Spatio-temporal and Multi-representation Modeling: A Contribution to Active Conceptual Modeling

Stefano Spaccapietra[1], Christine Parent[2], and Esteban Zimányi[3]

[1] Database Laboratory, Ecole Polytechnique Fédérale de Lausanne,
CH-1015 Lausanne, Switzerland
`Stefano.Spaccapietra@epfl.ch`
[2] HEC ISI, University of Lausanne,
CH-1015 Lausanne, Switzerland
`Christine.Parent@unil.ch`
[3] Department of Computer & Decision Engineering (CoDE), Université Libre de Bruxelles,
50 av. F.D. Roosevelt, 1050 Bruxelles, Belgium
`ezimanyi@ulb.ac.be`

Abstract. Worldwide globalization increases the complexity of problem solving and decision-making, whatever the endeavor is. This calls for a more accurate and complete understanding of underlying data, processes and events. Data representations have to be as accurate as possible, spanning from the current status of affairs to its past and future statuses, so that it becomes feasible, in particular, to elaborate strategies for the future based on an analysis of past events. Active conceptual modeling is a new framework intended to describe all aspects of a domain. It expands the traditional modeling scope to include, among others, the ability to memorize and use knowledge about the spatial and temporal context of the phenomena of interest, as well as the ability to analyze the same elements under different perspectives. In this paper we show how these advanced modeling features are provided by the MADS conceptual model.

Keywords: Active conceptual models, spatio-temporal information, multiple representations, multiple perspectives, MADS model.

1 Introduction

Globalization has significantly increased the complexity of the problems we face in many different areas, e.g., economic, social, and environmental. Events and phenomena have intricate and subtle interdependencies whose perception changes when seen from different perspectives. This calls for more knowledgeable data management systems, capable of managing a more comprehensive description of the world. Knowledge of cause-effect relationships between events, for example, contributes to the evaluation of the future impact of solutions to problems in a decision-making environment. Analyses of past events and phenomena supports decision makers with the possibility to learn from past experiences and to take them into account in choosing future actions. This is a very traditional learning pattern, but poorly supported by current data management systems.

More knowledgeable systems are the target of active modeling, a paradigm defined as a continual process describing the important and relevant aspects of the real world, including the activities and changes, under different perspectives [1]. It aims at providing control and traceability for the evolving and changing world state, helping to understand the relationships among changes. An active conceptual model provides at any given time a multilevel and multi-perspective high-level abstraction of reality. Consequently, one of its basic features is integrating time and space modeling[1].

Unfortunately, the current established conceptual modeling practices do not achieve these objectives. They are still limited to mainly deal with organizing basic data structures, e.g. describing entities, relationships, properties and possibly processes. Other directions emerge that slowly influence progress in DBMS development. Supporting the description of spatial features has recently become part of the functionality of most recent DBMS, although to a still rather primitive extent. Temporal information is most frequently interleaved with spatial information, but is very poorly supported, which entangles the development of a multitude of applications that have to deal with spatio-temporal data, e.g. moving objects.

In this paper we show how spatial and temporal characteristics of phenomena can be captured into a conceptual model. We also discuss how to provide multiple representations of phenomena, thus allowing to analyze them under different perspectives, e.g., with respect to different stakeholders. Concrete modeling concepts are proposed using the constructs provided by the MADS conceptual model [2]. For readers already familiar with MADS, the paper offers a short presentation of the major features of the model.

2 Conceptual Modeling: The MADS Approach

Conventional conceptual models and DBMS have been tailored to manage a static view of the world of interest. They capture the state of affairs at a given moment in time (in the terminology of the temporal database community, they capture a snapshot). Active modeling stems from a dynamic view of the world as something that is continuously changing and where knowledge about the changes is as important as knowledge about the current status. This view entails their focus on capturing discrete as well as continual changes (i.e. time-varying information) and cause-effect relationships that help in understanding these changes. Since work on temporal databases, temporal aspects have evolved into spatio-temporal aspects, which capture phenomena that vary in both space and time. Moving objects (e.g. people, parcels, cars, clouds) are typical examples of spatio-temporal objects. Thanks to the availability of mobile devices, e.g. GPS, and ubiquitous computing, interest in mobile objects applications has exploded in recent years. Active modeling appears as the natural evolution we need today to face such innovative applications.

