Proceedings of the 7th Hellenic Conference on Informatics (HCI '99)
University of Ioannina, Greece, pp. III 53-60, 26-29 August, 1999

A METHODOLOGY AND A TOOL FOR SPATIOTEMPORAL DATABASE DESIGN

Nectaria Tryfona, S∅ren Andersen, S∅ren Rolander Mogensen and Christian S. Jensen

Computer Science Department, Aalborg University
Fredrik Bajers Vej 7E, DK-9220 Aalborg ∅st, Denmark
{tryfona, san, moge, csj}@cs.auc.dk

1. SUMMARY

This paper concerns a methodology and its supporting prototype tool for database design of spatiotemporal applications. The methodology focuses on the main phases of conceptual and logical modeling with each phase being accompanied by models specifically constructed to handle spatiotemporal peculiarities. A database design tool that guides the designer through the conceptual and logical modeling as well as implementation, while dealing with applications involving space and time, is further presented. Starting from the conceptual modeling phase, the tool provides a specific environment to support the SpatioTemporal Entity-Relationship (STER) model, an extension of the Entity-Relationship model, towards the spatial and temporal dimension. An intermediate representation phase, namely, the logical phase, follows; in this, conceptual schemata are mapped into maps and relations, using an extension of the relational model, the SpatioTemporal Relational model (STR). Translation rules from conceptual to logical schemata are given. The resulted logical schemata are further translated into different underlying target DBMSs with spatial support; Oracle and the Spatial Data Option are used as a prototype. The STER and STR models, as well as the proposed tool are tested with extended examples from real applications.

2. INTRODUCTION

Spatiotemporal applications deal with the analysis, modeling, retrieval and representation of time varying, geo-referenced information. Typical examples are cadastral systems that capture histories of landparcels, routing systems computing possible routes of vehicles, and forecast-prediction systems. Database design holds a central role in spatiotemporal applications development. In order to enjoy the benefits of portability, expandability, ease-of-maintenance, and–most importantly–correctness, database design often follows a complete methodology cycle, including, among others, the phases of *conceptual* and *logical modeling* [3]. The conceptual modeling phase focuses on the representation of the application requirements in a way that uses no computer metaphors, is understandable to the user, and that is complete, so that it can be translated into the logical phase that follows without any further user input. Popular conceptual models include the Entity-Relationship model [2] and OMT [9]. The logical modeling phase describes formally the static properties and integrity constraints of the application. It results in a schema depended on the specific architecture of a Data Base Management System (DBMS), but independent of the target software. The most well-known and extensively used logical model is the relational.

For conventional applications, exemplified by the "supplier-supplies-part" paradigm, software tools that support database design methodologies *do* exist [6]. Their main goal is to (a) provide the appropriate environment for each phase of a methodology; for example, graphical tools for conceptual modeling, or a Data Definition Language at the logical phase, and (b) to facilitate the transition from each phase to the next one, e.g., from the logical schema to implementation specifications.

This work focuses on the development of a database design tool for spatiotemporal applications. Although, in the last years, considerable research efforts have been performed in extending conceptual and logical models [4] to accommodate the needs of a spatiotemporal environment, little has been done on the development of tools incorporating these results [8] [13]. Our goal is to provide a tool that guides the developer through the main phases of a database design methodology for spatiotemporal applications. The tool is built on top of existing models and DBMSs. Starting from the conceptual modeling phase, it provides a specific environment to support the SpatioTemporal Entity-Relationship (STER) model, an extension of the Entity-Relationship model, towards the spatial and temporal dimension. An intermediate representation phase, namely, the logical phase, follows; in this, conceptual schemata are mapped into maps and relations, using an extension of the relational model, the SpatioTemporal Relational model (STR). Translation rules from conceptual to logical schemata are given. The resulted logical schemata can be further translated into different underlying target systems with spatial support; Oracle and its Spatial Data Option [5] have been used as a prototypical example. The tool itself is a prototype and part of a larger research effort, centered around spatiotemporal applications and tools, starting with the analysis of their requirements [7], continuing with the study of conceptual [11] [12] and logical [10] spatiotemporal modeling and closing with the development of a tool that supports these phases.

