Augmenting Database Design-Support Environments to Capture the Spatio-Temporal Data Semantics

Vijay Khatri, Sudha Ram and Richard T. Snodgrass

August 27, 2003

TR-73

A TIMECENTER Technical Report

Title	Augmenting Database Design-Support Environments to Capture the Spatio-Temporal Data Semantics
Author(s)	Vijay Khatri, Sudha Ram and Richard T. Snodgrass
Publication History	August 2003. A TIMECENTER Technical Report.

TIMECENTER Participants

Aalborg University, Denmark
Christian S. Jensen (codirector), Michael H. Böhlen, Heidi Gregersen, Simonas Šaltenis, Janne Skyt, Giedrius Slivinskas, Kristian Torp

University of Arizona, USA
Richard T. Snodgrass (codirector), Dengfeng Gao, Bongki Moon, Sudha Ram

Individual participants
Curtis E. Dyreson, Washington State University, USA; Fabio Grandi, University of Bologna, Italy; Vijay Khatri, Indiana University, USA; Nick Kline, Microsoft, USA; Gerhard Knolmayer, University of Bern, Switzerland; Thomas Myrach, University of Bern, Switzerland; Kwang W. Nam, Chungbuk National University, Korea; Mario A. Nascimento, University of Alberta, Canada; John F. Roddick, Flinders University, Australia; Keun H. Ryu, Chungbuk National University, Korea; Dennis Shasha, New York University, USA; Michael D. Soo, amazon.com, USA; Andreas Steiner, TimeConsult, Switzerland; Paolo Terenziani, University of Torino; Vassilis Tsotras, University of California, Riverside, USA; Jef Wijsen, University of Mons-Hainaut, Belgium; and Carlo Zaniolo, University of California, Los Angeles, USA

For additional information, see The TIMECENTER Homepage:
URL: `<http://www.cs.auc.dk/TimeCenter>`

The TIMECENTER icon on the cover combines two "arrows." These "arrows" are letters in the so-called *Rune* alphabet used one millennium ago by the Vikings, as well as by their precedessors and successors. The Rune alphabet (second phase) has 16 letters, all of which have angular shapes and lack horizontal lines because the primary storage medium was wood. Runes may also be found on jewelry, tools, and weapons and were perceived by many as having magic, hidden powers.

The two Rune arrows in the icon denote "T" and "C," respectively.

Abstract: Many real world applications need to organize data based on time (e.g., accounting, portfolio management, personnel management, inventory management) and/or space (e.g., facility management, market analysis, transportation, logistics). Underlying these applications is temporal and/or spatial data, referred to as spatio-temporal data. Conventional conceptual models provide a mechanism to elicit data semantics related to "what" is important for an application rather than the "when" and "where" semantics. For spatio-temporal applications listed above, it is left to the database designers to discover, design and implement—on an ad-hoc basis—the temporal and spatial concepts that they need. We describe a database design-support environment that exemplifies our spatio-temporal conceptual design approach: (i) first capture current reality using a conventional conceptual model without considering the temporal or spatial aspects (i.e., "what"); and only then (ii) annotate a conventional schema with spatio-temporal data semantics (i.e., "when/where"). We show how segregating "what" from "when/where" via annotations is straightforward to implement, satisfies ontology- and cognition-based requirements, dovetails with existing database design methodologies, and can support seamless integration of schemas across diverse design-support environments having different syntax but the same underlying semantics. We demonstrate how such an overall design-support environment that supports elicitation of the spatio-temporal semantics can help bridge the semantic gap between the real world and its spatio-temporal representation in the information systems.

Keywords: Semantic Model, Data Semantics, Database Design, Spatio-Temporal Database, CASE Tool

1 Introduction

A design-support environment—a tool that automates many of the analyst's tasks in developing software—helps reduce time and money spent on a project and can improve the quality of the end-product [21]. Additionally, such environments improve the effectiveness of software development, an important issue facing information systems managers [10]. However, extant design-support environments do not adequately support applications that need to organize data based on time (e.g., accounting, portfolio management, personnel management) and/or space (e.g., facility management, transportation, logistics). *Conventional conceptual models*, e.g., [11, 16], provide a mechanism to elicit data semantics related to "what" is important for an application rather than the "when" and "where" semantics. For spatio-temporal applications listed above, it is left to the database designers to discover, design and implement—on an ad-hoc basis—the temporal and spatial concepts that they need to represent the "miniworld," or an aspect of the real world [16]. To capture the data semantics related to space and/or time—at an abstract-level independent of the physical data model—there is a need for a design-support environment that is based on a *spatio-temporal conceptual model*.