[1] cf. the Call for Papers of the First International Workshop on Active Conceptual Modeling of Learning (ACM-L 2006), Tucson, Arizona, November 8, 2006.

Finally, conventional models poorly support different and multiple perspectives of the same real-world phenomena, another characteristic of modern applications that are ruled by decentralized paradigms. Decentralization entails diversity and complementarities. There is not anymore a single enterprise-wide apprehension of data. Instead, the flexibility offered by applications handling multiple perspectives is a decisive advantage towards better strategies and decision-making processes.

The MADS model has been defined as an answer to the above requirements. MADS [2,3] is indeed a conceptual spatio-temporal data model with multi-representation support. It handles its four modeling dimensions - structural, spatial, temporal, and multi-representation – in a way that purposely makes the modeling dimensions *orthogonal* to each other (i.e., modeling in one dimension is not constrained by modeling choices in another dimension). Consequently, MADS can also be efficiently used in simpler environments, such as classical non-spatial, non-temporal, and mono-representation databases.

In terms of the framework for active conceptual modeling proposed in [4], MADS model covers several of the areas stated as necessary. It provides a comprehensive approach for *multi-level and multi-perspective modeling*, where multiplicity of level and perspectives can be handled at both the schema and instance levels. This guarantees maximum flexibility and precise match with specific application requirements. MADS allows both to customize the view of the system provided to different users, as well as cope with the inevitable inconsistencies and incompatibilities that arise when providing different perspectives of the same information. More precisely, MADS allows inconsistencies to exist if needed, btu guarantees that each perception is consistent if takes in isolation. *Space and time* features are smoothly integrated into the conceptual model allowing designers to precisely define how knowledge about where and when the events of interest occurred has to be kept. MADS aims to be an *executable conceptual model* in two different ways. First, it provides an associated query and manipulation language at the conceptual level, thus covering the full lifecycle of the information system in a uniform approach. Second, the conceptual specifications, whether for defining the schema or manipulating the database, are translated through CASE tools into the particular language provided by current implementation platforms (e.g., SQL) [3]. These tools provide users with visual interaction functionality, relieving them from textual languages and logical-level concerns. Finally, MADS definition language as well as its query and manipulation languages are formally defined.

MADS is not the only conceptual spatio-temporal model that has been described in the literature. The work reported in [5] also offers good support for spatio-temporal modeling, but does not address multiplicity of perspectives. Many more spatio-temporal data models are reported in [6]. Perceptory is the only spatio-temporal model that also proposes a mechanism for supporting multiple perspectives [7]. This mechanism is close to traditional database view techniques and does not have the full functionality MADS offers.

In the following sections we briefly describe some of the main characteristics of the four modeling dimensions of MADS, namely, structural, spatio-temporal, and multi-representation. We use as a running example a small excerpt of a real-word application coping with natural risk management in mountainous regions, e.g., avalanches and landslides (cf. Fig. 1). A full description of MADS is available in [2].

