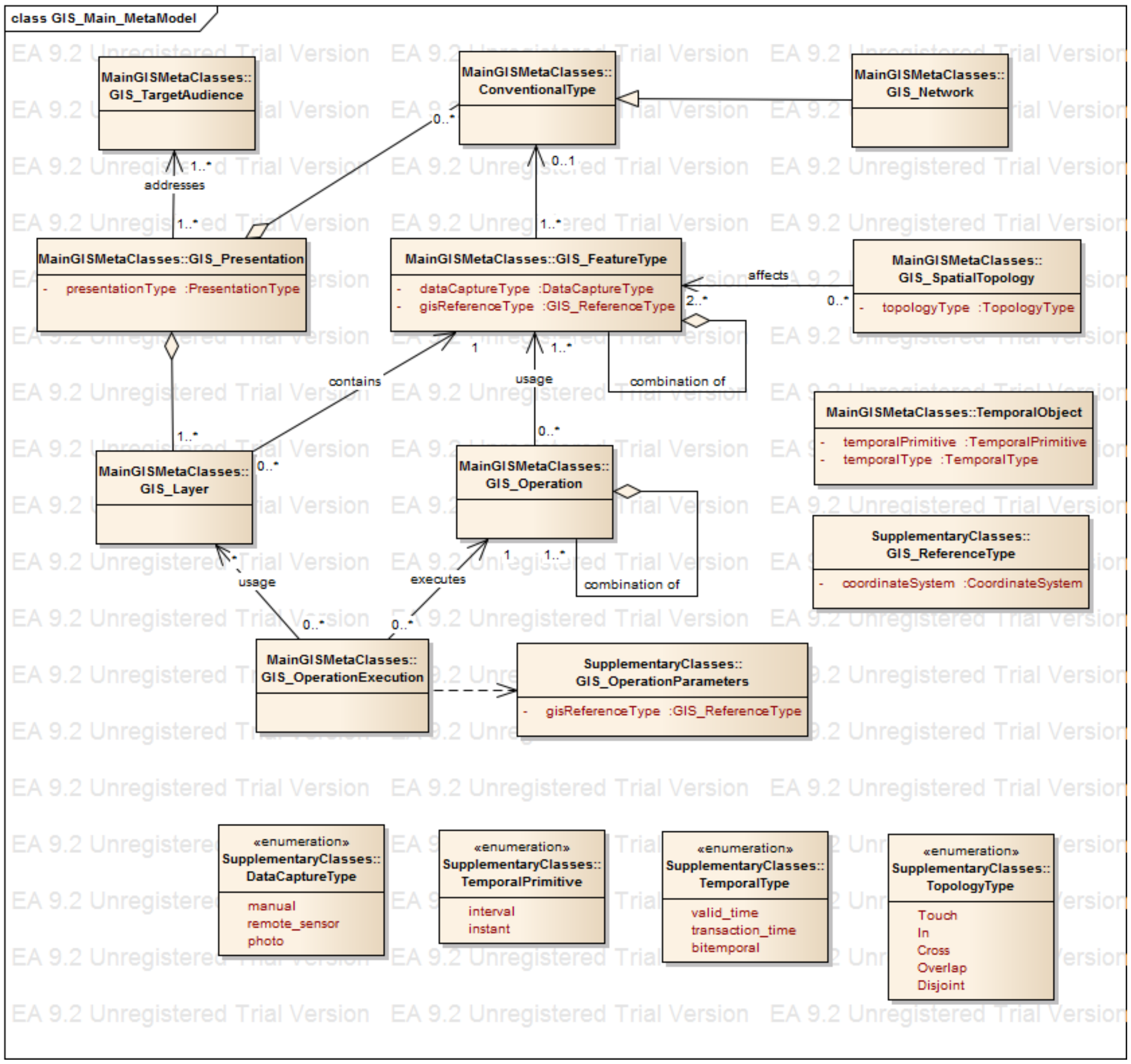

class GIS_Main_MetaModel
MainGISMetaClasses::GIS_TargetAudience
MainGISMetaClasses::ConventionalType
MainGISMetaClasses::GIS_Network
MainGISMetaClasses::GIS_Presentation
- presentationType :PresentationType
MainGISMetaClasses::GIS_FeatureType
- dataCaptureType :DataCaptureType
- gisReferenceType :GIS_ReferenceType
MainGISMetaClasses::GIS_SpatialTopology
- topologyType :TopologyType
MainGISMetaClasses::TemporalObject
- temporalPrimitive :TemporalPrimitive
- temporalType :TemporalType
MainGISMetaClasses::GIS_Layer
MainGISMetaClasses::GIS_Operation
SupplementaryClasses::GIS_ReferenceType
- coordinateSystem :CoordinateSystem
MainGISMetaClasses::GIS_OperationExecution
SupplementaryClasses::GIS_OperationParameters
- gisReferenceType :GIS_ReferenceType
addresses
contains
usage
combination of
affects
executes
1..*
0..*
0..1
1
2..*
«enumeration» SupplementaryClasses::DataCaptureType
manual
remote_sensor
photo
«enumeration» SupplementaryClasses::TemporalPrimitive
interval
instant
«enumeration» SupplementaryClasses::TemporalType
valid_time
transaction_time
bitemporal
«enumeration» SupplementaryClasses::TopologyType
Touch
In
Cross
Overlap
Disjoint

Figure 13: GIS meta model - main view

The status of a GIS_FeatureType or a spatial topology can vary over time. For this purpose the meta class *TemporalObject* was added to the GIS meta model. This class has two attributes that characterize temporal information. One of these attributes indicates the temporal type (validity time, transaction time or bitemporal time), whereas the other defines the used temporal primitive type (instant or interval). There are two enumerations (*TemporalType* and *TemporalPrimitive*) for the possible values of these attributes.

Each GIS_FeatureType and each GIS_OperationExecution has a GIS_ReferenceType such as a coordinate system. The DataCaptureType is an attribute of GIS_FeatureType and specifies the data origins and how they have been captured.

The ConventionalType might be or might not be related to a spatial type. Also a combination can be used in a GIS_Presentation.

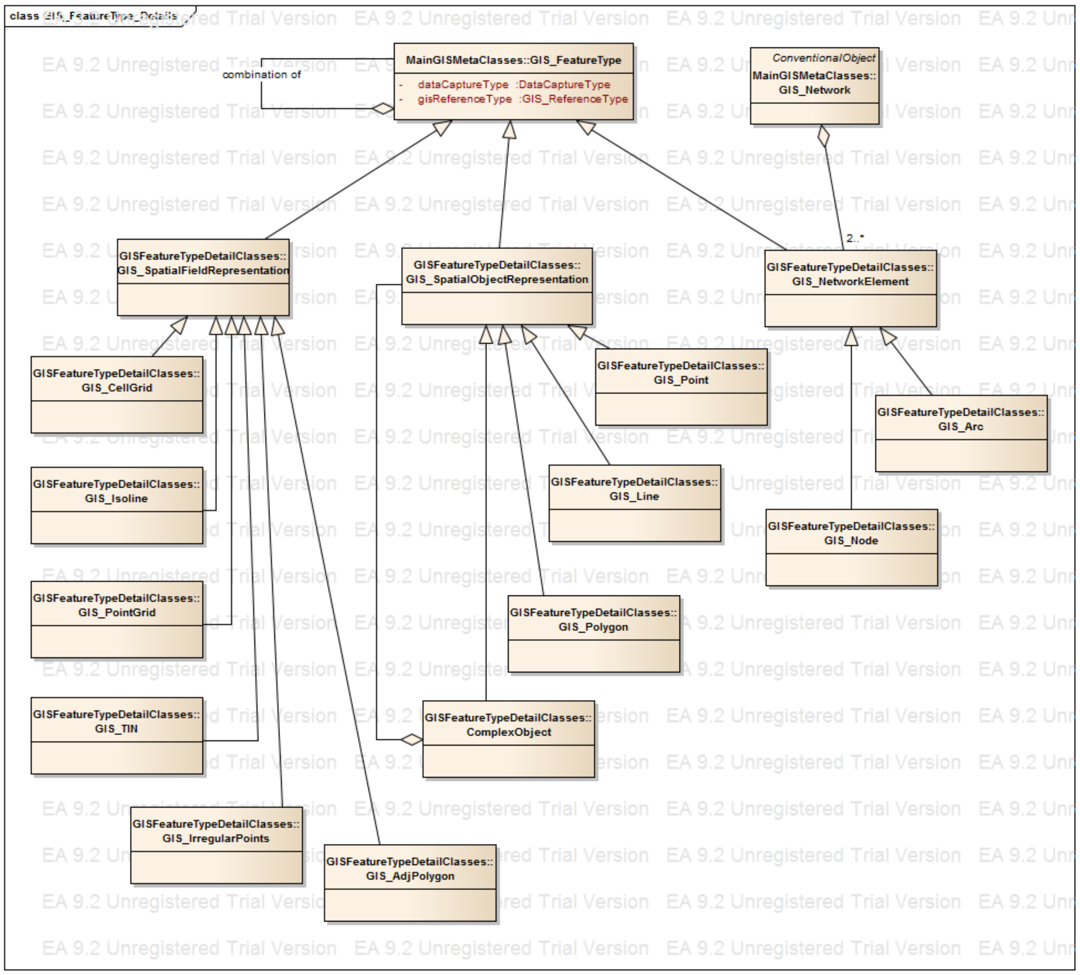

class GIS_FeatureType_Details
combination of
MainGISMetaClasses::GIS_FeatureType
- dataCaptureType :DataCaptureType
- gisReferenceType :GIS_ReferenceType
ConventionalObject
MainGISMetaClasses::GIS_Network
2..*
GISFeatureTypeDetailClasses::GIS_SpatialFieldRepresentation
GISFeatureTypeDetailClasses::GIS_SpatialObjectRepresentation
GISFeatureTypeDetailClasses::GIS_NetworkElement
GISFeatureTypeDetailClasses::GIS_CellGrid
GISFeatureTypeDetailClasses::GIS_Point
GISFeatureTypeDetailClasses::GIS_Arc
GISFeatureTypeDetailClasses::GIS_Isoline
GISFeatureTypeDetailClasses::GIS_Line
GISFeatureTypeDetailClasses::GIS_Node
GISFeatureTypeDetailClasses::GIS_PointGrid
GISFeatureTypeDetailClasses::GIS_Polygon
GISFeatureTypeDetailClasses::GIS_TIN
GISFeatureTypeDetailClasses::ComplexObject
GISFeatureTypeDetailClasses::GIS_IrregularPoints
GISFeatureTypeDetailClasses::GIS_AdjPolygon

Figure 14: GIS meta model - GIS_FeatureType details

The diagram above shows the breakdown of the GIS_FeatureType in different sub-types representing different spatial data types and shapes including the network type.

