Spatio-Temporal Data Modelling for "4D"-Databases

Alexander Zipf

Prof. Dr. Alexander Zipf
Professor for Applied Computer Science
Department for Geoinformatics and Surveying
FH Mainz, University of Applied Sciences
Mainz Germany
Tel.:+49 - 6131 - 2859 - 625
Fax: +49 - 6131 - 2859 - 615
Email: zipf@geoinform.fh-mainz.de
Web: http://www.geoinform.fh-mainz.de/~zipf/
1 Introduction

Conventional GIS are usually quite static, as they do not cover dynamic aspects of geo-objects in their data model. The information on the modeled domain is usually separated into model of geometric space (2D/3D) and thematic aspects (attributes). But if someone wants to develop a system that is capable of modeling objects of the environment including their history, presence and future most available systems lack expressive power. It has been demanded that a temporal GIS (TGIS) needs to provide functionality for spatio-temporal data storage, data handling, analysis as well as visualization. These functions are usually more complex as in conventional GIS and still an area of active research.

Within the Deep Map/GIS project a flexible and extensive temporal object-oriented model had been developed [for Deep Map see: 23, 20, 7]. The aim was to allow the management of 3D geo-objects of urban areas over historic epochs and act as basis for the data management components of temporal 3D-GIS (“3D-TGIS” or more colloquial "4D-GIS") to be developed in the future. Since the temporal part of this model is a self-consistent OO-model for temporal structures it can also be used with 2D-geodata.

The proposed framework is a contribution towards the development of a temporal 3D GIS by offering guidelines how to model the time in a sophisticated way. It also shows how to integrate these temporal aspects of geo-objects along with their 3D spatial (topological) and thematic aspects. Working prototypes have been realized that implement these models in an object-oriented and an object-relational DBMS, showing the applicability of the proposed concepts. The model has been demonstrated within the domain of 3D historic city models for a city information system.
2 Spatio-Temporal Data Modeling

A geo-object or feature in general consists of the aspects theme, geometry, topology and time [1][14]. Still today’s GIS don’t handle all aspects equally well. The temporal dimension is an important aspect of most real world phenomena. But databases or GIS delivered only a snapshot of the real world. Therefore there was a need for new data models that allows the handling of temporal data [13],[15]. Within the last years a range of temporal models were also developed in the field of object-oriented databases. [9][10] and [11] give a summary. To name only a view [3],[6],[16] and [18] present possibilities for an object-oriented integration of temporal models into 2D GIS.

To represent the basic elements of the temporal framework some important concepts are defined briefly: The period of the physical process used to measure time is called "chronon" while the duration of the period is described as a "granularity. A temporal framework should provide means to representing arbitrary calendars. Further aspects of time as explained in more detail in [5].

3 Topological modelling of three-dimensional geo-objects

The development o the data model for 3D geometry is largely influenced from the model of Molenaar]. It combines the geometry and topology of 3D geodata and allows retrieving multiple topological properties directly from the model.

The basic concepts include the primitives Node (point), Arc (Line) and Face (Area). Thematic attribute data is attached using feature identifiers. Molenaar extended earlier models by the new primitives edge and body to model the third dimension [25].
The topology of the 3D primitives has been modelled through several 1:n relationships between the 5 primitives:

- For every Arc there exists exactly one start-and endpoint (node)
- A node can belong to several arcs.
- A face can only margins two bodys, while one body can have several faces.
- There are links between Arcs and Nodes to the face they belong to or the body they are part of.
- Face and body both consist of several Nodes or Arcs.

*fig. 1: topological relationships between the 3D primitives [after [25]].*
A Unified Modeling Language (UML) class diagram that models the geometry model of the framework is depicted in figure 2.

The data model introduced so far describes the topology of up to three-dimensional objects. The actual geometrical data is integrated by relating multiple versions of geometry to the primitives. In the case of nodes these are the actual coordinates, for an arc these are the coordinates of the vertices (points in between nodes, representing geometry).