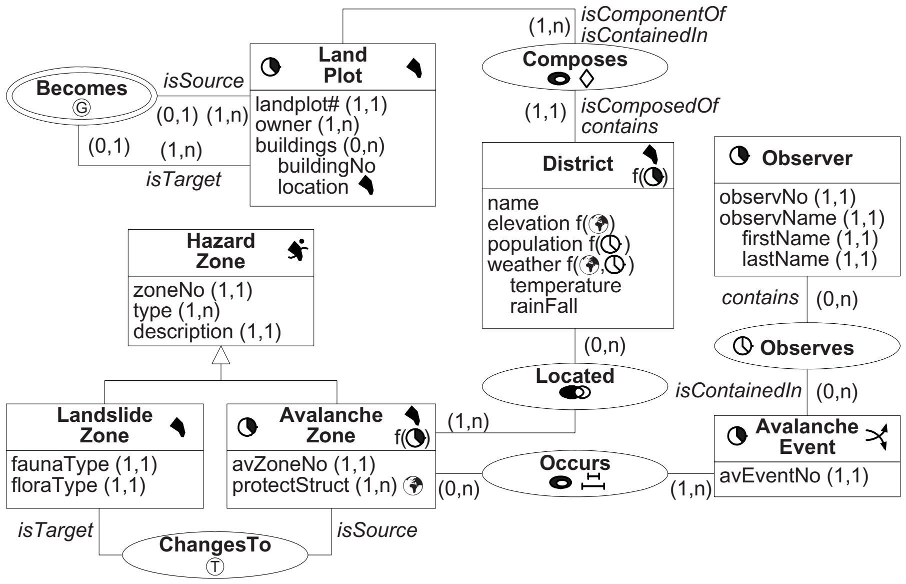


Fig. 1. An excerpt of an application for managing natural risks

3 Structural Modeling

MADS provides rich mechanisms for describing the data structure component of applications. In the description that follows we concentrate on novel aspects of MADS with respect to traditional conceptual models.

MADS structural dimension includes well-known features such as objects, relationships, attributes, and methods. Objects and relationships have an identity and may bear attributes. Attributes are mono-valued or multi-valued, simple or complex (i.e., composed of other attributes), optional or mandatory, and may be derived.

MADS identifies two basic kinds of relationship types, association and multi-association. An *association* is the typical kind of relationship type where each of its roles links one and only one instance of the linked object type. However, in some situations the basic association relationship does not allow to accurately represent real-word links existing between objects. For example, due to territorial reorganization, one or several land plots may be split/merged into one or several land plots. Fig. 1 shows a multi-association (drawn with a double oval) Becomes. This multi-association allows setting up a direct link between the set of land plots that is the input of the reorganization process and the set of land plots that is the output. Consequently, each role in a multi-association relationship type bears two pairs of (minimum, maximum) cardinalities. A first pair is the conventional one that, as in association relationship types, defines for each object instance, how many relationship instances it can be linked to via the role. The second pair defines for each relationship instance, how many object instances it can link with this role. Its value for minimum is at least 1. Obviously, an association is nothing but a special case of multi-association (with all maxima for the second cardinality pairs equal to 1). Pragmatic reasons (simplicity, frequency of use, user familiarity with the concept) make it nevertheless worth having associations as a separate construct.

Semantic data models usually provide the possibility to link objects through different types of relationships, each one with a specific semantics. MADS, instead, separates the definition of relationships into two facets. First, the appropriate link is built using either the association or the multi-association construct (this is the structural component of the link). Second, whenever needed, the link is given one or more specific semantics that convey the semantic implications of the link. MADS supports aggregation, generation, transition, topological, synchronization, and inter-representation semantics for the relationships. *Aggregation* is the most common one: It defines mereological (also termed component or part-of) semantics. An example is the relationship Composes in Fig. 1 (identified by the ◇ icon). *Generation* relationships record that one or several target objects have been generated by other source objects. An example in Fig. 1 is the Becomes relationship. Finally, *transition* semantics expresses that an object in a source object type has evolved to a new state that causes it to be instantiated in another target object type. For example, whenever efficient protection structures are built in an avalanche zone, the zone is no longer regarded as an avalanche zone but may become a zone subject to landslide: The instance moves from the AvalancheZone object type to the LandslideZone object type. If the application needs recording this change of classification, the ChangesTo transition relationship type (identified by the Ⓣ icon) can be added to the schema, as shown in Fig. 1. Other possible semantics for relationships are defined in the following subsections, to deal with spatial, temporal, and multi-representation features.