The different spatial operations behind GIS_Operation represent the twenty spatial operations as identified by [Albrecht, 1996] and mentioned in chapter 4.1.2 of this document. They could be combined to perform any kind of spatial operation.

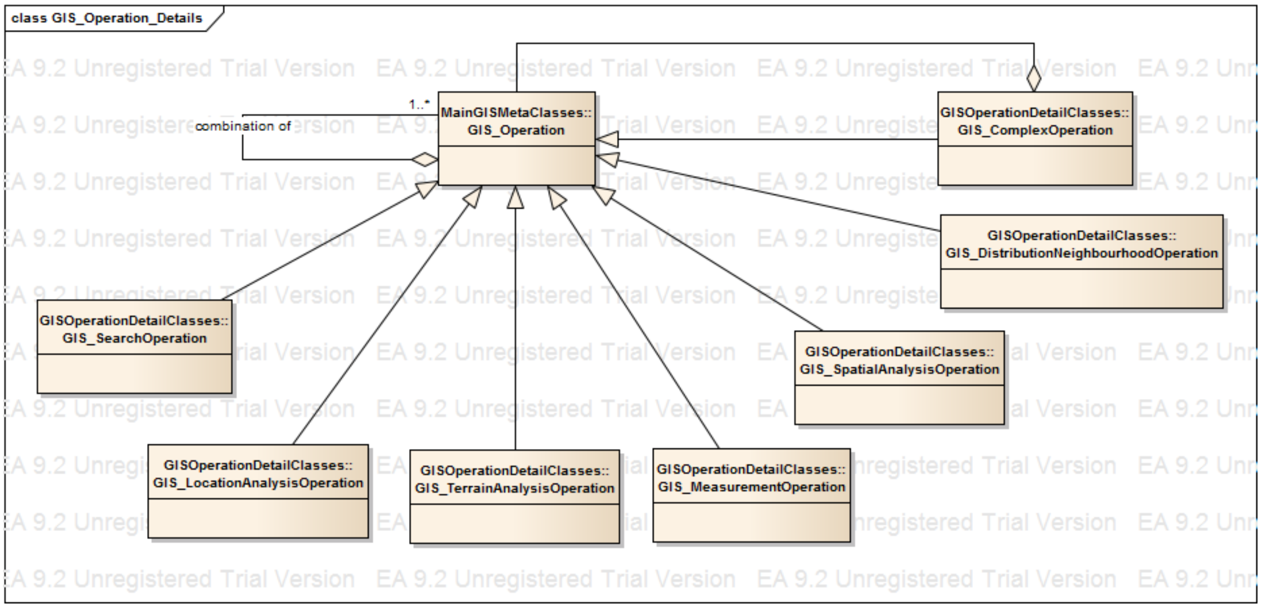

class GIS_Operation_Details
1..*
combination of
MainGISMetaClasses::GIS_Operation
GISOperationDetailClasses::GIS_ComplexOperation
GISOperationDetailClasses::GIS_DistributionNeighbourhoodOperation
GISOperationDetailClasses::GIS_SearchOperation
GISOperationDetailClasses::GIS_SpatialAnalysisOperation
GISOperationDetailClasses::GIS_LocationAnalysisOperation
GISOperationDetailClasses::GIS_TerrainAnalysisOperation
GISOperationDetailClasses::GIS_MeasurementOperation

Figure 15: GIS meta model - GIS_Operation details

The following picture is just a sample of the actual Search Operations category according to [Albrecht, 1996].

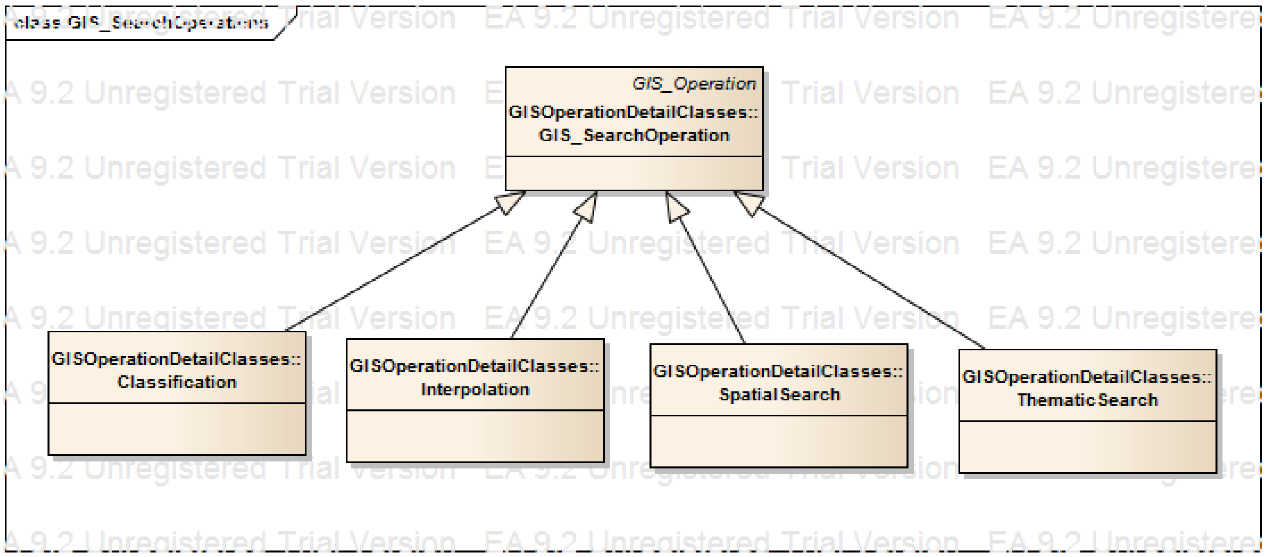

class GIS_SearchOperations
GIS_Operation
GISOperationDetailClasses::GIS_SearchOperation
GISOperationDetailClasses::Classification
GISOperationDetailClasses::Interpolation
GISOperationDetailClasses::SpatialSearch
GISOperationDetailClasses::ThematicSearch
EA 9.2 Unregistered Trial Version

Figure 16: GIS_Operation details - Search Operation details

The presentation aspects of the meta model could be represented by different presentation types such as maps or a simple result list of attributes.

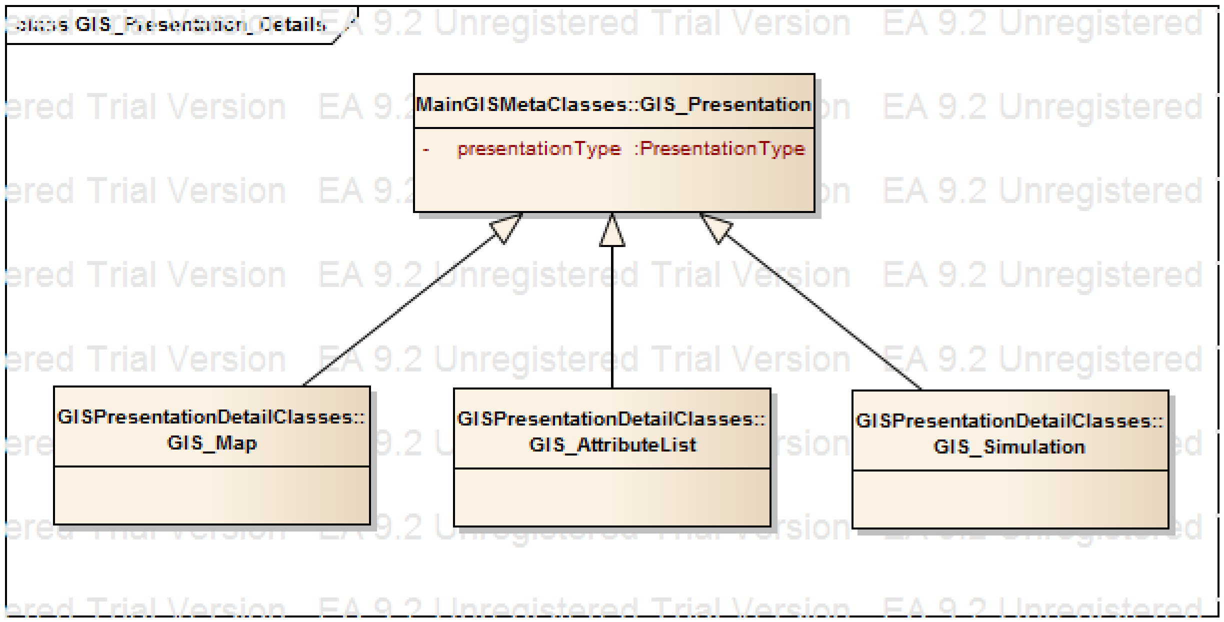

class GIS_Presentation_Details
MainGISMetaClasses::GIS_Presentation
- presentationType :PresentationType
GISPresentationDetailClasses::GIS_Map
GISPresentationDetailClasses::GIS_AttributeList
GISPresentationDetailClasses::GIS_Simulation
EA 9.2 Unregistered Trial Version

Figure 17: GIS meta model - GIS_Presentation details

With this meta model as (partially) shown in the diagrams before, all three GIS-specific aspects of chapter two can be modeled, and GIS domain specific topics can be modeled across various business domains.

6.3 GIS UML profile as domain specific language

6.3.1 Introduction

After consideration of the various research topics as explained in chapter 4 of this document, the author has decided to develop a graphical DSL based on UML standard[127] and its extension mechanism of UML profiles. The advantage of this decision is that the notation is widely known in the software industry and support is provided by several commercial and non-commercial UML modeling tools.