We describe a design-support environment that exemplifies our spatio-temporal conceptual design approach that advocates: (i) first capturing *current reality* using a conventional conceptual model *without considering spatial or temporal aspects* (i.e., "what"); and only then (ii) annotating a conventional schema to capture spatio-temporal requirements (i.e., "when/where"). This approach is embodied in a prototype, a DesIgn-Support environment for SpaTIotemporaL data, or *DISTIL*. To date there has been little work done in characterizing the impact of adding spatio-temporal semantics to an existing database design-support environment. We demonstrate how augmenting an extant database design-support

environment is straightforward to implement, satisfies ontology- and cognition-based requirements, dovetails with existing database design methodologies, and can support seamless integration of schemas across diverse design-support environments having different syntax but the same underlying semantics.

Todd et al. [67] define CASE (Computer-Aided Software/System Engineering) tool as "the automation of part of, or the entire, system development process." Various studies [4, 20, 43] have concluded that CASE tools improve the development process. However, there does not exist a CASE tool that can be employed for designing and developing spatio-temporal database applications. Research interest in spatio-temporal data has increased dramatically over the past decade, as is evident from published bibliographies [2, 35, 38, 63, 64, 69, 78]. From a practitioner's perspective, recent advances in technologies like high-resolution satellite-borne imaging systems, mobile systems, global positioning systems and the overall decrease in hardware costs is resulting in temporal and spatial data finding its way into many traditional applications.

Via DISTIL, we exemplify our spatio-temporal design methodology that includes: (i) support for development of a *conventional conceptual schema* using the Unifying Semantic Model (USM) [49], an extended Entity-Relationship (ER) Model [11]; (ii) means for associating the conventional schema with spatio-temporal annotations, thus, helping encapsulate spatio-temporal semantics via the *ST-USM* (Spatio-Temporal Unifying Semantic Model) *schema* [33, 34]; (iii) assistance in validating consistency of the captured spatio-temporal semantics; (iv) a mechanism for explicating the encapsulated spatio-temporal semantics via the *translated USM schema*; (v) support for conversion of a translated USM schema to a logical schema; (vi) provision for translation of the annotated conceptual schema to an XML document, thus, facilitating sharing of conceptual schemas developed with, possibly, diverse design tools having different syntax but same underlying semantics.

Our work makes several contributions to spatio-temporal conceptual design. (i) One of the problems with developing spatio-temporal applications is that there is "a gulf between the richness of knowledge structures in the application domains and the relative simplicity of the data model in which the structures can be expressed" [77]. We demonstrate how spatio-temporal conceptual modeling can be augmented and realized into an existing design-support environment. (ii) The challenge in adding the space and time dimension is balancing simplicity and understandability with preciseness and completeness. The annotations—based on spatio-temporal ontology—are precisely defined in Backus-Naur Form (BNF) (cf. Appendix) and first-order logic [33]. At the same time, our approach supports multiple levels of abstraction and helps elicit spatio-temporal requirements in a manner that is consistent with human cognition. (iii) Our approach augments existing database design methodology and is *upward compatible* [62], i.e., it does not invalidate any legacy schemas. (iv) Our proposed methodology is straightforward to implement. We show how adding spatio-temporal annotations requires few changes to existing code and to the structure of the database that underlies the design-support environment. (v) Various types of

abstractions supported by typical conventional conceptual models, e.g., [5, 11, 16, 49, 55], include *classification*, *association*, *generalization*/*specialization* and *aggregation*. While different modeling formalisms provide different syntax, they share similar underlying data semantics. We show how conversion of a conceptual schema to an XML document can support exchange and sharing of conceptual schemas "across" design-support environments that are based on abstractions like classification, association, generalization/specialization and aggregation.