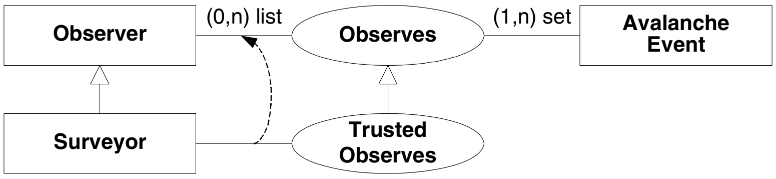


Fig. 2. A relationship subtype refining a role to link a subtype of the original object type

The generalization/specialization (or is-a) relationship allows relating a generic type, the supertype (e.g., HazardZone in Fig. 1) and a specific one, the subtype (AvalancheZone in Fig. 1) stating that they convey different representations of the same real-world phenomena. MADS supports is-a links between relationship types, with the same characteristics as between object types. In Fig. 2, the designer wants to create a separate relationship type for observations of avalanches made by surveyors, since any person may be introduced in the database as reporting an avalanche. A sub-relationship type may have additional properties and additional roles with respect to the super-relationship type. Further, as shown in the figure, existing roles may be refined as associated to a subtype of the otherwise inherited object type. The design ensures that only observations made by surveyors are registered as instances of TrustedObserves.

4 Spatio-temporal Modeling

In MADS space and time description is orthogonal to data structure description, which means that the description of a phenomenon may be enhanced by spatial and temporal features whatever data structure (i.e., object, relationship, attribute) has been chosen to represent it. MADS allows describing spatial and temporal features with either a discrete or a continuous view. These are described next.

The *discrete view* (or *object view*) of space and time defines the spatial and temporal extents of the phenomena of interest. The *spatial extent* is the set of 2-dimensional or 3-dimensional points (defined by their geographical coordinates <x,y> or <x,y,z>) that the phenomenon occupies in space. The *temporal extent* is the set of instants that the phenomenon occupies in time. Temporality in MADS corresponds to *valid time*, which conveys information on when a given fact, stored in the database, is considered valid from the application point of view.

Specific data types support the definition, manipulation, and querying of spatial and temporal values. MADS supports two hierarchies of dedicated data types, one for spatial data types, and one for temporal data types. Generic spatial (resp. temporal) data types allow describing object types whose instances may have different types of spatial extents. For example, a River object type may contain large rivers with an extent of type Surface and small rivers with an extent of type Line. Examples of spatial data types are: Geo (🌍), the most generic spatial data type, Surface (◆), and SurfaceBag (▲). The latter is useful for describing objects with a non-connected surface, like an archipelago. Examples of temporal data types are: Instant (◯), Interval (◑), and IntervalBag (⊛). The latter is useful for describing the periods of activity of non-continuous phenomena.

A *spatial* (*temporal*) *object type* is an object type that holds spatial (temporal) information pertaining to the object itself. For example, in Fig. 1 Landplot is both a spatial and temporal object type as shown by the Surface (◆) and the Interval (◑) icons on the right and left sides of the object type name. Following common practice, we call *spatio-temporal* an object type that either has both a spatial and a temporal extent, separately, or has a time-varying spatial extent, i.e., its spatial extent changes over time and the history of extent values is recorded (e.g., District and AvalancheZone in Fig. 1). Time and space-varying attributes are described hereinafter. Similarly, *spatial*, *temporal*, and *spatio-temporal relationship types* hold spatial and/or temporal information pertaining to the relationship as a whole, exactly as for an object type. For example, in Fig. 1, the Observes relationship type is temporal, of kind Instant, to record when observations are made.

The spatial and temporal extents of an object (or relationship) type are kept in dedicated system-defined attributes: geometry for the spatial extent and lifecycle for temporal extent. geometry is a spatial attribute (see below) with any spatial data type as domain. When representing a moving or deforming object (e.g. AvalancheZone), geometry is a time-varying spatial attribute. On the other hand, the lifecycle allows database users to record when, in the real world, the object (or link) was (or is planned to be) created and deleted. It may also support recording that an object is temporarily suspended, like an employee who is on temporary leave. Therefore the lifecycle of an instance says at each instant what is the status of the corresponding real world object (or link): scheduled, active, suspended, or disabled.

A *spatial* (*temporal*) *attribute* is a simple attribute whose domain of values belongs to one of the spatial (temporal) data types. Each object and relationship type, whether spatial, temporal, or plain, may have spatial, temporal, and spatio-temporal attributes. For example, in Fig. 1 the LandPlot object type includes, in addition to its spatial extent, a complex and multivalued attribute buildings whose second component attribute, location, is a spatial attribute describing, for each building, its spatial extent.