Next, we discuss the main phases of the methodology our design tool is based on. We give the general architecture of the developed tool, its components and their interaction. The functionality and characteristics of the tool are presented. We conclude with the results and future work.

3. A SPATIOTEMPORAL DATABASE DESIGN METHODOLOGY

A database design methodology consists of a series of ordered phases, leading from loosely specified user requirements to the implementation. Basic among them are the phases of conceptual and logical modeling and implementation [3].

Due to large and intricate set of spatial and temporal needs, the available models and tools to support database design methodologies, are not adequate for spatiotemporal applications. Our thesis is that, spatiotemporal applications design, on the one hand, should take advantage of the benefits of using standard models and tools available for each phase, while on the other, these models and tools should be extended to accommodate the spatiotemporal peculiarities.

3.1 The conceptual modeling phase

Under this perspective, for the conceptual modeling phase, we proposed [11] a set of semantic constructs to accommodate spatiotemporal peculiarities. We do not invite the designers to use a specific conceptual model. We rather argue that by enhancing any conceptual model with these constructs, the resulted model handles efficiently spatiotemporal information. In [11] the Entity-Relationship Model has been chosen as a prototypical environment, resulting into SpatioTemporal ER (STER). STER includes classical ER constructs, i.e., entity sets, relationship sets and attributes, with built-in spatial, temporal, and spatiotemporal functionality. A construct that captures a temporal, spatial, and spatiotemporal aspect is called *temporal*, *spatial*, or *spatiotemporal*, respectively. STER is accompanied by a textual language; every diagrammatic schema can be translated into a textual schema and vice versa. Figure 1 represents landparcels with "soil type" (shaded) as property inherited from space.

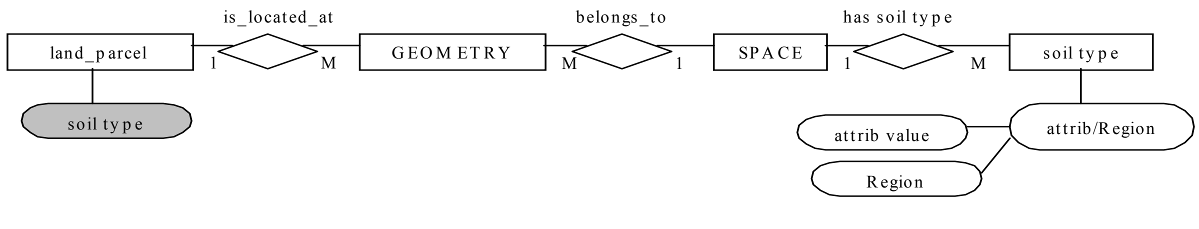


Figure 1: A conceptual modeling pattern capturing soil type of landparcels in STER.

Moreover, while applying STER in real environments [1] [14], it became clear that, many times, specific, autonomous and semantically rich parts of the STER schema need to be repeated in an application schema in order to capture conceptually similar situations. For example, the fact that an entity is spatial, or an attribute is spatiotemporal. We proposed [10] to capture autonomous and semantically meaningful excerpts of diagrams that occur frequently as *spatiotemporal modeling patterns*. To facilitate the conceptual design process, we proposed the abbreviation of these patterns by corresponding spatial, temporal and spatiotemporal *pattern abstractions*, termed *components*. Figure 2 shows the semantically equivalent representation of Figure 1, by using modeling components. "R" indicates that "soil type" has different values per region, and so it is captured in regions.

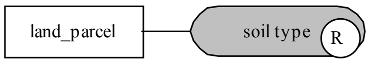


Figure 2: "soil type" of landparcels with the use of components.