The rest of the paper is organized as follows. We first motivate the need for spatio-temporal conceptual design using a study at the United States Geological Survey (USGS). According to Wand et al. [72], "the power of a modeling language lies in the semantics of its constructs" and "ontology can be used to define concepts that should be represented by the modeling language." The basis for dialog panels in our proposed design-support environment is a time and space ontology, which is summarized in Section 3. In Section 4 we describe a database analyst's interaction with DISTIL using the USGS example of Section 2. Next, we provide an architecture for a spatio-temporal database design-support environment in Section 5. In Section 6, we evaluate the DISTIL prototype and show how augmenting an existing design-support environment with spatio-temporal annotations is straightforward to implement. We compare our work with that of other design-support environments in Section 7, and conclude in Section 8.

2 Motivation

Woods [74] defines data semantics as the meaning and use of data. In the information systems context, data semantics [54] refers to a set of mappings from a representation language to agreed-upon concepts in the real world. Data semantics provide "a connection from a database to the real world outside the database" [54] and a *conceptual model* provides a mechanism to capture the data semantics. Using an example of an application at USGS, we demonstrate the need to capture spatio-temporal data semantics.

Earth-science studies, e.g., landslide hazards, earthquakes, volcanoes, subsurface tunneling and construction, environmental contamination, water availability, provide the knowledge required for society to make decisions related to managing natural resources in a sustainable manner. While many advances in the understanding of earth-science processes and applications are being used by researchers, technologies for storing, sharing and managing the spatio-temporal data are far less developed. As an example, in addressing concerns related to water quality, water-availability and contamination, large amounts of multidisciplinary data must be acquired, synthesized, analyzed and disseminated. Capturing the space and time semantics of the vast amounts of interdisciplinary data is the core of this infrastructure.

We are working with a group of researchers who are developing a *ground-water flow model* [12] for the Death Valley region. Beneath the earth's surface, the zone where all interstices are filled with water is

referred to as *ground water*. In arid regions like Death Valley, which encompasses approximately 80,000 km^2 in Nevada and California, ground water provides a large percentage of water for domestic, industrial and agricultural uses. The objective of the ground-water flow model for the Death Valley Region is to characterize regional 3D ground-water flow paths so that policy makers can make decisions related to radio-nuclide contaminant transport and the impact of ground-water pumping on national parks and local communities in the region. However, the quality of model outputs and predictions based on the model are dependent on the data that forms an input to the model. We describe a subset of the input data required for the ground-water flow model.

Two key objects of interest for the ground-water flow model are *spring-water sites* and *borehole sites*. Both of these need to be spatially referenced to the Earth and are uniquely identified by a *site id*. A spring-water site is a point on the surface of the Earth given by geographic *x*- and *y*- coordinates, with a spatial granularity of dms-second. Geographically spring-water sites exist within a *spring* (represented as a region) and there can be many spring-water sites within a spring. A spring usually has a *name* by which it is known locally. An important characteristic of a spring is the *permanence* of discharge at the spring, e.g., `perennial springs` discharge continuously and `intermittent springs` are periodically dry. A borehole site refers to a part of the borehole whose 3D location is given by *x*- and *y*- coordinates on the Earth's surface, with a spatial granularity of dms-second, and depth below the surface with a spatial granularity of foot. While there can be one or more borehole sites at different depths within a drilled hole at the same surface location, each borehole site is associated with exactly one borehole. A borehole site is characterized by tests like *horizontal* (*hydraulic*) *conductivity* and *diffusivity*, and the values for these tests are valid for the minute at which the test was conducted. Borehole sites and spring-water sites also have an associated *status*, e.g., `site was dry and no water level was recorded` and `site was recently pumped`. Spring-water sites also have an associated *description* that characterizes and defines the site. The measurements at the borehole site and spring site are taken by a *source agency*, and these measurements embody the ground-water flow model.

A borehole site may have a pumplift that removes water from the borehole site and this can affect other data collected at the borehole site. Some of the characteristics of a pumplift are *type* (e.g., `air lift`, `rotary pump`, `jet pump`), *manufacturer* and *serial number*. A pumplift has a lifespan that specifies the time periods when the pumplift was installed and was operational. The time periods of pumplift existence denote the times when data collected at a borehole site can be influenced by a given pumplift.