Practically, the implementation of a spatial attribute, as well as the one of a geometry attribute, varies according to the domain of the attribute. For instance, a geometry of kind Point (in 2D space) is usually implemented by a couple of coordinates <X,Y> for each value, a geometry of kind Surface by a list of couples <X,Y> per value, and a geometry of kind SurfaceBag by a list of lists of couples <X,Y> per value.

Constraining relationships are binary relationships linking spatial (or temporal) object types stating that the geometries (or lifecycles) of the linked objects must comply with a spatial (or temporal) constraint. For example, in Fig. 1 Composes is an aggregation and also a constraining relationship of kind topological inclusion, as shown by the icon. The constraint states that a district and a land plot may be linked only if the spatial extent of the district effectively contains the spatial extent of the land plot. Relationship types may simultaneously bear multiple semantics. For example, Occurs is both a topological and synchronization constraining relationship type. It ensures that both spatial and temporal extents of AvalancheZone are defined such that every avalanche event always occurs inside (topological constraint) a currently existing (synchronization within constraint) avalanche zone.

Beyond the discrete view, there is a need to support another perception of space and time, the *continuous view* (or *field view*). In the continuous view a phenomenon is perceived as a function associating to each point (or instant) of a spatial (or temporal) extent a value. In MADS the continuous view is supported by space (and/or time) *varying attributes*, which are attributes whose value is a function that records the history – and possibly the future – of the value. The domain of the function is a spatial (and/or temporal) extent. Its range can be a set of simple values (e.g., Real for temperature, Point for a moving car), a set of composite values if the attribute is complex as, in Fig. 1, weather, or a powerset of values if the attribute is multivalued.

District in Fig. 1 shows examples of varying attributes and their visual notation in MADS (e.g., f()). elevation is a space-varying attribute defined over the geometry of the district. It provides for each geographic point of the district its elevation. population is a time-varying attribute defined over a constant time interval, e.g. [1900-2006]. weather is a space and time-varying complex attribute which records for each point of the spatial extent of the district and for each instant of a constant time interval a composite value describing the weather at this location and this instant. Such attributes that are space and time-varying are also called *spatio-temporal attributes*. The geometry attribute can also be time-varying, like any spatial attribute. For instance in Fig. 1, both AvalancheZone and District have a time-varying geometry. The deformation of their spatial extent can therefore be recorded.

Practically, the implementation of a continuous time-varying attribute is usually made up of 1) a list of <instant, value> pairs that records the sample values, and 2) a method that performs linear interpolation between two sample values. For instance, a time-varying point would be implemented by a list of triples <instant, X,Y>. On the other hand, the deforming surface of District, which is time-varying in a step-wise manner, would be implemented by a list of couples <time interval, LIST(<X,Y>)>.

A constraining topological relationship may link moving or deforming objects, i.e., spatial objects whose geometries are time-varying. An example in Fig. 1 is the relationship Located linking District and AvalancheZone, which have both a geometry of domain time-varying surface. In this case two possible interpretations can be given to the topological predicate, depending on whether it must be satisfied either for at least one instant or for every instant belonging to both time extents of the varying geometries. Applied to the example of Fig. 1, this means that relationship Located only accepts instances that link a district and an avalanche zone such that their geometries intersect for at least one instant or for every instant belonging to both lifespans. When defining the relationship type, the designer has to specify which interpretation holds.

5 Multi-representation Modeling

Databases store representations of real-world phenomena that are of interest to a given set of applications. However, while the real world is supposed to be unique, its representation depends on the intended purpose. Thus, each application has a peculiar perception of the real world of interest. These perceptions may vary both in terms of what information is to be kept and in terms of how the information is to be represented. Fully coping with such diversity entails that any database element may have several descriptions, each one associated to the perceptions it belongs to. These multiple descriptions are called the *representations* of the element. Both metadata (descriptions of objects, relationships, attributes, is-a links) and data (instances and attribute values) may have multiple representations. There is a bidirectional mapping linking each perception to the representations perceived through this perception.