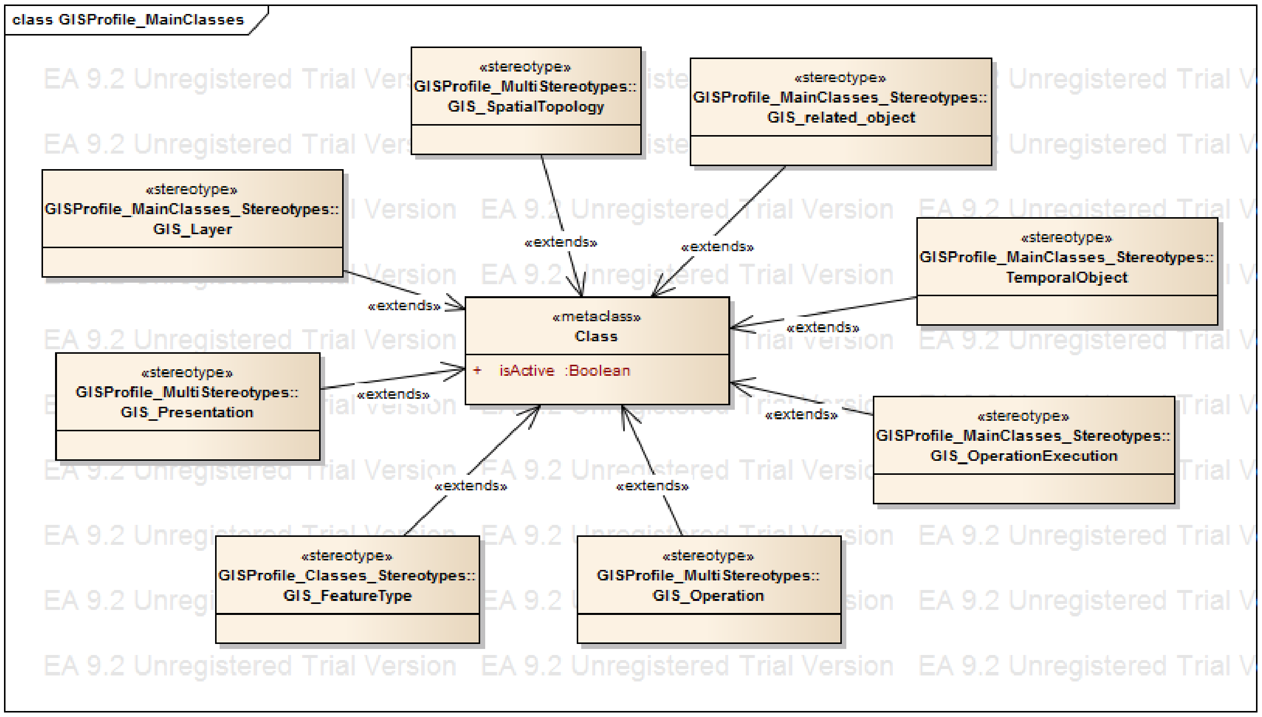
Based on the GIS meta model UML meta classes will be extended as listed below:

- GIS_FeatureType extends
    - UML package
    - UML class
- GIS_Operation extends
    - UML Activity
    - Action
    - State Transition
    - Flow Control
- Spatial Topology extends
    - UML Association
- GIS_Presentation extends
    - DiagramFrame

[127] Version 2.4.x has been used for modeling purpose

6.3.2 GIS UML Profile

The following diagram shows the stereotypes which extend the metaclass Class of UML. None of the stereotypes is declared as abstract since it could be used on different abstractions of a model such as in analysis or design.



class GISProfile_MainClasses
«stereotype» GISProfile_MultiStereotypes::GIS_SpatialTopology
«stereotype» GISProfile_MainClasses_Stereotypes::GIS_related_object
«stereotype» GISProfile_MainClasses_Stereotypes::GIS_Layer
«stereotype» GISProfile_MainClasses_Stereotypes::TemporalObject
«metaclass» Class
+ isActive :Boolean
«stereotype» GISProfile_MultiStereotypes::GIS_Presentation
«stereotype» GISProfile_MainClasses_Stereotypes::GIS_OperationExecution
«stereotype» GISProfile_Classes_Stereotypes::GIS_FeatureType
«stereotype» GISProfile_MultiStereotypes::GIS_Operation
«extends»

Figure 18: GIS Profile - main classes stereotypes

The metaclass Association of UML is extended by stereotypes defining the usage of data (left side of diagram in Figure 18) and the spatial topologies between objects (right side of diagram in Figure 18) with orientation on [Lisboa et. al., 2010].[128] A GIS_SpatialTopology has to fulfill a constraint that such an association can exist between two or more GIS_FeatureTypes only. Additional constraints can be added to specify more accurately which topologies are allowed and which might be forbidden. Th. Ubeda and M. Engenhofer describe such a concept of constraints in [Ubeda/Engenhofer, 1997]. Based on this concept constraints could be defined using OCL in a GIS UML profile and used for validations during modeling and more importantly during model transformations.

[128] With the detailed topological stereotypes the author follows [Lisboa et. al., 2010] based on [Clementini et. al., 1993] since the focus of the UML is practical usage rather than scientific formalism.

Since an association of two GIS_FeatureTypes can vary during time, an association can have a stereotype *temporal* applied.

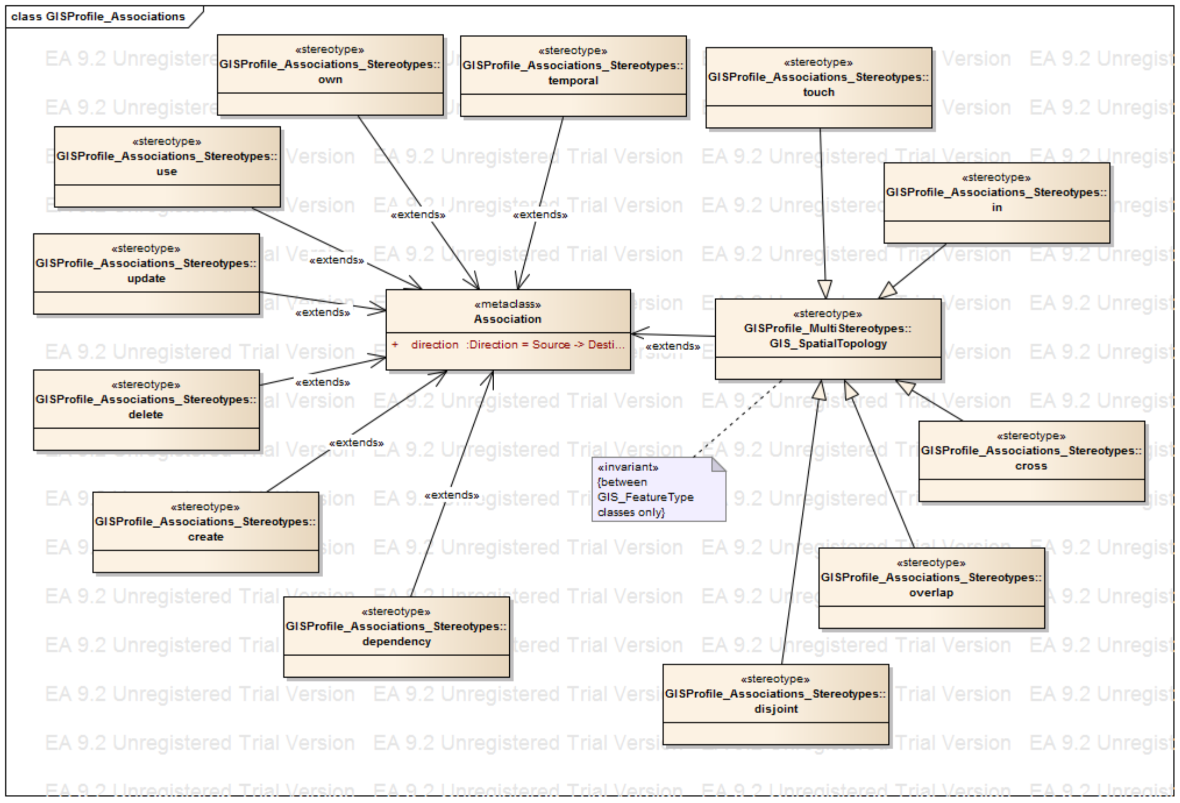

class GISProfile_Associations
«stereotype» GISProfile_Associations_Stereotypes::own
«stereotype» GISProfile_Associations_Stereotypes::temporal
«stereotype» GISProfile_Associations_Stereotypes::touch
«stereotype» GISProfile_Associations_Stereotypes::use
«stereotype» GISProfile_Associations_Stereotypes::in
«stereotype» GISProfile_Associations_Stereotypes::update
«metaclass» Association
+ direction :Direction = Source -> Desti...
«stereotype» GISProfile_MultiStereotypes::GIS_SpatialTopology
«stereotype» GISProfile_Associations_Stereotypes::delete
«stereotype» GISProfile_Associations_Stereotypes::cross
«invariant» {between GIS_FeatureType classes only}
«stereotype» GISProfile_Associations_Stereotypes::create
«stereotype» GISProfile_Associations_Stereotypes::overlap
«stereotype» GISProfile_Associations_Stereotypes::dependency
«stereotype» GISProfile_Associations_Stereotypes::disjoint
«extends»
EA 9.2 Unregistered Trial Version

Figure 19: GIS Profile - associations and spatial topology stereotypes

Besides the actual data structure and their relations the GIS operations are important to model Geographic Information Systems in a complete manner. The GIS operations extend different UML meta classes, mostly Activity and Action. The actual operations have been modeled as stereotypes, while the operation categories as defined by J. Albrecht as mentioned before are not represented by a stereotype.

A GIS_Operation stereotype and all of its sub-stereotypes have to fulfill a constraint that is to use GIS_FeatureType as input or output.