As exemplified by the spatio-temporal application described above, we summarize the requirements for a design-support environment as follows: (i) Support for conventional conceptual modeling, e.g., entity class (e.g., SOURCE_AGENCY), attribute (e.g., agency_code and tech_name) and relationship (i.e., sp_measure between SPRING_SITE and SOURCE_AGENCY). (ii) Ability to capture semantics associated

with *valid time* [59], i.e., when a given fact was true in the real-world. Some facts, e.g., diffusivity and horizontal conductivity, need to be represented as *events*. An event occurs at a point of time, i.e., an event has no duration. On the other hand, the lifespan of pumplift needs to be represented as *state*. A state has duration, e.g., pumplift existed from 04/12/1999 to 07/27/2001. (iii) All the temporal data need to be associated with temporal granularity. For example, the lifespan of a pumplift needs to be captured to the granularity of day while that for diffusivity and horizontal conductivity needs to be captured to the granularity of minute. (iv) The spatial objects are characterized by *shape* and *position* [13]. For example, spring site needs to be represented as a point on the *x-y* plane and spring needs to be represented as a region on the *x-y* plane. (v) All spatial data needs to associated with spatial granularity, e.g., dms-sec for SPRING_SITE (vi) Augmenting a design-support environment should also take into consideration the extant (non- temporal/non-spatial) CASE tools. Considering the large number of extant design-support environments, any spatio-temporal extension should not require extensive changes to an existing tool. (vii) Jarzabek and Huang [28] argue that current design tools "are weak in addressing *soft* aspects of software development." Thus, the CASE tool interface design should also take into consideration how we perceive and process spatio-temporal data. (viii) The recent revolution in network interconnectivity and the World-Wide Web (WWW) presents a unique opportunity for distributed teams to employ an integrated virtual environment for designing information systems. The design tool should support translation of a conceptual schema to an XML document, thus, enabling integration of schemas developed by different design-support environments.

Having summarized the requirements for a design-support environment that supports spatio-temporal conceptual modeling, we next formalize the temporal and spatial data semantics that can be elicited with a design-support environment, e.g., DISTIL, that facilitates spatio-temporal conceptual modeling.

3 Needed Ontological Concepts

Ontology is the specification of the representational vocabulary for a shared domain of discourse [25], and space and time ontology is the basis for the dialog panel in the design-support environment and the associated annotations. We first formalize ontological concepts related to temporal [1, 7, 8, 14, 15, 29, 58-61] and spatial [13, 22, 50, 75, 76] data that are embedded in DISTIL. One of the deficiencies of the existing conceptual models that can represent geographic phenomena is their inability to "represent information in way that is more natural to humans" [39, 47]. We next describe how DISTIL takes into consideration cognition related to spatio-temporal data.

3.1 Temporal Ontology

The basis of time ontology is the definition of the time domain. We review extant definitions associated with how facts can interact with time [59-61]. Intrinsic to temporal data is temporal granularity. We summarize existing definitions associated with temporal granularity [7, 8, 14].

A *time domain* is denoted by the pair $(T, \leq)$, where T is a nonempty set of *time instants* and "≤" is the total order on T. We can assume the time domain is either *discrete* or *dense*. While there is no general agreement if time domain is dense or discrete, the temporal database community agrees that a discrete model of time is generally adequate for representing reality [31]. Additionally, time is assumed to be bounded at both ends, i.e., the past and the future [57]. An *instant* is a time point on the time line. For example, $(\mathbf{Z}, \leq)$ represents a discrete time domain where instants are isomorphic to integers, implying that every instant has a unique successor.

Facts can interact with time in two orthogonal ways resulting in *transaction time* and *valid time* [59]. Valid time denotes when the fact is true in the real world and implies the storage of histories related to facts. On the other hand, transaction time links an object to the time it is current in the database and implies the storage of versions of a database object. While the temporal granularity can be specified for valid time, that for transaction time is system-defined. The transaction time has duration from insertion to (logical) deletion [32] and can include granules only up to the current time granule in the real world. Time-varying data may be modeled as an *event* or a *state* [30]. An event occurs at a point of time, i.e., an event has no duration. A state has duration, e.g., a storm occurred from 5:07 PM to 5:46 PM.