The body primitive does not need further geometric information as it is described by the constituting faces. The classes for the geometry were realized similarly according to the 3D model using the primitives Point, Face and Body. They shall be called 0Cell (cell0), 1-Cell (cell1), 3-Cell (cell2) and 3-Cell (cell3) according to their dimensionality, see figure 3.
Within the spatio-temporal model only the primitives 2-Cell or 3-Cell have been used. The realisation of the relationships between the spatial and temporal parts of the model have been achieved using coupling classes. This class is called `combCell`. Both primitives 2-Cell and 3-Cell inherit properties from that. Modelling these relationships using coupling classes offers the following benefits: First, redundancy is minimized and secondly the geometrical components can be coupled in a more flexible way with temporal aspects, as the individual parts of the model can be exchanged or altered freely. If another class also inherits from `combCell` it can replace the spatial model we used with a different one quite easily.

4 Modelling of thematic data – the example of the history of a city

The structures describing the thematic aspects of the features (geo-objects) are also being realised using an object-oriented model. The thematic model cannot be generic but is oriented towards the application domain. In the case of the Deep Map project this was a city information system, where individual buildings with their visible parts (from outside) and other man-made structures within a city are being modelled. Other geographic domains can also be applied by extending or exchanging the thematic model.

The most important three-dimensional real world objects are in our case `buildings`, `monuments`, `bridges`, `fountains`, `gates` and `roads`. Parts of such 3D objects may belong to the classes `body` (of a building), `stair`, `tower`, `roof`, `wall` or `yard`. But as it is likely that more complex 3D-objects need
to be represented, it seems sensible to be able to aggregate such objects to a more complicated semantic unit. This is realized through the relationships between the class `threeD_Obj` and `part_threeD_Obj`. This allows within the thematic model to assemble several parts of a 3D-object together. An example is for example to define an object “southern wing” (e.g. of the building “Villa Bosch”) by combining the objects “body” (of south wing), `roof` (of south wing) and further parts of the south wing. These objects can also be used in queries to the database.

![Class diagram for thematic information]

*fig. 4: class diagram for thematic information*

A further requirement on the data model was that it should allow queries to details of a facade of a building, like “Which parts belong to the northern facade of an object?” or “What are the properties of the window next to the entrance door?” In order to allow this, the main elements of
a facade are being modelled explicitly. This includes classes for *balcony*, *door*, *moulding*, *painting*, *window* or *ornament* which all can be attached to a part of the facade. So in a similar way as there are 3D-objects and their parts, there are surfaces which can be separated in several parts of a surface that can be addressed independently.

A part of the thematic model for buildings is depicted in figure 4: The classes *threeD_Obj*, *part_threeD_Obj*, *surface* and *part_surface* are used for realizing the corresponding main aspects of a *threeD_Object*, *part_threeD_Object*, *surface* (facade) and *part_surface*. Using these classes the properties of the corresponding sub-types are modelled.

As already explained, generalization allows not only minimizing redundancy when defining sub-types, but also results in an well extensible structure. The integration of new subtypes can be achieved by defining inheritance relationships to the corresponding main class.

*fig. 5: class diagram for the relationship between thematic aspects and geometry*
In order to model the relationships between 3D-objects and their parts, base bodies and their facades as well as between facades and the objects belonging to a facade bi-directional 1:n relationships are being used.

In the realized prototype only parts of 3D-objects, facades and parts of facades are being linked to spatiotemporal data structures. This is very application specific. In order to change this relations easily these kind of relations are modelled using an extra class, which is called “themGeo”. This improves flexibility, for example to exchange the geometry model with a different description (e.g. GML, [22]).

Figure 5 shows the realized relationships between thematic and geometric data within the “4D”-model explained. The geometric description for the thematic classes part_threeD_Obj and part_surface is realized using the class cell3 (body) while for the thematic class surface the class cell2 (face) is being used. When there is only 3D information available for a part of a 3D-object or for parts of a facade, this can be expressed by the modeller through the usage of the hierarchical structure of the spatial model by using 3-Cells that only use a 2-Cell (face). The geometry of a 3D-object is represented through the geometries (3-cells) of the parts of the 3D-object.