The basic outcome of using patterns and components at the conceptual phase is the reduced complexity of the resulted schemata. Modeling patterns and components can be used in two ways: (a) *bottom-up*: the designer captures all the information of a specific application in a detailed diagram and, next, by marking (i.e., recognizing) the appropriate constructs (i.e., forming a pattern), replaces them by the corresponding component, and (b) *top-down*: the expert user or the designer adds the spatial, temporal, or spatiotemporal dimension to an application schema by plugging-in the appropriate component. Table 1 shows an excerpt of the library of spatial, temporal and spatiotemporal components. The spatial extent, indicated by a "s", can be "P" (point), "L" (line) and/or "R" (region). The temporal extent can be existence time ("et"), valid time ("vt") and transaction time ("tt").

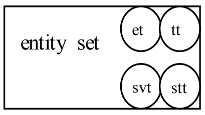
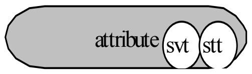
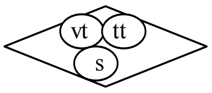
Descritpion	Component	Rule of Usage
Spatio-Temporal Entity set		Entity set, with spatial extent for which valid time and transaction time are recorded. Existence time and transaction time of the entity set are also recorded.
Spatio-Temporal Attribute		Spatial attribute for which valid andtransaction time are recorded.
Spatio-Temporal Relationship Set		Relationship set with spatial extent. Valid time and transaction time of the relationship set are also recorded.

Table 1: Spatial, temporal and spatiotemporal components in STER and the rules of their usage.

3.2 The logical modeling phase

The conceptual modeling phase is followed be the logical one, which serves as an intermediate translation from the conceptual schema to implementation specifications. A formal description of static properties and integrity constraints is given using a logical model. The logical modeling phase leads to two schemata: (a) the logical schema, which includes the definitions of objects, their attributes and their relationships and (b) the external schema, which is the integration of all user views. (Usually, the term "logical schema" is used for both).

Theoretical research into the logical phase [10], as well as experience from real applications [1] [14] show that, for the logical schema the designer needs techniques and tools to:

- Define *attributes of space*, organize them into *layers* (or maps) and give them the *temporal dimension* (resulting into spatiotemporal attributes). For example, "soil_erosion" is a property of space organized in a layer, representing sets of regions (with different values). Time (valid and transaction) need to be recorded for the layer.

- Specify *non-spatial attributes*, organize them in *relations*, assign them to time and connect them to spatial ones. For example, a relation "types_of_erosion" describing different kinds of erosion (i.e., descriptive information such as "10%-30%", "30%-60%" "60%-90%", etc) is connected to the corresponding layer ("soil_erosion"), which has codes (such as "low", "medium", "high", etc) for the represented type of erosion in each region.
- Specify *spatiotemporal integrity constraints*, imposed either by the user, or by the designer for the integrity of the database. For example: if the "soil_erosion" is more than 60%, and a "pipe" of a utility network located there, is more than 8 years old, then it needs to be replaced in the next year.
- Define *complex attributes*, computable from others, spatiotemporal or not. For example, "soil_type" depends on "soil_erosion" and "soil_acidity."
- Organize *complex (virtual) layers* defined on common geometric features. (We use the term "virtual" layers to denote such collections of computable spatial attributes.) An example is "soil_type" of an area.
- Define *complex classes of spatiotemporal objects*. Spatiotemporal objects have positions of some geometric type (point, line, region or combination there of); they combine spatial and non-spatial attributes, with temporal or not, dimension from several layers and adhere to spatiotemporal or other integrity constraints. A "landparcel" is an object with "soil_type" and "soil_erosion." A "utility network" passes through it, consisting of "pipe-segments" with "erosion" which depends on the "soil_erosion."