Temporal granularity is a measure of the time datum. A *temporal granularity* is defined as a mapping *TG* from *index i* to subsets of the time domain such that: (i) granules $TG(i)$ in a temporal granularity do not overlap; (ii) the index order of a temporal granularity corresponds with the time domain order; (iii) the index set of a temporal granularity provides a contiguous granule encoding; and (iv) a special granule called the *origin*, $TG(0)$ is non-empty. Although the index of a temporal granularity is constrained to be contiguous, the granules are not constrained to be contiguous on the time domain. Thus, a temporal granularity defines countable set of non-decomposable granules that can be composed of a set of contiguous instants or non-contiguous instants. Some examples of temporal granularities are Gregorian day, business day and business week. While Gregorian day is a temporal granularity with contiguous granules of hour, business day is not. Each non-empty granule may have a textual representation termed a *label* (e.g., "November 25, 2000"), which can be mapped to the index integer with a mapping called the *label mapping*. The earliest time domain element in the origin is referred to as an *anchor* with respect to the time domain. The union of time granules is called an *image* of a temporal granularity. The smallest interval of the time domain that contains the image of a temporal granularity is called the *extent* of that granularity. The image of a temporal granularity can be contiguous or have holes in it. Gregorian day and

business day are granularities with discrete image of days. However, Gregorian day has contiguous granules of hour while business day includes non-contiguous granules of hour.

3.2 Spatial Ontology

Any data that can be associated with location on the Earth are referred to as geographic data [13]. Geographic space based on Euclidean geometry is the basis for most GISs [37]. In this case, the location can be expressed by a set of coordinates, e.g., latitude and longitude. We briefly review concepts related to space and spatial granularity.

The space domain may be represented as a set (e.g., $\mathbf{R}^3$, $\mathbf{R}^2$, $\mathbf{N}^3$, $\mathbf{N}^2$) with elements referred to as *points*. A spatial object is associated with *geometry* and *position*. Geometry represents the shape and size of an object [13]. The position in space is based on coordinates in a mathematically-defined reference system, e.g., latitude and longitude. Geometry of the spatial object may be 0-, 1- or 2- dimensional corresponding to a *point*, a *line* or a *region*. A point is "a zero-dimensional spatial object with coordinates," a line is "a sequence of ordered points, where the beginning of the line may have a special start node and the end a special end node" and a region or polygon consists of "one outer and zero or more inner rings" [70]. David et al. [13] define an area as "a bounded continuous two-dimensional geometric primitive delimited by one outer non-intersecting boundary and zero or more nested non-intersecting inner boundaries." They differentiate between a line and a region: the line itself is "the carrier" of the information while in a region, the area is of primary importance and the "boundary is secondary…to limit the area."

For geographic applications, horizontal space is segregated from vertical space. Correspondingly, we define *horizontal* and *vertical spatial granularities* [34]. Intuitively, the horizontal space domain corresponds to the Earth's surface while vertical space domain corresponds to the depth/height below/above the sea level. We define horizontal spatial granularity as a mapping from integers to any partition of horizontal space, where the partition may arise from pixellation of space and may be a regular square or any other shape such as a triangular irregular network (TIN) or even an irregular shape (e.g., county). Formally, a horizontal spatial granularity may be defined as a mapping SG_{xy} from index i to a subset of space domain such that: (i) granules from a spatial granularity do not overlap; (ii) the index set of a spatial granularity provides a contiguous encoding, though the granules in the space domain are not constrained to be contiguous in the underlying spatial domain; and (iii) origin granule $SG_{xy}(0)$ is nonempty. Examples of horizontal spatial granularities are dms-deg, dms-min and county. Each non-empty granule can have a textual representation called *label*, which can be mapped to the index integer by a mapping function called *label mapping*. For example, `45°23′E/24°35′N` is an example of a label that represents a point in space whose granularity is dms-min for both latitude/longitude. For granularities like

dms-deg, space is partitioned along two perpendicular directions and the granularity is construed to be dms-deg along the two dimensions. On the other hand, county is an example of an irregular horizontal spatial granularity.