5 An object-oriented model for temporal data

The object-orientated paradigm has also been used for the modelling of a general time framework. The range of possible different applications put quite complex requirements on temporal support. First it is necessary to identify the dimensions limiting the modelling space of a general temporal model. Further the components and properties have to be determined in order to be able to define an adaptable structure that fulfils the various requirements. From these a
framework for building temporal models was developed using the identified components. It supports design alternatives by the provision of a range of classes and accompanying properties. These temporal classes can be integrated with the models for the geometry and for the thematic aspects already introduced to a composite model for temporal 3D-geoobjects.

Regarding time one can distinguish the following general aspects:

- **Temporal Structure** – defines a structure using temporal primitives, domains and structures concerning temporal determination (certain or uncertain representations).

- **Temporal Order** - describes the possible types of orders of temporal structures.

- **Temporal History** - describes the semantic meaning of the different states the object.

- **Temporal Representation** - describes how to represent calendars and granularities.

**Temporal Structure**

The temporal structure defines through its parts a base for the temporal model. This temporal “structure” can have the following properties [12]:

1. **Temporal Primitives** are represented either as absolutes (anchored, “date”, e.g.: 5-9-1999) or relative (unanchored, “period of time”, e.g.: 30 days).

2. **Temporal domain**: It is possible to distinguish discrete and continuous domains. In the field of temporal databases a discrete time domain is usually being used.

3. **Temporal determination**: In the deterministic case complete and exact knowledge is available for temporal primitives. On the other hand these aren't determined exactly in indeterministic cases [4], e.g. fuzzy temporal borders.

The topmost level of the temporal structure-model consists of absolute (anchored) and relative (unanchored) temporal primitives. The next hierarchical level supplements the structure with domains, being either discrete or continuous. The deterministic and not-deterministic primitives
form the last component. A temporal structure consists of a combination of all of the represented
temporal primitives. Through the combination of the different properties offered within the three
levels of the hierarchy to model temporal aspects of the world it is possible to distinguish eleven
temporal types as “temporal primitives” (the twelfth one is only a theoretical combination as
“non-deterministic continuous time points (instants)” are not possible because of contradicting
properties). The temporal primitives represent the fundament for representing temporal data.
Further it is necessary to distinguish between the logical and physical representation of a time
value. If the time value is described by means of a calendar, it is a logical representation.
One can define a broad range of operations for the suggested data types. [12] defines a range of
categories for the operations according to their purpose and the types of arguments and results as
stated below. [5] explains the realized operators within our model in more detail.
- **Build-in-functions** allow the type conversion between temporal data types as well as
  combination- or comparison functions.
- **Arithmetical Operators** offer the corresponding adaptation of the basic arithmetic functions.
- **Comparison operations** give back a Boolean value (they are used for checking the
correctness of selection criteria)
- **Aggregation functions** The well-known aggregation functions from SQL like COUNT, SUM,
  AVG, MAX and MIN can also adapted for temporal data types.
fig. 6: class diagram of the prototypical implemented temporal structures

**Temporal Representation**

The proposed temporal primitive data types offer a basis for the representation of temporal data. For a temporal value that is represented by an instance of such a temporal data type it is necessary to distinguish between logical and physical representations of this value. If the value is represented using a calendar it is a logical representation. While supporting multiple logical calendars, the value of temporal data types is being stored independent from a calendar within the framework implemented. This means that the point of time is stored as a chronon of the base watch. But within the framework there are classes for different calendars available which define the logical representation through the definition of usable granularities, referencing the chronons.
of the base watch. They also offer functions for converting between the physical and logical representations of the temporal objects.

**Temporal Order**

The course of the time can be classified as linear, sub linear or branching. In both cases time is generally regarded as running linearly from past to future. They only differ regarding the handling of subordinate spatial basic types (primitives). In the linear case overlapping borders of temporal primitives are forbidden, while they are possible in the sub-linear case. A sub-linear order can also be used for managing indeterministic temporal phenomena. This can for example be used for the temporal description of the changes of an object which are only known roughly. Further the concept of branching order time allows to regard time to be linear only up to a certain point of time. A typical example would be town planning, where different planning alternatives can be managed in different branches of the resulting temporal tree. In each of the branches of that tree a partial order of time is defined.

**Temporal History Type**

One of the fundamental requirements for a temporal model is to represent the development of real world objects over time –regarding geometric, topological or thematic attributes. This is a basic functionality needed within a TGIS. This development of the object – the set of observations of these attributes within time - forms the "temporal history" of the object and can be distinguished in valid time and transaction time [13].