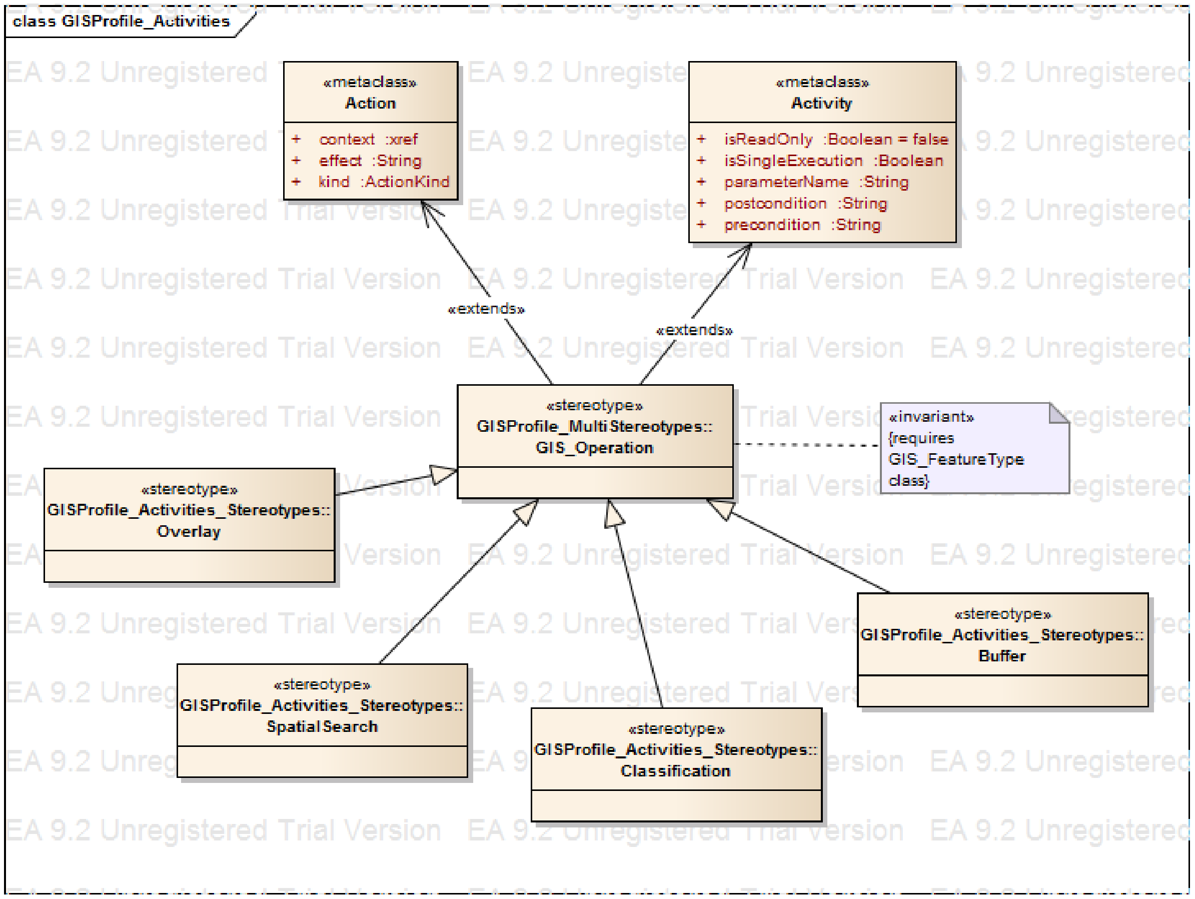


Figure 20: GIS Profile - GIS operation stereotypes

Although it might not be used too often during GIS domain modeling, any GIS domain specific presentation can be marked with specific stereotypes too as shown on the next diagram picture. Most important are map and list presentation types, but additional types such as for simulation could be added.

7. Modeling example

7.1 Introduction

The modeling example got its inspiration from the INSPIRE application schema specification of *ProtectedSites* in the consolidated UML model [Inspire, 2010]. So the sample model has basically two meta models, on one side the INSPIRE UML model, especially the application schema specification of *ProtectedSites*, and on the other side the GIS domain meta model as developed in the previous chapter.

So it becomes clear that modeling a particular domain might include multiple meta models and related domain specific languages. Considering A. Kleppes concept of vertical and horizontal domain specific languages the INSPIRE *ProtectedSites* application schema is the meta model for a vertical domain specific language[129] and the GIS domain meta model for the horizontal GIS domain specific language.

The structural aspects of the sample are modeled in a business object model (Class Diagram). Parks and Woodland are *ProtectedSites*. *Parks* have a *ProtectionZone* around them. Streets touch Parks or Woodland. Streets are planned by an Agency in a *DevelopmentPlan*, which affects the Community. Community and Agency have *Requirements* to the *DevelopmentPlan*.

[129] For the purpose of this sample the author has created an additional UML profile with the stereotype „ProtectedSite“ to represent the vertical DSL.

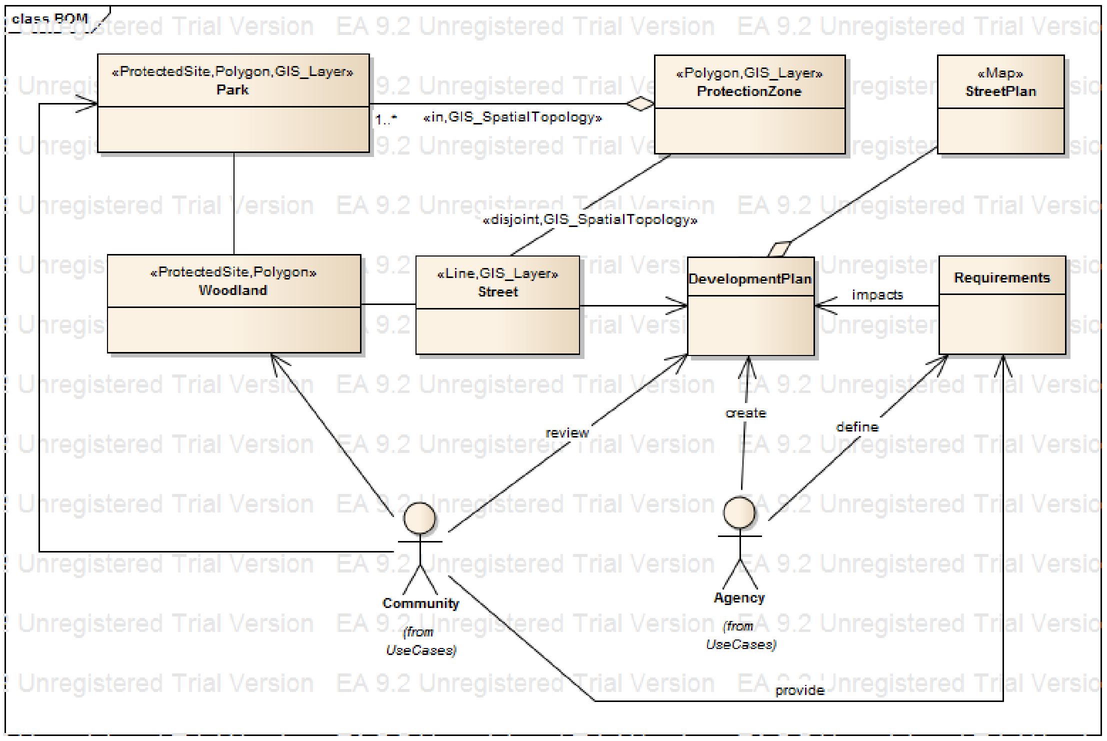


Figure 21: Sample Business Object Model

Any behavioral aspects are modeled on a conceptual level using a Use Case Diagram. But this diagram includes the business objects relevant for this use case too as an extension of the regular Use Case diagram content as specified by the OMG. The association stereotypes show the usage of these business objects by the use case.

In this sample three business objects are required by the Use Case *Determine ProtectionZone* to create a result object of type *ProtectionZone*.

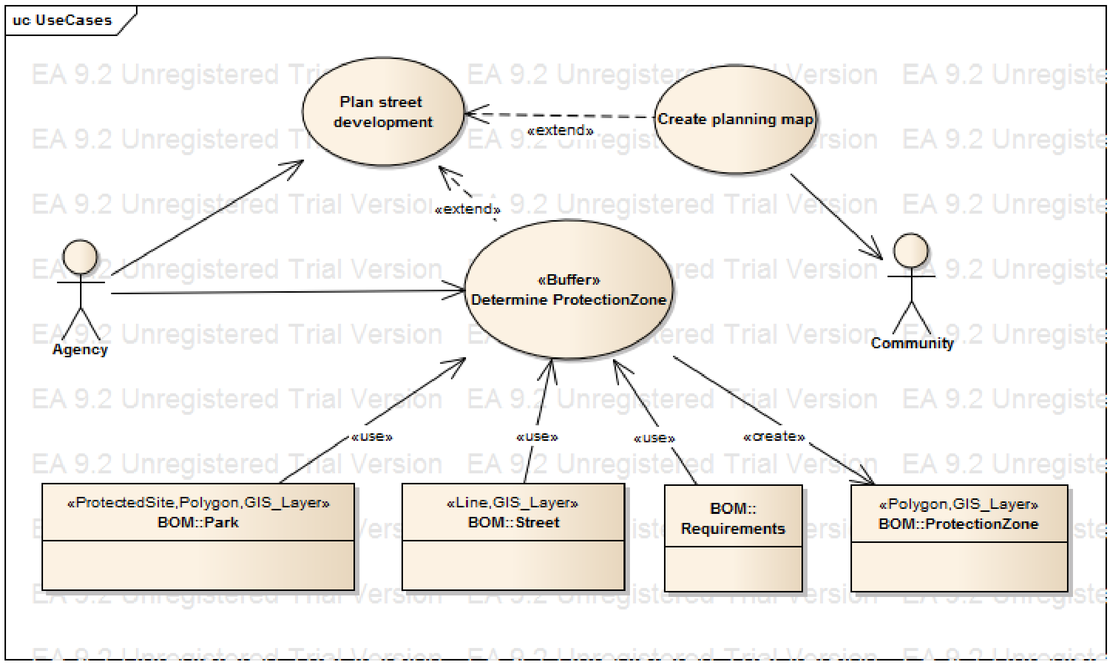

uc UseCases
Plan street development
«extend»
Create planning map
«extend»
Agency
«Buffer» Determine ProtectionZone
Community
«use»
«use»
«use»
«create»
«ProtectedSite,Polygon,GIS_Layer» BOM::Park
«Line,GIS_Layer» BOM::Street
BOM:: Requirements
«Polygon,GIS_Layer» BOM::ProtectionZone

Figure 22: Sample Use Cases

The actual processing is described in more detail in a related Activity Diagram of the "Buffer" Use Case. The usage of the business objects by a particular Action is specified there in more detail using stereotypes of the GIS UML profile for Associations.