Classic databases usually store for each real-world entity or link a unique, generic representation, hosting whatever is needed to globally comply with all application perceptions. An exception exists for databases supporting generalization hierarchies, which allow storing several representations of the same entity in increasing levels of specificity. These classic databases have no knowledge of perceptions: Applications have to resort to the view mechanism to define data sets that correspond to their own perception of the database. Instead, MADS explicitly supports multiple perceptions for the same database. A *multi-perception database* is a database allowing users to store one or several representations for each database element, and records for each perception the representations it is made up.

Geographical applications have strong requirements in terms of multiple representations. For example, cartographic applications need to keep multiple geometries for each object, each geometry corresponding to a representation of the extent of the object at a given scale. Multiscale data/representations are needed as there is still no complete set of algorithms for cartographic generalization, i.e. the process to automatically derive a representation at some less detailed resolution from a representation at a more precise resolution.

In MADS, each perception has a user-defined identifier, called its *perception stamp*, or just *stamp*. In the sequel, perception stamps are denoted as s1, s2, … sn. From data definitions (metadata) to data values, anything in a database (object type, relationship type, attribute, role, instance, value) belongs to one or several perceptions. Stamping an element of the schema defines for which perceptions the element is relevant. In the diagrams, e.g. Fig 3, the line identified by the 👁 icon defines the set of perceptions for which this type is valid. Similarly, the specification of the relevant stamps is attached to each attribute and method definition.

There are two complementary techniques to organize multiple representations. One solution is to build a single object type that contains several representations, the knowledge of "which representation belongs to which perception" being provided by the stamps of the properties of the type. Following this approach, in Fig. 3 the designer has defined a single object type RoadSegment, grouping two representations, one for perception s1 and one for perception s2. An object or relationship type is *multi-representation* if at least one of its characteristics has at least two different representations. The characteristic may be at the description level (e.g., an attribute with different definitions) or at the instance level (i.e., different sets of instances or an instance with two different values).

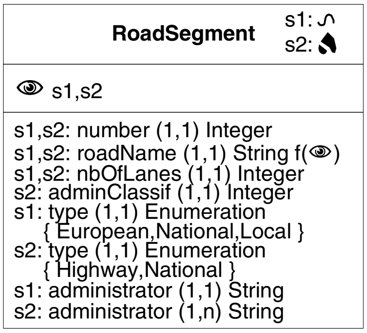
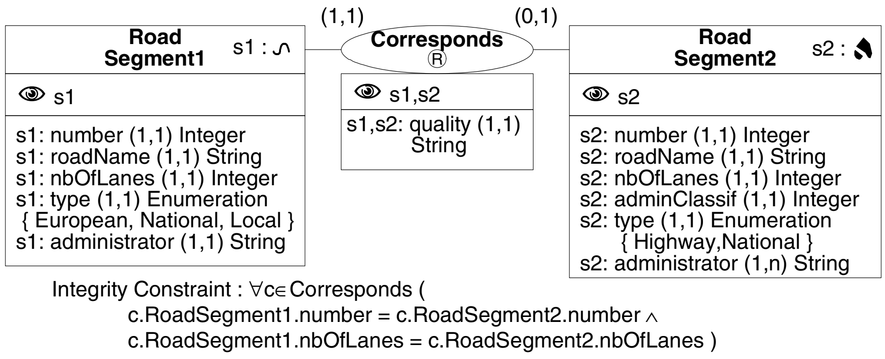

RoadSegment
s1:
s2:
s1,s2
s1,s2: number (1,1) Integer
s1,s2: roadName (1,1) String f(👁)
s1,s2: nbOfLanes (1,1) Integer
s2: adminClassif (1,1) Integer
s1: type (1,1) Enumeration
{ European,National,Local }
s2: type (1,1) Enumeration
{ Highway,National }
s1: administrator (1,1) String
s2: administrator (1,n) String

Fig. 3. An illustration of a bi-representation type, defined for perceptions s1 and s2

The alternative solution to organize multiple representations is to define two separate object types, each one bearing the corresponding stamp(s) (cf. Fig. 4). The knowledge that the two representations describe the same entities is then conveyed by linking the object types with a relationship type that holds a specific *inter-representation* semantics (indicated by the ® icon). In the example Fig. 4, the same real-world road segment is materialized in the database as two object instances, one in RoadSegment1 and one in RoadSegment2. Instances of the relationship type Corresponds tell which object instances represent the same road segment.