The valid time describes the time for which an entity of the real world is “valid” or a statement about the entity is true. E.g. „Fountain A is in front of the building B“.
The transaction time deals with the points of time when a value is being inserted into the database. A database management system that supports both aspects of time is generally being called “bi-temporal”.

Putting the Components Together

The focus of the following is on the relationships between the components defined through the design alternatives for a temporal model that have been explained so far.

A temporal model can support either one or many histories of the types **valid time**, **transaction time**, **event**- or **user defined histories**. Each of these histories consists of a set of temporal orders (being either linear, sub-linear or branching). For example the borders of structures that

---

**fig. 7: basic data types for a temporal model**
belong to linear orders cannot overlap. These represent a total temporal order. On the other hand is it possible for sub-linear or branching orders to have (absolute) overlapping temporal primitives. In the case of branching orders they may represent multiple partial temporal orders. Each of these temporal orders includes a temporal structure which consists either of all or a subset of the eleven temporal primitive types that have been introduced (fig. 7). For each of these temporal primitive types it is necessary to define a function to convert between the physical representation (real, integer, …) and one of the logical representations (e.g. “March 11, the 1971, 8:22:45”) and vice versa. In order to allow a maximum degree of flexibility, it is possible to define different calendars that can be related to the temporal primitives. This relationship is represented in figure 8 through a “has”-relationship. This figure shows a summary of the different alternatives for modeling a temporal structure.
Temporal Structure Design Space

Temporal Primitives

- Instants
  - Discrete Instants
  - Continuous Instants
- Intervals
  - Discrete Intervals
  - Continuous Intervals
- Spans
  - Discrete Spans
  - Continuous Spans

Anchored Primitives

- Discrete Instants
- Continuous Instants
- Discrete Intervals
- Continuous Intervals
- Discrete Spans
- Continuous Spans

Unanchored Primitives

Determinacy-Domain-based Temporal Primitives

- Determinate Discrete Instants
- Indeterminate Discrete Instants
- Determinate Continuous Instants
- Indeterminate Continuous Instants
- Determinate Discrete Intervals
- Indeterminate Discrete Intervals
- Determinate Continuous Intervals
- Indeterminate Continuous Intervals
- Determinate Discrete Spans
- Indeterminate Discrete Spans
- Determinate Continuous Spans
- Indeterminate Continuous Spans

fig 8: design space for temporal structures
7 Integrating geometry, thematic and temporal model

As explained object-oriented modelling allows to model the different aspects of geoobjects within their own class hierarchies and then afterwards to combine these by defining relationships between the classes. Figure 9 shows the relationships between the main classes of the resulting temporal 3D model. So how can this be applied?

When defining a temporal object within the proposed framework the first step involves the definition of the thematic structure. Each thematic structure can be linked with a temporal order to model the change of the (spatial) data of that object over time. Within each order the spatio-temporal relationship is being expressed using objects of the class 

These allow the aggregation of spatial and temporal information of the represented objects.

In order to make this work as explained it seems sensible to introduce structures (classes) for the coupling of the three data models into a common model. This is done using the new classes 

These do model the relationship between thematic objects and temporal orders on the one hand (themGeo) and the relationship between temporal and spatial parts of the model within a spatio-temporal structure on the other hand.

This way it is possible to model the classes of the thematically or spatial (partial) models through the definition of inheritance relationships of the classes themGeo or combCell. Through the decoupling of the structure that links the thematic with the spatial data model it is possible to replace one part of the model through a different one quite easily. This might be useful for adaptations to other application domains.
fig. 9: class diagram for the relationship between thematic, spatial and temporal data

8 Object-versus Attribute Time Stamping

Through assigning a temporal order to every time variant feature or attribute it is possible to model temporally changing spatial data. Object-time-stamping can be used to describe the changes of a complex object (e.g. GML-feature) over time. In this case the whole object (feature) will be time-stamped by adding a reference to an order.