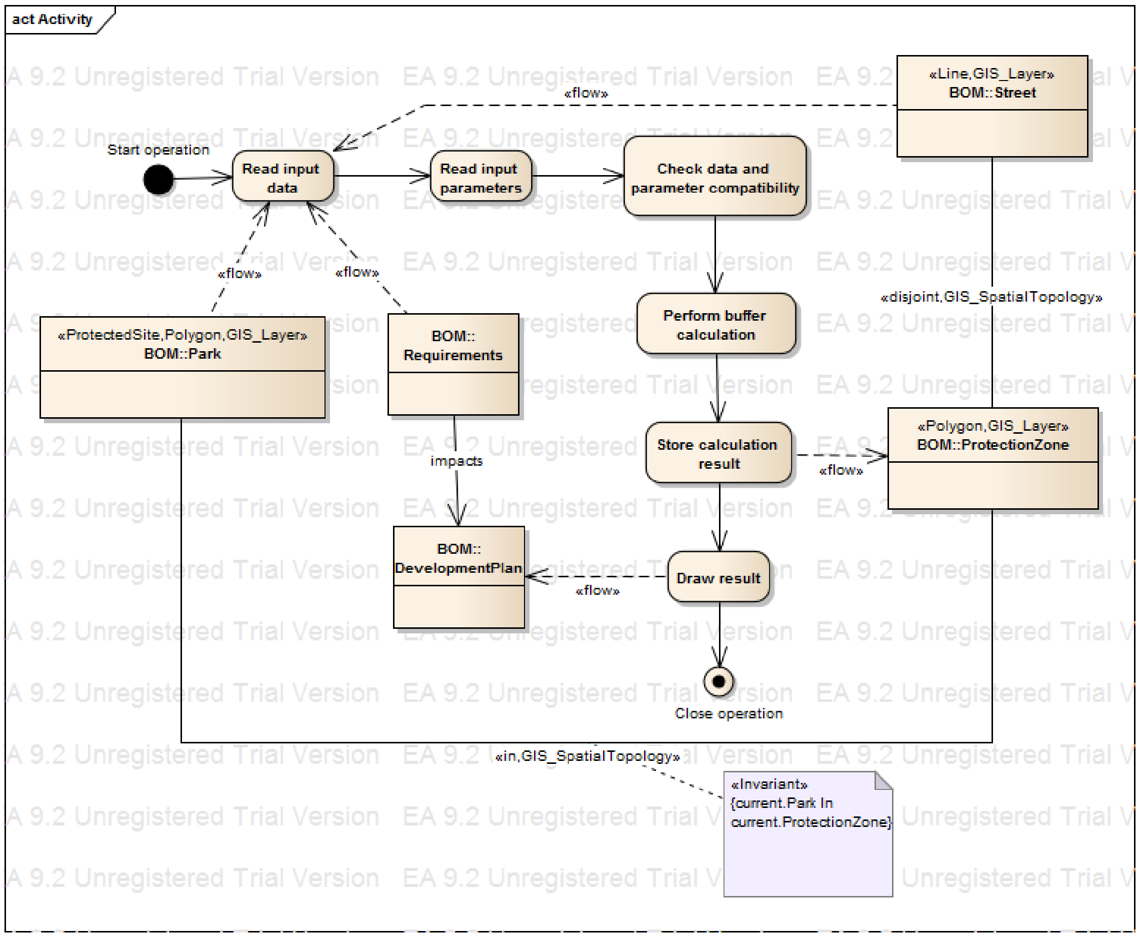

act Activity
«flow»
«Line,GIS_Layer» BOM::Street
Start operation
Read input data
Read input parameters
Check data and parameter compatibility
«flow»
«flow»
«disjoint,GIS_SpatialTopology»
«ProtectedSite,Polygon,GIS_Layer» BOM::Park
BOM:: Requirements
Perform buffer calculation
Store calculation result
«Polygon,GIS_Layer» BOM::ProtectionZone
«flow»
impacts
BOM:: DevelopmentPlan
Draw result
«flow»
Close operation
«in,GIS_SpatialTopology»
«Invariant» {current.Park In current.ProtectionZone}
EA 9.2 Unregistered Trial Version

Figure 23: Sample Activity as detailing of the Use Case "Determine ProtectionZone”

The topology of the data and constraints in their usage are shown in the Activity diagram by Associations with spatial topology information using stereotypes.

7.2 Expected transformation results

Although the description of the actual transformation of the sample model would exceed the boundaries of this Master Thesis document, potential results of such transformations should be shown here briefly.

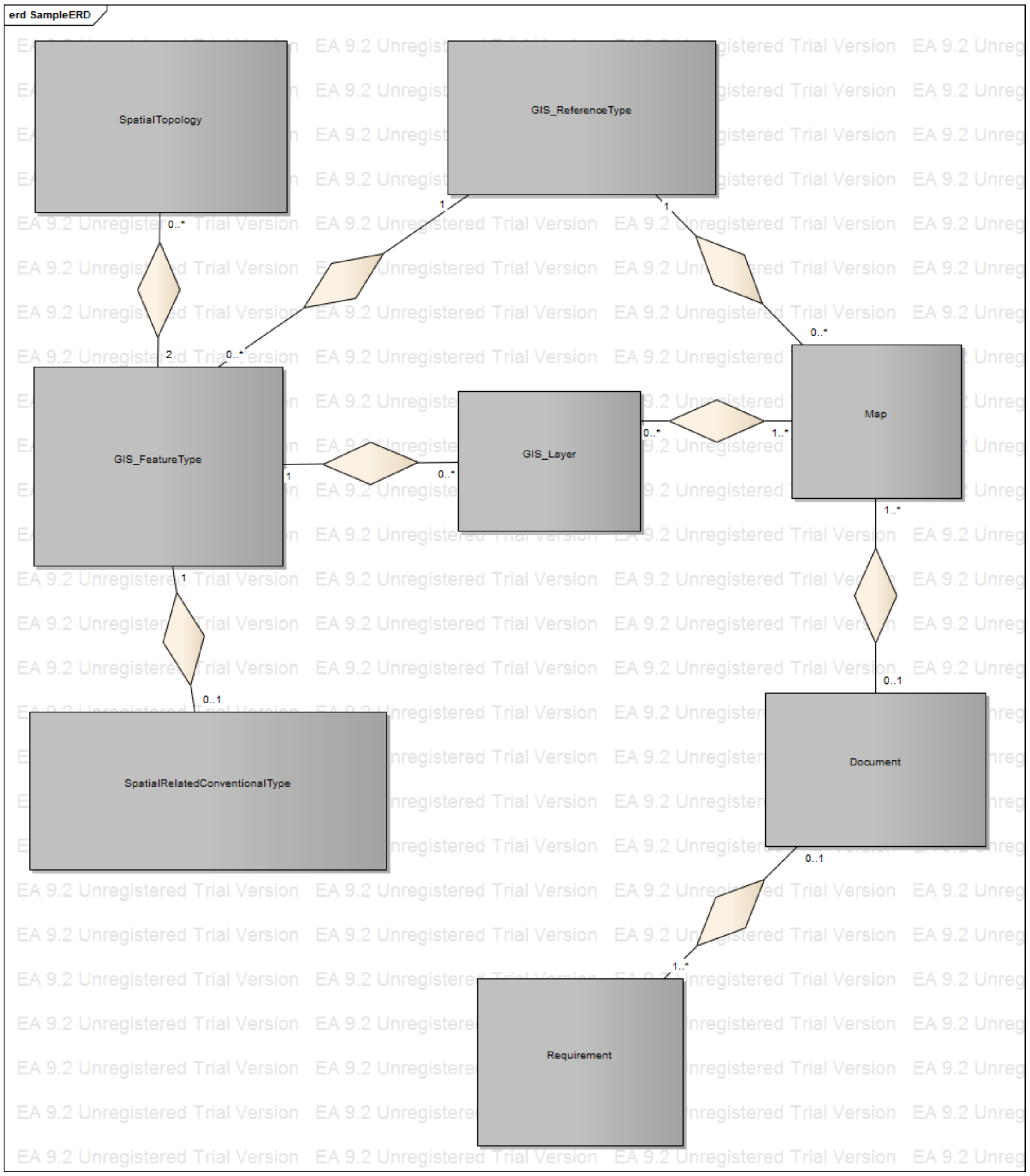


Figure 24: Sample Entity-Relationship Diagram without specific database platform information

The structural aspects of the sample in the Business Object Model could be first transformed to an Entity-Relationship model and then to a database specific data model with the corresponding DDL.[130]