Having the relational as the starting point (there are many good reasons to make use of the sound theory behind it [10]). The resulted model, the SpatioTemporal Relational model (STR) allows for the definition of layers, relations, virtual layers, objects, and constraints. The model is formal, yet practical, and constitutes part of a full automatable application design methodology. Next we provide an excerpt of its syntax:

DEFINE LAYER n <layer_name>
<VALID TIME [start_time, end_time] >
<TRANSACTION TIME [start_time, end_time]>
ATTR (attr_name$_1^n$, Domain$_1^n$, <UNIQUE>),…,
(attr_name$_{mn}^n$, Domain$_{mn}^n$, <UNIQUE>)
GEOMETRIC TYPE Geometric_type
<POSITIONING Coordinate_system>
<CONSTRAINT spatiotemporal_condition>

DEFINE OBJECT CLASS obj_class_name
<GEOMETRIC TYPE Geometric_type>
<SUBTYPE OF sup_obj_class>
<ON LAYERS layer_id$_1$,…,layer_id$_k$>
<WITH ATTRIBUTES attr_name$_1$,…,attr_name$_m$>
<CONSTRAINT constraint_name>

Note that, STR is independent of platform–as it is not based on the extension of any commercial relational system–but depended on specific architecture (i.e., relational).

4. THE SPATIOTEMPORAL DATABASE DESIGN TOOL

Based on the methodology and its supporting models (i.e., STER and STR) presented in Section 3, we built a spatiotemporal database design tool. Here we give an overview of it. The final goal is to guide the designer through the main phases of the methodology, by providing the appropriate for each phase environment, and facilitating the transition from each phase to the next one. The tool is semi-automatic, in the sense, that some steps are transformed into another without any further input from the designer. For example, transformation of the STER schema into a syntactical conceptual language, while others are based on design decisions, for example how are complex (i.e., computable) layers produced.

Database design tools are, often, parts of larger, Computer-Aided Software Engineering (CASE) tools, supporting not only the design, but also the management and maintenance of the database. Following the categorization of CASE tools, presented in [6], database design tools can be divided into three categories: (a) *upper tools*, which handle the management of the database specification by supporting strategy, planning and the construction of the conceptual level of a methodology. For this purpose the upper tools support traditional diagrammatic languages such as Entity-Relationship Diagrams, Data Flow Diagrams and Structure Chart, and provide drawing, storage and documentation facilities. (b) *lower tools*, which utilize mapping algorithms to transform formal specifications into an executable form. The starting point of the lower tool is the conceptual schema. (c) *integrated tools*, which provide support for both the early stages as well as the implementation stages of a database. Note that a tool can be in any of the above categories, without providing all the described activities [6]. For example, an ER diagram generator is considered an upper tool, since it facilitates the conceptual modeling phase, without any further support.

Next, we describe the components and functionality of our prototype spatiotemporal database design tool.

4.1 Components and functionality of the spatiotemporal database design tool

The spatiotemporal database design tool has two major components: (a) the CASTER (Computer-Aided SpatioTemporal Entity Relationship) component, which supports the conceptual modeling phase, and (b) the CASTR (Computer-Aided SpatioTemporal Relational) component for the logical modeling phase. It is built on top of Oracle, using the Spatial Data Option [5] for spatial support. Figure 3 shows the basic architecture of the tool, while the next sections describe each component.