The actual representation of instances of multi-representation object types changes from one perception to another. In the object type RoadSegment of Fig. 3 the spatial extent is represented either as a surface (more precise description, perception s2) or as a line (less precise description, perception s1) depending on resolution. Furthermore, perception s1 needs attributes number, roadName, numberOfLanes, type, and administrator. Perception s2 needs attributes number, roadName, numberOfLanes, adminClassification, type, and administrator. The type attribute takes its values from predefined sets of values, the sets being different for s1 and s2. Several administrators for a road segment may be recorded for s2, while s1 records only one. While the road segment number and the number of lanes are the same for s1 and s2, the name of the road is different, although a string in both cases. For instance, the same road may have name "RN85" in perception s1 and name "Route Napoléon" in s2. We call this a *perception-varying attribute* identified by the f(👁) notation. An attribute is perception-varying if its value in an instance may change from one perception to another. A perception-varying attribute is a function whose domain is the set of perceptions of the object (or relationship) type and whose range is the value domain defined for this attribute. These attributes are the counterpart of space-varying and time-varying attributes in the space and time modeling dimensions. Stamps may also be specified at the instance level. This allows defining different subsets of instances that are visible for different perceptions. For example, in the RoadSegment type in Fig. 3 it is possible to define instances that are only visible to s1, instances that are only visible to s2, and instances that are visible to both s1 and s2.



(1,1)
(0,1)
Road Segment1
s1 :
Corresponds
Ⓡ
Road Segment2
s2 :
s1
s1,s2
s2
s1: number (1,1) Integer
s1: roadName (1,1) String
s1: nbOfLanes (1,1) Integer
s1: type (1,1) Enumeration
{ European, National, Local }
s1: administrator (1,1) String
s1,s2: quality (1,1) String
s2: number (1,1) Integer
s2: roadName (1,1) String
s2: nbOfLanes (1,1) Integer
s2: adminClassif (1,1) Integer
s2: type (1,1) Enumeration
{ Highway,National }
s2: administrator (1,n) String
Integrity Constraint : ∀c∈Corresponds (
c.RoadSegment1.number = c.RoadSegment2.number ∧
c.RoadSegment1.nbOfLanes = c.RoadSegment2.nbOfLanes)

Fig. 4. The RoadSegment type (from Fig. 3) split into two mono-representation object types and an inter-representation relationship type

Relationship types are as dependent on perception as object types are. Therefore they can be multi-representation, like object types. Their structure (roles and association/multi-association kind) and semantics (e.g., topology, synchronization) may also have different definitions depending on the perception. For example in Fig. 1 it may be the case that the designer defines the relationship Occurs as 1) a topological and synchronization constraining relationship type for a perception s1, and 2) a plain relationship without any peculiar semantics or constraint for perception s2. A relationship type may have different roles for different perceptions. For example, in Fig. 1 Observes could be perceived in a representation s1 as a binary relationship between an observer and an avalanche event, while a perception s2 sees the same observation as a ternary relationship involving also the surveyor who has validated the observation.

Inter-representation relationship types, like any relationship type, belong to perceptions, and, although this is not likely to frequent happen, they also may have different representations according to the perceptions. In Fig. 4, the inter-representation Corresponds relationship type belongs to perceptions, s1 and s2, and has a unique representation. In particular, its attribute quality, which describes how well the two road segments, one from RoadSegment1 and one from RoadSegment2, correspond to each other[2], is shared by the two perceptions.