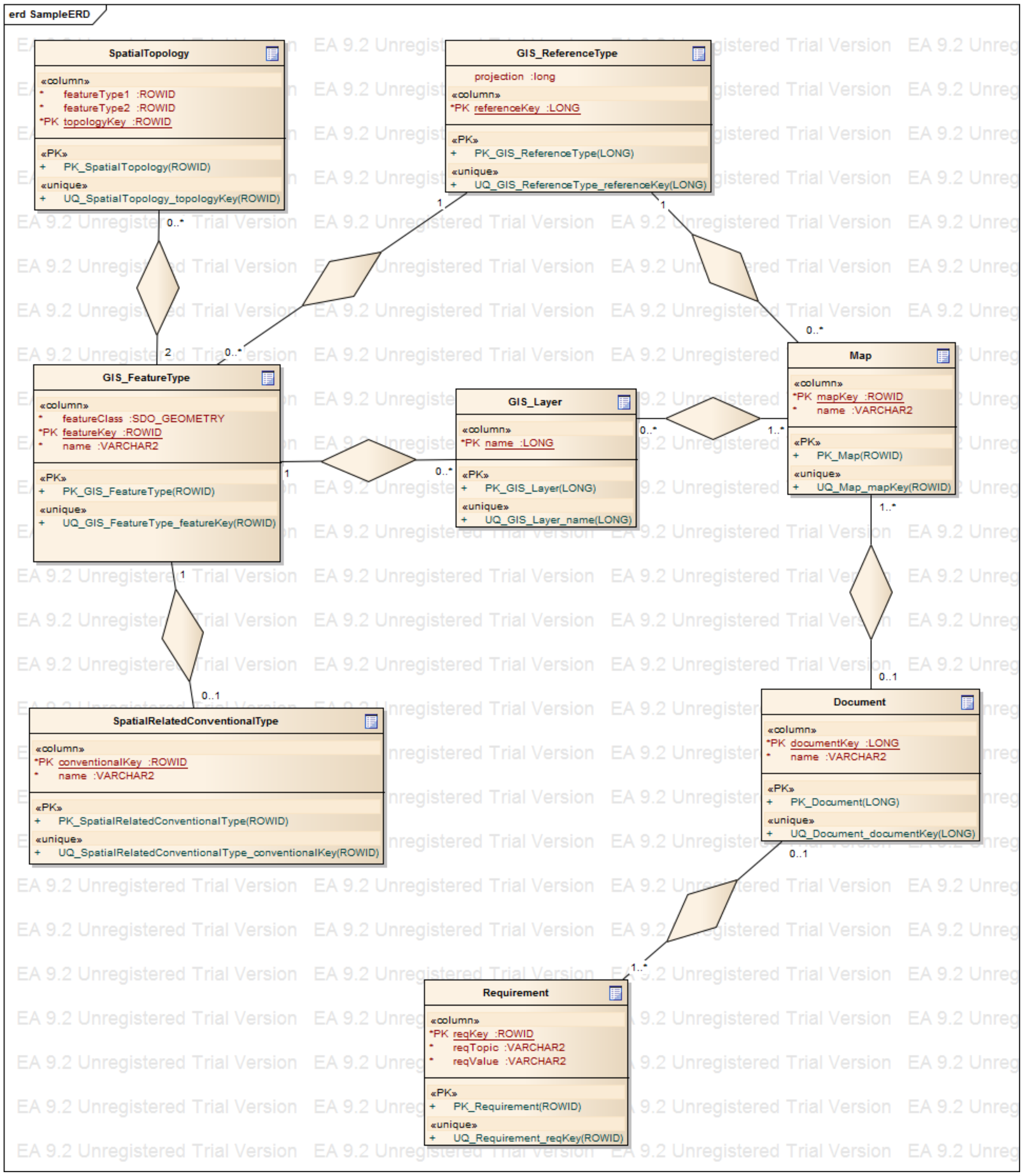
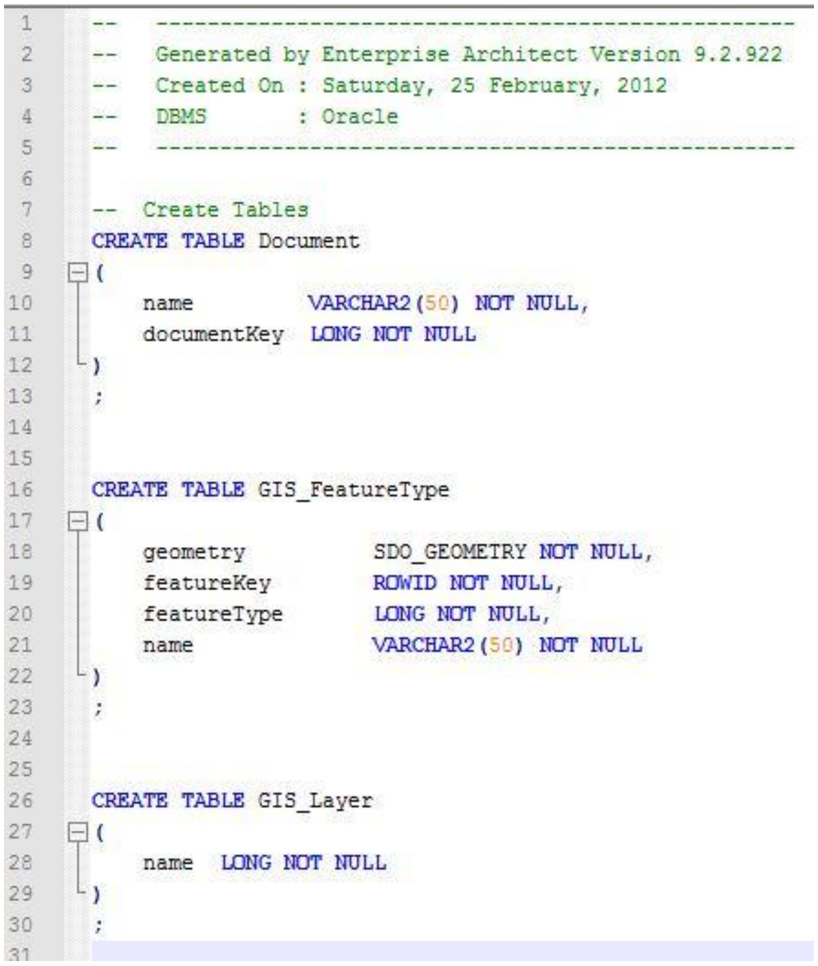


Figure 25: Entity Relationship diagram for Oracle database platform

[130] DDL = Data Definition Language

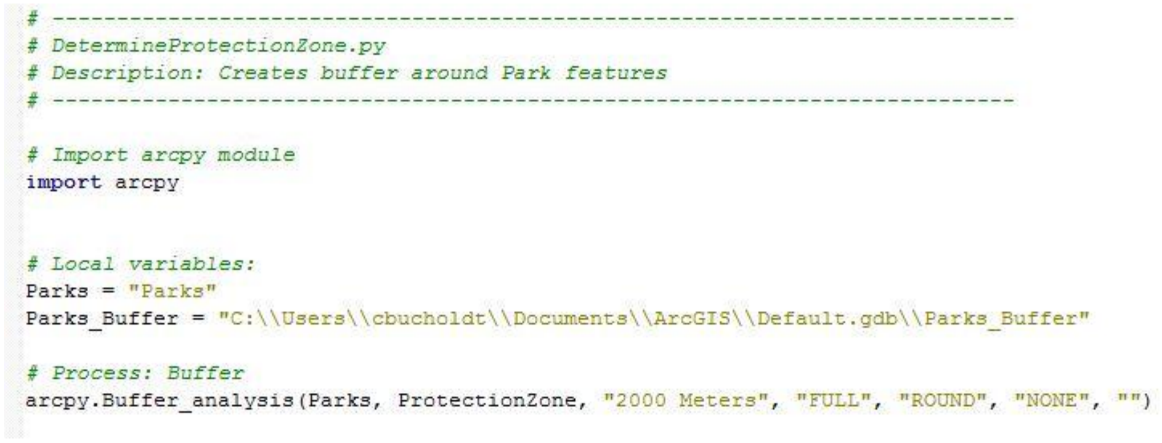
From the platform-specific model the actual SQL code for the database creation could be generated as shown in the code sample below.



```
--   ---------------------------------------------------
--   Generated by Enterprise Architect Version 9.2.922
--   Created On : Saturday, 25 February, 2012
--   DBMS       : Oracle
--   ---------------------------------------------------

--  Create Tables
CREATE TABLE Document
(
    name         VARCHAR2(50) NOT NULL,
    documentKey  LONG NOT NULL
)
;


CREATE TABLE GIS_FeatureType
(
    geometry          SDO_GEOMETRY NOT NULL,
    featureKey        ROWID NOT NULL,
    featureType       LONG NOT NULL,
    name              VARCHAR2(50) NOT NULL
)
;


CREATE TABLE GIS_Layer
(
    name  LONG NOT NULL
)
;

```

Figure 26: Sample DDL for Oracle as generated from the ER model

The behavioral aspects of the small sample around the buffer creation around a protected site would be managed in a different way. The actual spatial operation of the buffer creation around a Park object could result in a Python script as seen below. Although the Python script is already more platform specific than the GIS_Operation modeling in the UML model it is still platform independent in the sense of a particular platform to execute the script.



```
# ---------------------------------------------------------------------------
# DetermineProtectionZone.py
# Description: Creates buffer around Park features
# ---------------------------------------------------------------------------

# Import arcpy module
import arcpy


# Local variables:
Parks = "Parks"
Parks_Buffer = "C:\\Users\\cbucholdt\\Documents\\ArcGIS\\Default.gdb\\Parks_Buffer"

# Process: Buffer
arcpy.Buffer_analysis(Parks, ProtectionZone, "2000 Meters", "FULL", "ROUND", "NONE", "")
```

Figure 27: Python script of buffer creation in GIS_Operation "Determine ProtectionZone"

The script could then be imported in the ArcGIS modeler and executed there on the specific platform of a technological product similar to the Oracle database platform for the structural aspects mentioned before.[131] The result of the *ProtectionZone* is represented as a *GIS_Layer* and will be part of a map, which displays the results to a target audience as part of a map.