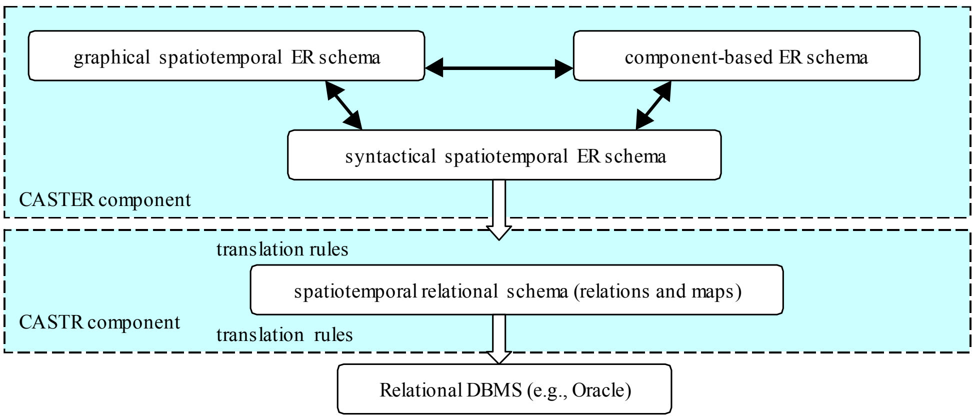

graphical spatiotemporal ER schema
component-based ER schema
syntactical spatiotemporal ER schema
CASTER component
translation rules
spatiotemporal relational schema (relations and maps)
CASTR component
translation rules
Relational DBMS (e.g., Oracle)

Figure 3: The architecture of the spatiotemporal database design tool.

4.1.1 *The CASTER component*

Based on the categorization of database design tools, the CASTER component acts as an upper tool. It allows for the definition of an STER schema in any of the three formats: (a) fully-detailed diagrammatic description, using the spatial, temporal and spatiotemporal constructs of STER, (b) component-based description, using the spatial, temporal and spatiotemporal components of the implemented library of Table 1, and (c) textual description, using the syntactical language of STER. Any of these can be translated to any other. Each format can be translated to any of the two others, without loss of information. It further provides drawing, storage and documentation facilities. Mechanisms to check the violation of integrity constraints (e.g., no two entities can be connected without a relationship) have also been included.

As said in Section 3.1, large, semantically meaningful STER diagrams form patterns and can be replaced by modeling components, to facilitate the design process. In Figure 4, "land_parcel" is depicted as a spatiotemporal entity set, and as such can be replaced by the component for spatiotemporal entities. Figure 5 shows the semantically equivalent component for "land_parcel" and the way it is constructed; an entity set is firstly created and spatial and temporal characteristics for it can be selected.

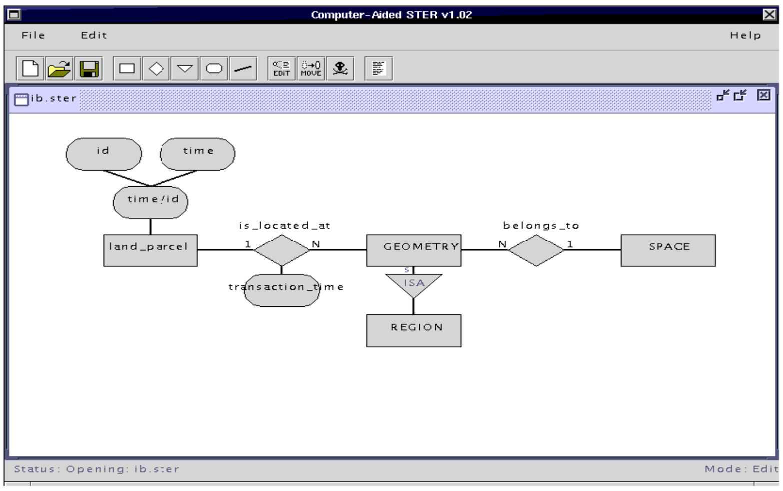
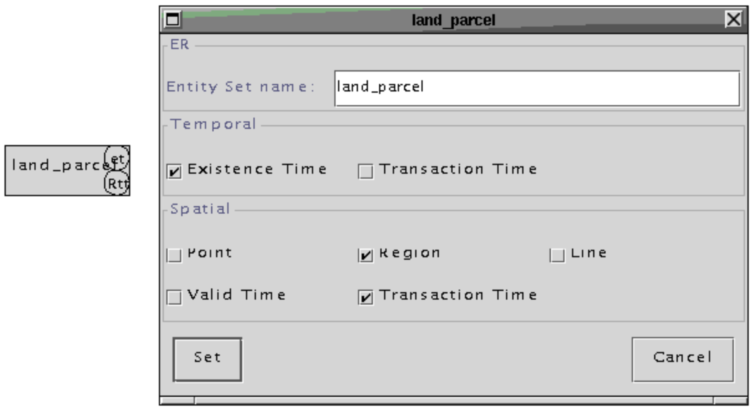