If an object has only a few time-invariant-attributes attribute-time-stamping may be more efficient in order to reduce redundant data. In this case each time-variable attribute is extended by a reference to an order. This can be applied to any attribute of the feature, so that every attribute can have its own temporal constructs.
Zipf and Krüger [2001] [21] illustrate how such a link could be realized using an XML representation of geographic entities like GML. This shows that the temporal framework cannot only be coupled with 3D-objects as explained before, but also with other representations of geo-objects and their spatial as well as non-spatial attributes offering a high degree of flexibility.

But in not all cases one does have the full control of all available class packages that model domain issues, but often it is necessary to extend just existing software libraries. AOP now gives the possibility to do that without touching the actual code of the original software. This results in even greater freedom to mix domain models that deal with different aspects of real world into a new and richer domain representation. Zipf and Merdes [2003] [28] propose that these benefits can even be realized in a more automated way by using existing formal ontologies to derive aspect descriptions. This will be explained shortly in the following section.

9 Dynamical extensions of spatial class hierarchies with “aspects”

Let a class hierarchy for features be given - e.g. one based on OGC standards. Now we want to extend this through a “time”-aspect, which can be modeled in more or less sophisticated ways. In order to combine both, it is necessary to modify the original model by either enhancing the super-class or defining coupling classes. This might not be desirable or possible in all situations. In such cases a new paradigm of software engineering called “aspect-orientation” (AO) offers help to enhance existing class-libraries to enrich them with new (cross-cutting) aspects dynamically. But what is the benefit over conventional approaches?

Object-oriented-programming has become mainstream. There, in short, the classes assembling software have well defined responsibilities. But often some parts cannot be traced down as being the responsibility of only one class. These cross-cut several classes and affect whole system. One
can add code to each class separately in order to handle such parts - but that violates the basic rule that each class has well-defined responsibilities. Here comes Aspect Oriented Software Development (AOSD) [HTTP://WWW.AOSD.NET] into play: AOSD defines a new program or language construct, called an “aspect”. This allows capturing cross-cutting aspects of software in separate program entities. This new concept has recently been added to several programming languages as an extension. In Java it is called AspectJ [HTTP://WWW.ASPECTJ.ORG]. Aspects (in AspectJ) have much in common with classes: They can have methods and fields, extend normal java classes, implement interfaces, and may be abstract. They also can extend other aspects and can contain new constructs called pointcut and advice. Pointcuts provide a mechanism for specifying join points, i.e. well-defined points in the execution of the program. Examples for join points include object initialization, method calls, and field access. When defining a join point related to a method call it is possible to use powerful wildcards semantics for the method signature including name, arguments, return type as well as target object. The definition of an executable piece of functionality is called advice. An advice is defined with respect to a pointcut and can be run in a variety of ways, e.g. before, after, or even instead of a method call. Elements of the surrounding non-aspect code such as method call parameters can be made accessible within an advice. AspectJ also offers a mechanism for adding elements (fields, methods) to existing classes and change the inheritance and interface structure. This mechanism is called introduction. Introduction effectively changes the static structure of a program at compile-time as opposed to the dynamic nature of join points.

An example for combining spatial and temporal models on the fly is presented using Java and AspectJ notation:

```java
1 package aspectExample;
3 import org.eml.modell_4d.temporal.structure.interv;
4 import org.eml.deepmap.gml.objects.*;
6 // simplified example interface for time dependency------
```
First, a standard Java interface named `TimeDependent` is being defined. This is greatly simplified for the sake of clarity. Then an AspectJ aspect named `TimeDependency` is being declared. This aspect will contain all program elements relevant to the temporal modeling. Here these elements are static introductions such as parent, constructor, method, and field introductions, as well as pointcut and advice definitions. The aspect shall affect all members of the two separate class hierarchies with the root classes `Geometry` and `Feature`. Therefore new parents are being introduced into both classes in line 16. These implement the interface `TimeDependent` as if it was declared in the original source code. In order to be valid it is necessary to introduce both methods of the interface into both classes (lines 31-39). Both methods reference an instance
variable named validTime of type interv (which was introduced into the Feature and Geometry classes in line 19).