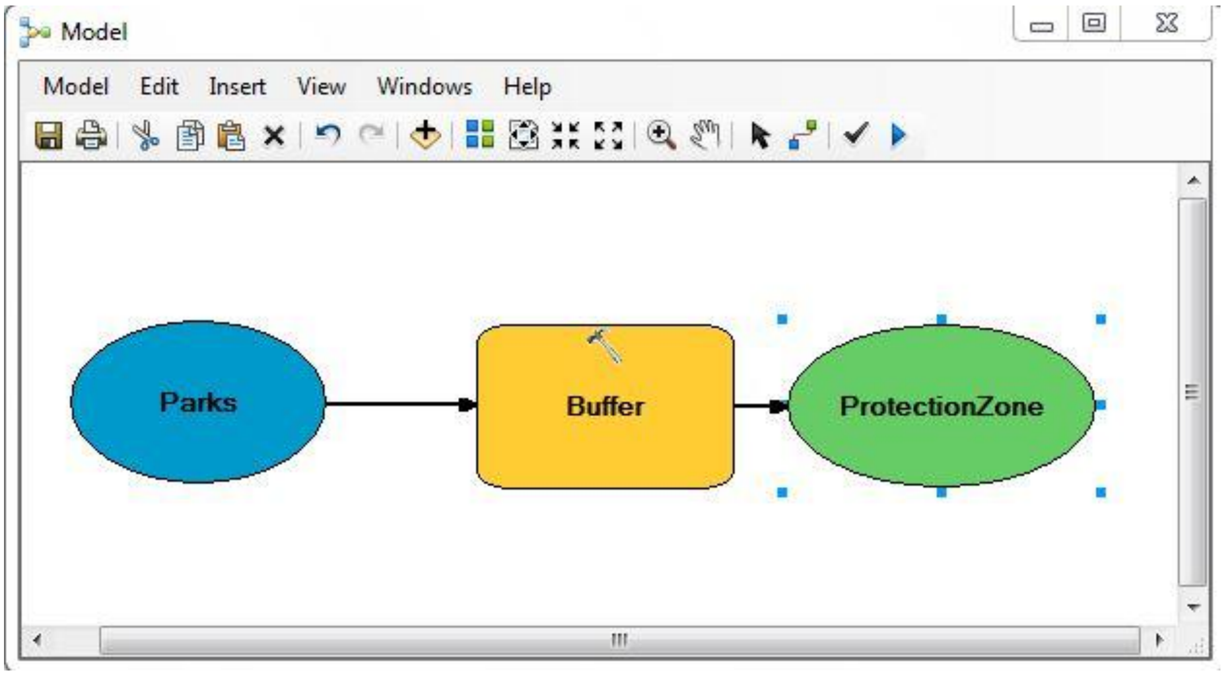


Figure 28: ArcGIS representation of Python script

[131] For details see [Allen, 2010]

8. Conclusions and future work

Based on the presentation of the data that needs to be managed, the operations to be performed and the kind of result presentation this Master Thesis has positioned GIS as a very specific IT domain in comparison to business applications as most people know from their daily experience. This makes GIS a good candidate for the usage of domain specific techniques for application development.

The review of the research in this area and the state of the art in regards to domain specific languages and model-driven development has shown that many attempts have been made to introduce GIS specific modeling techniques, e.g. [Kösters, 1996] or [Albrecht, 1997]. None of these could be established for broader use in GIS application development. Just in recent years when usage of DSLs and UML has been introduced in the broad IT community for application development the base is given to establish GIS domain specific techniques as solutions on top of international standards. The in depth review of DSL developments has shown that the success of establishing a DSL goes very much along with approaches for model-driven development in order to introduce efficiency gains in the software development process. The work of B. Selic [Selic, 2009] and J. Lisboa [Lisboa et. al., 2010] shows that GIS DSL development based on a general IT standard such as UML is taking another attempt.

In order to introduce domain specific languages or techniques for GIS application development the GIS domain has been specified in more detail. Based on A. Kleppes [Kleppe, 2009] concept of vertical and horizontal domain, the GIS domain has been characterized as a horizontal service which is of or in use in many businesses. This provides the base for the application of a horizontal DSL in GIS application development or development of GIS aspects in any (non GIS specific) IT system.

The graphical nature of GIS in general based on the map orientation of the representation of its result, the usefulness of the meta model approach for verification purpose of models and DSLs and the availability of a well-established modeling standard with a graphical notation such as UML and the available tool support have led the author to the conclusion that a DSL for the GIS domain needs to be based on UML and its built-in extension mechanism of Profiles.

Comparing modeling approaches in the GIS area such as [Lisboa et. al. 2010] and others to the actual specifics of GIS as explained in the beginning of this Master Thesis it is clear that the area of data modeling needs to be extended by modeling of GIS operations and most likely of GIS representation too. Based on this finding, a meta model for modeling the GIS domain specific aspects has been developed. A UML profile has been established as concrete DSL syntax for usage in standard UML tools. In order to verify the completeness of a first version of such a profile an example model has been established which applies the GIS Profile to data and operations aspects. The example contains potential results of model transformations based on the applied GIS UML profile.

This approach follows the separation of platform independent domain model and platform specific technology as proposed by the MDA standard. It enables potential transformations to other platform standards (e.g. ISO 191xx) and various technical platform outputs such as database definition language scripts or programs for execution in a GIS tool such as the ArcGIS model builder. This approach could be expressed in terms of DSL development as separation between vertical and horizontal modeling.

[Nalon et. al., 2011] is using the GeoProfile as introduced by [Lisboa et. al., 2010] to show the application of a model driven architecture example by transforming a conceptual model to a platform specific data model according to the ISO standard for spatial data modeling. Since the transformation basically contains just a 1:1 mapping of conceptual data classes to ISO conformant data classes the approach has to be considered rudimentary.

The example provided by the author of this Master Thesis shows a simple example including GIS operations along with the data modeling aspects. One experience is that by following A. Kleppe's approach of separating between vertical and horizontal DSLs a particular modeling example is affected by multiple DSLs and potentially multiple meta models. This makes validation of a model and automated transformation to a particular output difficult since the different meta models might cause conflicts.

From the experience of both examples and the feedback from the research community it has become clear that transformation of models and generation from models is still the biggest pain point in the practical application of model-driven architecture and development. This is not just related to GIS, but a general issue in the IT domain. The most important pain points in this regards are the missing standardization and tool support for model transformations. The OMG "standard" QVT is basically a conglomerate of three different standards. The model to model and model to text transformation support by the Eclipse Modeling Project is based on the EMF meta model and not the core UML meta model [Gronback, 2009]. The tools available require significant setup effort and are not considered "easy to use" as shown in [Bagheri/Sullivan, 2010] or [Wienands/Golm, 2009].

Although MDA is further progressed when it comes to data structures, a broad field for further research is left of how to apply model-driven development to behavioral and dynamic aspects. Due to the specific nature of GIS operations it might be worth to invest effort in this regards in a small domain area rather than for general purpose modeling.

How the application of multiple meta models and multiple DSLs together in a particular occasion of application development should be handled has to be investigated. Interesting is the idea of a temporary merge of different DSLs for an application development project or how this could be managed over a complete application lifecycle of a GIS application. Especially, when considering the reuse of a horizontal DSL in different contexts this is an important topic.

9. Literature

[Albrecht, 1995] Albrecht, J. (1995) Virtual Geographic Information System (VGIS); Proceedings of the 28th Annual Hawaii International Conference on System Sciences – 1995, p. 141 – 150

[Albrecht, 1996] Albrecht J. (1996) Universal GIS Operations; at: http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/sf_papers/jochen_albrecht/jochen.santafe.html (14.10.2011)

[Albrecht et. al., 1997] Albrecht J., Jung S., and Mann S. (1997) VGIS: a GIS shell for the conceptual design of environmental models. in: Selected papers from the Fourth National Conference on GIS Research in the UK (GISRUK), Innovations in GIS 4, Z. Kemp (Editor), Taylor & Francis Inc., Bristol, PA, pp. 154-165.

[Allen, 2010] Allen, D. W. (2010) Getting to know ArcGIS ModelBuilder, ESRI Press, Redlands

[Bagheri/Sullivan, 2010] Bagheri, H., Sullivan, K. M. (2010) Model-Based Development of Software Architectures in: Proceedings, Part II of 13th International Conference, MODELS 2010, Springer 2010, pp. 376 – 390

[Bédard/Larrivée, 2008] Bedard Y., Larrivée S. (2008) Modeling with Pictogrammic Languages, In: Shekhar, S.; Xiong, H. (Eds.). Encyclopedia of GIS. Springer 2008, pp.716 - 724

[Borges et. al., 2001] Borges K.A.V., Davis Jr. C.A., Laender, A.H.F (2001) OMT-G: An object-oriented data model for geographic applications. GeoInformatica, 5(3), pp. 221-260

[Brewer, 2005] Brewer, C. A. (2005) Designing better maps: A guide for GIS users, ESRI Press, Redlands

[Chen, 1976] Chen, P.S. (1976) The Entity-RelationshipModel: Toward a Unified View of Data, ACM TODS,1, pp. 9-36

[Clementini et. al., 1993] Clementini, E., Di Felice, P., Oosterom, P. (1993) A small set of formal topological relationships suitable for end-user interaction. In: International Symposium on Advances in Spatial Databases. Springer 1993, pp. 277-295

[Clements, 2010] Clements P. et. al. (2010) Documenting Software Architectures: Views and Beyond. 2nd Ed.. Addison-Wesley, Upper Saddle River NJ

[Coad/Yourdon, 1991] Peter Coad , Edward Yourdon (1991): Object Oriented Analysis (2nd Edition) (Yourdon Press Computing Series), Engelwood Cliffs, NJ

[Cook, 2004] Cook S. (2004) Domain Specific Modeling and Model Driven Architecture, at: http://www.bptrends.com/publicationfiles/01-04%20COL%20Dom%20Spec%20Modeling%20Frankel-Cook.pdf (18.12.2011)

[Crow, 2000] Crow S. (2000) Spatial Modeling Environemnts, in: GIS/EMA 4, 2000; at: http://www.colorado.edu/research/cires/banff/pubpapers/2/ (23.10.2011)

[Czarnecki/Eisenecker, 2000] Czarnecki Krzysztof, Eisenecker Ulrich W. (2000) Generative Programming: methods, tools and applications, Addison-Wesley, Upper Saddle River NJ

[Ehrig et. al., 2009] Ehrig H., Ermel C.; Herman F.; Prange U. (2009) On-the-Fly Construction, Correctness and Completeness of Model Transformations Based on Triple Graph Grammars, in: Proceedings of 12th International Conference, MODELS 2009, Springer 2009, pp. 241 - 255

[Engenhofer, 1993] Engenhofer M. (1993) A Model for Detailed Binary Topological Relationships, in: Geomatica Vol. 47, No. 3 & 4, Autum 1993, pp. 261-273

[Evans, 2004] Evans E. (2004) Domain-Driven Design, Addison-Wesley, Upper Saddle River NJ