Figure 4: A STER diagram in the CASTER component.



land_parcel

ER

Entity Set name: land_parcel

Temporal

[✓ Existence Time] Transaction Time

Spatial

Point [✓ Region] Line

Valid Time [✓ Transaction Time]

Set Cancel

Figure 5: The dialog box for editing a "land_parcel" as a spatiotemporal entity set.

4.1.2 *The CASTR component*

The CASTR component acts as a lower tool: having the conceptual schema as a starting point, it uses a set of translation rules, to transform that schema to one closer to the implementation phase and then to an executable form. This is done in two phases:

(a) first, the STER schema is translated into an STR schema, following specific rules. The general outline of the rules for the translation from STER constructs (i.e., entity sets, attributes and relationship sets with spatial, temporal and/or spatiotemporal extend) to STR constructs (relations, layers, virtual layers, objects and constraints) is as follows.

- A conventional (i.e., non-spatial, non-temporal) entity set is translated into a relation.
- A spatial entity set is translated into a relation and a layer. More specifically, the spatial part forms a layer with the same geometry, while the non-spatial forms a relation.
- A temporal entity set is translated into a relation.
- A spatiotemporal entity set is translated into a relation and a layer.

- A conventional attribute is translated into a relation field. If the attribute is temporal, another field capturing time is created.
- A spatial attribute is translated into a layer.
- A conventional relationship set is translated into a relation.
- A spatial relationship set is translated into a relation or operation among layers, resulting into a virtual layer, as the outcome of the operation among these layers.
- A temporal relationship set is translated into a relation.
- A spatiotemporal relationship set is translated into a relation or operation among layers, resulting into a virtual layer, as the outcome of the operation among these layers. The temporal dimension of the relationship is recorded in the virtual layer.

(b) second, the STR schema is translated into Oracle specifications. The basic rules are:

- A STR relation becomes an Oracle table.
- A STR layer results into 4 tables, supported by the Spatial Data Option [5]: (i) one describing the dimensions of space being modeled, i.e., the upper and lower bounds of the dimensional axes, their granularity, and names, (ii) one describing the geometric figures of the layer, (iii) one for the spatial index of the geometric figures, and (iv) one containing information about the other tables such as number of ordinates per row, and tessellation level used during the indexing of the layer.
- Direct constraints, involving domains, and key-related issues are implemented in Oracle.
- Virtual layers, objects and application constraints imposed by the user or the application itself, are implemented as views. If spatial data is involved then virtual layers are implemented on top of the specific Geographic Information System (e.g., Arc/Info), which operates coupled with Oracle.

The STR schema acts as an intermediate step, bringing the application schema closer to implementation, but still one level independent of specific DBMS's. In that way, any another DBMS or spatial cartridge with spatial support can be used instead of Oracle and the Spatial Data Option.

Further, the CASTR component allows for the storage and retrieval of both logical and external schemata. As objects, virtual layers and constraints are part of the user view, the designer can create them, by using the CASTR component. For example, if "soil_erosion" and "soil_acidity" are present in the logical schema, layer "soil_type" can be created in the external schema, by giving only its non-spatial attributes and the basic geometric characteristics–region in this case. Later on, the spatial software we translate this definition into an actual map.