3.3 Time-Varying Spatial Ontology

In geography, space is indivisibly coupled with time [46]. Lately, there has been much interest in adding a temporal aspect to geographic databases [23] since time integrates human activity, orders events and separates cause from effect [71]. Three types of interaction between an object and space-time are possible [48, 68]: (i) moving objects, i.e., objects whose position changes continuously but whose shape does not (e.g., a car moving on a road network); (ii) objects whose spatial characteristics and position change with time discretely, i.e., changing shape (e.g., change of the shape of land parcels in a cadastral application); and, (iii) integration of the above two behaviors, i.e., continuous moving and changing phenomena (e.g., modeling a storm). While a point and a line may "*move* over time", a region can change its location (i.e., move) and change its shape.

3.4 Cognition of Spatio-Temporal Data

Since a conceptual schema acts as a communication mechanism among users, database analysts and the database implementation, the formalism provided by a conceptual model for developing a schema should be comprehensible and straightforward to use. Therefore, it is imperative to propose a spatio-temporal modeling approach that takes into account cognition, as this would ensure that the proposed schema is comprehensible and straightforward to use.

Prior research [3, 51] posits that all human knowledge is stored as abstract conceptual propositions. The *propositions* are assertions about the real world.

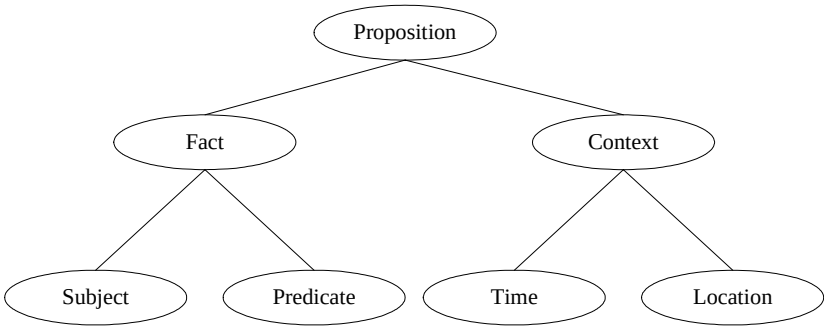

Figure 1: Human Associative Memory Model [3]

As shown in Figure 1, Anderson and Bower's [3] Human Associative Model (HAM) is based on propositions, which are composed of *facts* and *context* associated with the facts. The *subject* and *predicate* correspond with a topic and a comment about the topic, and this corresponds to the

representation as object-property or property-value pairs. For some applications, the context in which the fact is true can be the key to reasoning about the mini-world. This context in turn is composed of *time* and *location* associated with the fact. Our spatio-temporal conceptual design approach focuses first on facts (*what*) and then on context (*where* and *when*) related to those facts corresponds with Anderson and Bower's Human Associative Memory Model [3]. Thus, we expect that such an approach will lead to schemas that are comprehensible and straightforward to use.

4 Spatio-Temporal Conceptual Design

As shown in Figure 2, our spatio-temporal conceptual modeling approach involves first developing a non-spatio-temporal conceptual schema referred to as a *core conceptual schema* (cf. Section 4.1), augmenting the core schema with spatio-temporal annotations leading to an *annotated conceptual schema* (cf. 4.2), validating the consistency of the annotated schema (cf. 4.3), explicating the semantics of the annotated schema (cf. 4.4) resulting in the *translated conceptual schema* and translating the conceptual schema to a *logical schema* (cf. 4.5).