The constructor implementation additionally initializes the introduced instance variable validTime. This constructor is introduced into all members of both class hierarchies individually. This allows the new constructors to be used as if they were declared within the respective class definitions: Box box = new Box( new interv() ). Boxes can then be created with a time interval constructor argument just like any other subclass of Geometry or Feature. These static introductions change the class structure, hierarchy, and dependencies at compile-time. They do this in a crosscutting manner, which means that they affect a lot of different and (potentially) unrelated files from a single aspect definition.

In the aspect we further some pointcut and advice definitions can be found. These modify the behavior of the classes at run-time. In the example given a composite pointcut named DesiredGetterMethods is being defined (lines 46-49). It selects certain getter-methods of objects of type TimeDependent, that is, instances of Geometry, Feature or any of their subclasses.

Now the pointcut DesiredGetterMethods can be used to define a piece of advice, i.e., the functionality that is to be executed before, after, or instead of the methods selected by the pointcut. In the example, the former behavior of all getter-methods (that is, all access methods) is being replaced by a new, time-dependent behavior. This is done through a around advice (lines 52-58). For the sake of simplicity it is only checked if the time of the method invocation is within the time span defined as valid. If not, null is returned. Otherwise the keyword proceed signals to proceed as usually (line 57).

This illustrated some advantages of using aspects to enhance GIS data models and class libraries, like:
- Extension of the functionality of an existing GIS library without modification of its sources
- Combination of an existing GIS library with an unrelated library from a different domain
- Modification of the behavior of classes scattered over many places in the source code in a single aspect

Similarly, it is possible to enhance other existing non-spatial-aware domain-models with spatial “aspects” in order to spatially enrich them. More classical examples include to “weave in” middleware-style features (e.g. logging, tracing, performance monitoring, persistence, error-handling, security, distribution) into existing GIS class libraries.

10 Conclusions

For a multitude of geographic applications and in particular for an information system covering the aspects of town history the management of changes in feature data (in our case buildings) is of particular importance. But since the temporal model introduced is very generic it can be used in different domains. The framework is capable to cover both valid-time as well as transaction-time in a flexible way. This is done by providing the necessary building blocks for temporal structures like intervals, time spans or instants. Each of these can be subdivided being either continuous or discrete on the one hand and determinate or indeterminate on the other hand. By supporting the definition of additional application specific calendars with their respective granularities the framework supports all the notions and notations of time that can be considered relevant for present practical applications.
fig 10: realized class diagram of the temporal framework

Further a prototype has been developed that allows editing and querying spatiotemporal features also through a graphical user interface (fig. 11 & 12).
The development of flexible and efficient temporal 3D-GIS is an attractive but demanding task for further GIS research. A „4D“-GIS that covers all aspects of GIS functions from data handling, -analysis and -visualization equally well will not appear within a short time. Spatio-temporal analysis possibilities should include and perform the analysis of attributes, geometry and topology equally well. For example there is a lack of research on the possible changes of
topological relationships in „4D“ space-time in particular within vector-oriented GIS. Similarly the sparse availability of functions for inter- and extrapolation within vector –oriented “4D”-GIS is not satisfying. As existing GIS use proprietary data models that could not be extended easily it soon became clear that only the development from scratch could cover all the requirements for a “4D”-database. This situation will change with the spreading of more open data models. Further research and developments are of course necessary – ranging on the technical side regarding the implementation of multi-dimensional indices for the efficient access to large data sets, to the query languages for 4D queries to representing moving objects.

Acknowledgments

This work has been undertaken in the context of the project Deep Map at the EML, supported by Klaus Tschira Foundation (KTS), and the BMBF project SMARTKOM. Special thanks go to Sven Krüger, now at Quadox AG, Germany, for contributing considerably to the modeling and implementing the model.

References


dimensionale Geo-Objekte im Deep Map-Projekt. Diplomarbeit. Institut für Praktische Informatik und
Medieninformatik, Technische Universität Ilmenau.


system. In: Fesemaiier, D. Klein, S. and Buhalis, D. (Eds.): Information and Communication Technologies in

http://www.opengis.org/techno/specs.htm

Informatics, University of Oslo.

Department of Informatics, University of Oslo.


International Conference on Management of Data. 236-246.


Annual Conference, San Jose, California, USA, American Society of Photogrammetry and Remote Sensing,

Geographical Information Systems, 8(4), pp. 385-399.