[Fowler, 2002] Fowler M. (2002) Patterns of Enterprise Application Architecture, Addison-Wesley, Upper Saddle River NJ

[Fowler, 2005] Fowler M. (2005) Language Workbenches. The Killer-App for Domain-Specific Languages?, at: http://martinfowler.com/articles/languageWorkbench.html (14.01.2012)

[Fowler, 2010] Fowler M. (2010) Domain-Specific Languages, Addison-Wesley, Upper Saddle River NJ

[Frankel, 2003] Frankel, D. S. (2003) Model Driven Architecture: Applying MDA to Enterprise Computing, OMG Press

[Gronback, 2009] Gronback, R. C. (2009) Eclipse Modeling Project: A domain specific language toolkit. Addison-Wesley, Upper Saddle River NJ

[Guerra et. al., 2010] Guerra E.; de Lara J. et. al. (2010) TransML: A Family of Languages to Model Model Transformations in: Proceedings, Part I of 13th International Conference, MODELS 2010, Springer 2010, pp. 106 – 120

[Guttmann, 2007] Guttman, M. (2007), Real-Life MDA, Morgan Kaufmann

[Inspire, 2010] INSPIRE Consolidated UML Model, 26.04.2010, at: http://inspire-twg.jrc.ec.europa.eu/inspire-model/ (04.01.2012)

[Inspire/GMES, 2011] Information Broschure, Inspire/GMES, 2011, 7th Ed., Editor: Matthäus Schilcher

[Kleppe, 2009] Kleppe A. (2009) Software Language Engineering. Creating Domain-Specific Languages Using Metamodels, Addison-Wesley, Upper Saddle River NJ

[Kösters, 1996] Kösters G., Pagel B.-U., Six H.-W. (1996) GeoOOA: Object-oriented Analysis for Geographic Information Systems, at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.8480&rep=rep1&type=pdf

[Kösters, 1995] Kösters G., Pagel B.-U. (1995) The GeoOOA-Tool and its Interface to Open GIS-Software Development environments, at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.6220&rep=rep1&type=pdf

[Kothuri, 2009] Kothuri R., Godfrind A., Beinat E. (2009) Oracle Spatial for Oracle Databases 11g, Apress, Berkley, CA

[Lisboa et. al., 2010] Lisboa Filho J. et. al. (2010), A UML Profile for Conceptual Modeling in GIS Domain, in: International Workshop on Domain Engineering (DE@CAiSE'2010), p.18-31.

[Lisboa/Iochpe, 1999] Lisboa Filho J., Iochpe C. (1999) Specifying Analysis Patterns for Geographic Databases on the Basis of a Conceptual Framework, in: Proceedings of the 7th ACM GIS, pp. 7-13.

[Longley et. al., 2011] Longley P., Goodchild M., Maguire D., Rhind D. (2011), Geographic Information Systems & Science, 3rd Ed., John Wiley & Sons, Inc., Hoboken, NJ

[Lucio et. al., 2010] Lucio L., Barroca B., Amaral V. (2010) A Technique for Automatic Validation of Model Transformations in: Proceedings, Part I of 13th International Conference, MODELS 2010, Springer 2010, pp. 136 – 150

[Mann, 1996] Mann S. (1996) Spatial process modeling for regional environmental decision making, in: Proceeding of 8th Annual Colloquium on Geographical Information Systems and Spatial Information Research, Dunedin, NZ, University of Otago

[Maxted, 2008] Maxted S. J. (2008) The importance of business domain modeling, October 2008, in: http://www.batimes.com/articles/the-importance-of-business-domain-modelling.html (14.01.2012)

[Miralles et. al., 2008] Miralles A., Libourel T. (2008) Modeling with Enriched Model Driven Architecture, In: Shekhar, S.; Xiong, H. (Eds.). Encyclopedia of GIS. Springer, pp.700 – 705

[Murray et. al., 2000] Murray, S. et. al. (2000) A Visual Framework for Spatial Modeling; in: GIS/EMA 4; at: http://www.colorado.edu/research/cires/banff/pubpapers/14 (23.10.2011)

[Nalon et. al., 2011] Nalon F. R. et. al. (2011) Using MDA and a UML Profile integrated with International standards to model geographic databases, in: Journal of Information and Data Management, Vol. 2, No. 2, June 2011, Pages 171–180

[OGC, 1999] Open GIS Consortium (1999) OpenGIS Simple Feature specification for SQL (Open GIS project document 99-049), Rev. 1.1, Open GIS Consortium Inc.

[OGC, 2011] Open GIS Consortium (2011) OpenGIS Implementation Standard for Geographic Information – Simple feature access – Part 1: common Architecture (Open GIS project document 06-103r4), Rev. 1.2.1, Open GIS Consortium Inc.

[Oldfield, 2002] Oldfield P. (2002) Domain Modeling, Appropriate Process Group, at: http://www.aptprocess.com/whitepapers/DomainModelling.pdf (15.01.2012)

[Orejas et. al., 2009] Orejas, F. et. al. (2009) Correctness, Completeness and Termination of Pattern-Based Model-to-Model Transformation: Long Version, Technische Universität Berlin, Forschungsberichte der Fakultät IV – Elektrotechnik und Informatik, 2009/9

[Parent et. al., 2008] Parent C., Spaccapietra S., Zimányi E. (2008) Modeling and multiple perceptions, in: Shekhar, S.; Xiong, H. (Eds.). Encyclopedia of GIS. Springer, pp.682-690

[Parmenter, 2007] Parmenter, B. (2007) Geographic Data Structures; Tufts University 2007, at: http://ocw.tufts.edu/data/54/676127.pdf (12.01.2012)

[Parr, 2010] Parr T. (2010) Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages, Pragmatic Bookshelf

[Rumbaugh, 1991] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., (1991) Object-Oriented Modeling and Design. Prentice-Hall, Englewood Cliffs, NJ

[Selic, 2007] Selic B. (2007) A Systematic Approach to Domain-Specific Language Design using UML. In: Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time distributed Computing, 2007, pp. 2-9

[Selic, 2009] Selic B. (2009) The Theory and Practice of Modeling Language Design for Model-Based Software Engineering - A Personal Perspective, in: Generative and Transformational Techniques in Software Engineering III, Lecture Notes in Computer Science, 2011, Volume 6491/2011, pp. 290-321

[Selic, 2011] Selic B. (2011) The Theory and Practice of Modeling Language Design, Tutorial T3, at: 14th International Conference, MODELS 2011, Wellington, New Zealand, October 2011, at: http://ecs.victoria.ac.nz/twiki/pub/Events/MODELS2011/Material/MODELS_2011_T3-Selic.ModelingLanguages.Tutorial.updated.v2.pdf (03.01.2012)

[Sheklar/Chawla, 2003] Shekhar S., Chawala S. (2003). Spatial Databases: A Tour, Pearson

[Shekar/Xiong, 2008] Shekhar S., Xiong H. (Eds.) (2008) Encyclopedia of GIS, Springer

[Stahl/Völter, 2006] Stahl Th., Völter M. (2006) Model-Driven Software Development. Technology, Engineering, Management, John Wiley & Sons, Inc., Hoboken, NJ

[Stempliuc, 2009] Stempliuc S. M., Lisboa Filho J., Andrade M. V. A., Borges K. V. A. (2009) Extending the UML-GeoFrame data model for conceptual modeling of network applications. In: Int. Conf. on Enterprise Information Systems (ICEIS), Milão pp. 164--170, at:

http://www.dpi.ufv.br/~jugurta/papers/Stempliuc%20-%20ICEIS%20-%20Publication.pdf (08.02.2012)

[Strobel, 2005] Strobel, Josef (2005) GI Science and technology – where next?, in: GIS@development, August 2005, Vol. 9 Issue 8

[Tomlin, 1990] Tomlin, C. D. (1990) Geographic Information Systems and Cartographic Modeling, Prentice Hall, Engelwood Cliffs, NJ

[Trask/Roman, 2011] Trask B., Roman A. (2011) Applying Model Driven Engineering Technologies in the Creation of Domain-Specific Modeling Languages, Tutorial T4, at: 14th International Conference, MODELS 2011, Wellington, New Zealand, October 2011, at: http://ecs.victoria.ac.nz/twiki/pub/Events/MODELS2011/Material/MODELS_2011_T4-Trask-Roman.pdf (14.01.2012)

[Tryfona, 1996] Tryfona N.; Pfoser, D. ; Hadzilacos T. (1996) Modeling Behavior of Geographic Objects: An Experience with the Object Modeling Technique; Technical report 96-11, at: http://lab.geog.ntu.edu.tw/course/ginformation/Homework/Prima/modeling%20behavior%20of%20geographic%20objects.pdf (07.11.2011)

[Ubeda/Engenhofer, 1997] Ubeda Th., Engenhofer M. (1997) Topological Error Corrections in GIS, in: Scholl M., Voisard (Eds.) Advances in Spatial Databases, Proceedings of the 5th International Symposium, SSD 1997, Berlin, pp. 283 - 297

[UML241I] Object Modeling Group (2011) UML 2.4.1 Specification: Infrastructure, formal/2011-08-05, http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF (02.02.2012)

[UML241S] Object Modeling Group (2011) UML 2.4.1 Specification: Superstructure, formal/2011-08-06, http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF (02.02.2012)

[Wagelaar et. al., 2011] Wagelaar D., Tisi M. et. al. (2011) Towards a General Composition Semantics for Rule-Based Model Transformations in: Proceedings of 14th International Conference, MODELS 2011, Springer, pp. 623 – 637

[Wienands/Golm, 2009] Wienands Ch., Golm M. (2009) Anatomy of a Visual Domain-Specific Language. Project in an Industrial Context. In: 12th International Conference, MODELS 2009, Denver, CO, USA, October 2009, Proceeding, LNCS 5795, pp. 453-467

[Yeung/Hall, 2007] Yeung A.; Hall G. B. (2007) Spatial Database Systems: Design, Implementation and Project Management, Springer, New York, NY

[Zeiler, 2010] Zeiler M. (2010) Modeling Our World: The ESRI Guide to Geodatabase Concepts, ESRI Press, 2nd Ed, Redlands