4.2 Characteristics of the spatiotemporal database design tool

The presented spatiotemporal design tool is an *integrated* tool, since it provides support for both the early stages as well as the implementation stages of an application. Its general features are:

- *User Interaction.* The user interface permits the designer to work with a graphical and textual description of the information system. The user can create, load, or save diagrams through either the tool bar or the menu bar. Multiple diagrams can be open at the same time. The diagrams can be edited either through the default graphical user interface or through a textual editor, accessible through either tool or menu bar.
- *Verification Support.* Both textual and graphical constraints are implemented in CASTER to ensure the integrity of the conceptual schema. For example, the graphical STER editor makes sure that two entity sets cannot be directly connected to each other or that relationship sets cannot be directly connected to other relationship sets. Similarly, in the textual STER editor, a syntax check is performed.
- *Modeling Support.* STER and STR schemas can be created, and the following transformations are possible: (i) a component-based STER, a syntactical STER and a diagrammatic STER can be

transformed to each other, (ii) Any STER description can be transformed to a STR schema, (iii) a STR schema can be transformed to an Oracle schema.

5. CONCLUSIONS AND FURTHER RESEARCH

An overview of a spatiotemporal database design tool guiding the designer from the conceptual modeling phase to implementation is presented. It supports both high-level activities, such as creation, storage and retrieval of a conceptual schema, as well as low-level ones, such as rules for the translation of a conceptual schema to implementation. Additionally, it allows for the mapping of any conceptual schema into different underlying target DBMSs with spatial support; Oracle and its Spatial Data Option have been used as a prototype. We are currently working on the support of spatiotemporal constraints at the conceptual modeling phase. Further, the mechanisms to ensure the preservation of these constraints at the logical phase are to be included.

Acknowledgments. Supported by the CHOROCHRONOS project, funded by the European Commission DG XII Science, Research and Development, as a Network Activity of the TMR Program, no. FMRX-CT96-0056. The prototypical tool is available at: http://www.cs.auc.dk/research/DBS/software/CASTAD.

6. REFERENCES

[1] Cadastral, *Definition of a Standard for the Exchange of Digital Cadastral Data*. Techical Report. National Technical University of Athens funded by CADASTRE SA, Greece (1997).

[2] P.S. Chen, The Entity-Relationship Model: Toward a unified view of Data. ACM TODS, 1(1) (1976).

[3] R Elmasri and S.B. Navathe, *Fundamentals of Database Design*. Addison-Wesley (1994).

[4] R. Guetting and M. Schneider, Realm-Based Spatial Data Types: The Rose Algebra. VLDB Journal 4 (1995).

[5] J. Hebert, Oracle8 Spatial Cartridge User's Guide and Reference. Oracle Coorporation (1997).

[6] P. Loucopoulos and R. Zicari*, Conceptual Modeling, Databases and CASE.* Wiley&Sons (1992).

[7] D. Pfoser and N. Tryfona, Requirements, Definitions and Notations for Spatiotemporal Application Environment. 6th ACM Workshop on Advances in GIS. ACM Press (1998).

[8] N. Pissinou, K. Makki, and N. Park, Towards the Design and Development of a New Architecture for Geographic Information Systems. Conference on Information Knowledge Management (1993).

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

[10] N. Tryfona and T. Hadzilacos, Logical Data Modeling of SpatioTemporal Applications: Definitions and a Model. Proc. of the Database Engineering and Applications Symposium (1998).

[11] N. Tryfona and C.S. Jensen, C. S., Conceptual Data Modeling for Spatio-Temporal Applications. Geoinformatica Journal (in press) (1999).

[12] N. Tryfona and C.S. Jensen, A Component-Based Conceptual Model for Spatiotemporal Applications Design. Chorochronos TR CH-98-10, under submission (1999).

[13] M.A. Vaz Salles, F. Pires, C.B. Medeiros, and J.L. de Oliviera, Development of a Computer Aided Geographic Database Design System. Proc. 13th Brazilian Database Symposium (1998).

[14] UtilNets, Work Program of Brite-Euram Project 7120, DG XII, European Union (1994-1997).