6 Conclusions and Future Work

Over the last decades conceptual modeling has been fundamental for improving our understanding about real-world phenomena as well as their implementation into computer systems. However, the globalization of the world as well as the rapidity at which social and economical changes occur calls for richer conceptual modeling approaches that can accumulate a larger variety of data as well as its evolution in time and in space, with detailed knowledge on change processes. Because of the many parties involved in complex data analyses, data representations must be flexible enough to allow recording many perspectives in view of exploring and managing alternative uses while keeping them interrelated so that more global analyses can also be performed. All of these features are advocated as important components of the active conceptual modeling paradigm that aims towards data management of the future.

In this paper we have shown how the MADS conceptual model can be used to capture the spatial and temporal characteristics of real-world phenomena, which are essential for their understanding in a dynamic geo-wide context. Further, we have shown how MADS allows keeping different perspectives of the same phenomenon. This is another essential requirement since complex phenomena must be analyzed under multiple perspectives, e.g., corresponding to economic, social, political, or environmental issues. Data analyses are in particular supported by data warehouses that aim at enabling strategic decision-making. Thus, a natural follow-on on active data modeling is active data warehousing, an emerging new research direction. A MADS companion and ongoing work is exploring a MADS inspired data model for data warehousing of spatial and temporal data [8].

MADS has already been used in many real-world applications, including the natural risk management application sketched in this paper. It thus provided fundamental support for decision making in these applications. Being conceptual it allows designers to focus on the issues at stake without being bothered by implementation constraints when designing and when manipulating the database. Further, specific CASE tools (available at http://cs.ulb.ac.be/mads_tools/) allow the

[2] The instances of the Corresponds relationship type are the result of a spatial matching process that looks for corresponding road segments in RoadSegment1 and RoadSegment2 by comparing their geometries. Roughly, the matching predicate means: Is the geometry of this road segment of RoadSegment1 topologically inside the geometry of that road segment of RoadSegment2?

translation of the data definition and manipulation languages into the languages provided by current implementation platforms.

The clean orthogonality we cared to follow in building the MADS approach is in our opinion what makes the real quality of the data model we propose. Orthogonality is the best way to provide maximum expressive power while keeping maximum simplicity in the constructs of the model. Needless to say, users' understanding of MADS and their ability to learn it and rapidly use it even in complex applications has been our greatest satisfaction. On these same premises, it will be easy to extend MADS to include more conceptual perspectives on additional modeling dimensions, e.g. uncertainty, multimedia, and movement. These are on our research agenda.

References

1. Chen, P.P., Thalheim, B., Wong, L.Y.: Future Directions of Conceptual Modeling. In: Chen, P.P., Akoka, J., Kangassalu, H., Thalheim, B. (eds.) Conceptual Modeling. LNCS, vol. 1565, pp. 287–301. Springer, Heidelberg (1999)
2. Parent, C., Spaccapietra, S., Zimányi, E.: Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach. Springer, Heidelberg (2006)
3. Parent, C., Spaccapietra, S., Zimányi, E.: The MurMur Project: Modeling and Querying Multi-Represented Spatio-Temporal Databases. Information Systems 31(8), 733–769 (2006)
4. Chen, P.P., Wong, L.Y.: A Proposed Preliminary Framework for Conceptual Modeling of Learning from Surprises. In: ICAI 2005. Proceedings of the International Conference on Artificial Intelligence, pp. 905–910. CSREA Press (2005)
5. Khatri, V., Ram, S., Snodgrass, R.: Augmenting a Conceptual Model with Geospatiotemporal Annotations. IEEE Transctions on Knowledge and Data Engineering 16, 1324–1338 (2004)
6. Pelekis, N., Theodooulidis, B., Kopanakis, I., Theodoridis, Y.: Literature review of spatio-temporal database models. The Knowledge Engineering Review 19, 235–274 (2004)
7. Bédard, Y., Bernier, E.: Supporting Multiple Representations with Spatial Databases Views Management and the Concept of VUEL. In: Proceedings of the Joint Workshop on Multi-Scale Representations of Spatial Data, ISPRS WG IV/3, ICA Comm. on Map Generalization (2002)
8. Malinowski, E., Zimányi, E.: Designing Conventional, Spatial, and Temporal Data Warehouses: Concepts and Methodological Framework. Springer, Heidelberg (to appear, 2007)