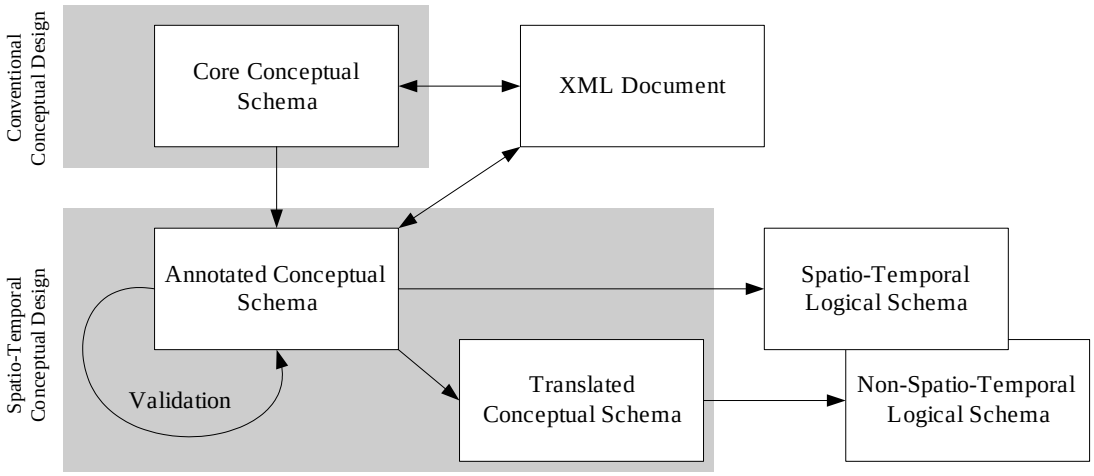

Figure 2: Overview of our Spatio-Temporal Conceptual Design Approach

A semantically richer spatio-temporal logical schema innately "understands" the spatio-temporal concepts that are encapsulated via annotations during spatio-temporal conceptual design. As a result, the encoded semantics that is "wrapped" in the annotated conceptual schema and is kept "wrapped" in the spatio-temporal logical schema. On the other hand, a non-spatio-temporal logical schema needs translation from a *translated conceptual schema*, where the spatio-temporal semantics need to be explicated. The translation of the core conceptual schema (i.e., non-temporal and non-spatial) or annotated schemas (i.e., spatio-temporal) to an XML document helps facilitate integration of schemas developed by different design-support environments (cf. 4.6).

4.1 Developing a Core Schema

We first summarize key terms and terminology related to USM [49], which is an extended version of the ER Model [11]. We next describe how our approach supports conceptual modeling using the formalism provided by USM.

We chose USM as the base model for capturing "what" are the data semantics that are important for an application as it provides an overall framework that carefully defines entity class (i.e., classification) and various types of relationships like interaction (i.e., aggregation), generalization/specialization, and *composite* and *grouping* relationships (i.e., association). Additionally, USM provides subtle semantics that segregate groupings from composites [5, 11, 16, 49, 55]. Note that USM is consistent with conventional conceptual models that include abstractions like classification, aggregation, generalization/specialization and association.

All real world objects are referred to by the term *entity*. Characteristics or properties of entities are called *attributes* (A_i, where $i = 1,\ldots, n$). Each attribute has associated with it an *attribute domain* ($dom(A_i)$) or simply *domain*, which is the set of values that an entity can take for the attribute. An *entity class* (or *class*) may be defined as $E = \cup_i (A_i, dom(A_i))$. The set of instantiations of an entity class is referred to as an *entity set*. In other words, an entity e of an entity class E may be designated as $e(E)$ and a set of entities of an entity class is represented by $S(E)$ where $e(E) \in S(E)$.

Attributes are created by *property relationships*. Associations between or among members of entity classes are called *class relationships*. If an attribute is created by a property relationship, its values are drawn from an *attribute domain*. An attribute created by a class relationship refers to members in some other entity class. In this case, the domain of the attribute is a set of entities belonging to another entity class. USM distinguishes between *simple* and *constructed* entity classes. In simple entity classes all attributes are created by property relationships, whereas in constructed classes there are one or more attributes that are created by class relationships. Constructed classes are used to model entities built from other entities of the database application. We briefly describe various class relationships: *interaction*, *generalization/specialization*, *composite* and *grouping*.

An interaction relationship refers members of one entity class to members of one or more entity classes. Formally, let R be an interaction relationship and $E_1, E_2,\ldots, E_n$ be classes that participate in the relationship. An interaction relationship set, $S(R)$, may be considered to be a subset of the Cartesian product $S(E_1) \times S(E_2) \times\ldots\times S(E_n)$. A relationship instance, $r(R)$, consists of exactly one entity from each participating entity set. Generalization is a form of abstraction in which similar objects are related to a higher-level generic object and the constituent objects may be considered as the specialization of the generic object [8]. A generalization proceeds from the recognition that a number of entity classes share some common features. The crucial property of higher- and lower-level entities created by specialization

and generalization is *attribute inheritance*, i.e., the attributes of higher-level entity classes are said to be *inherited* by the lower-level entity classes [55].

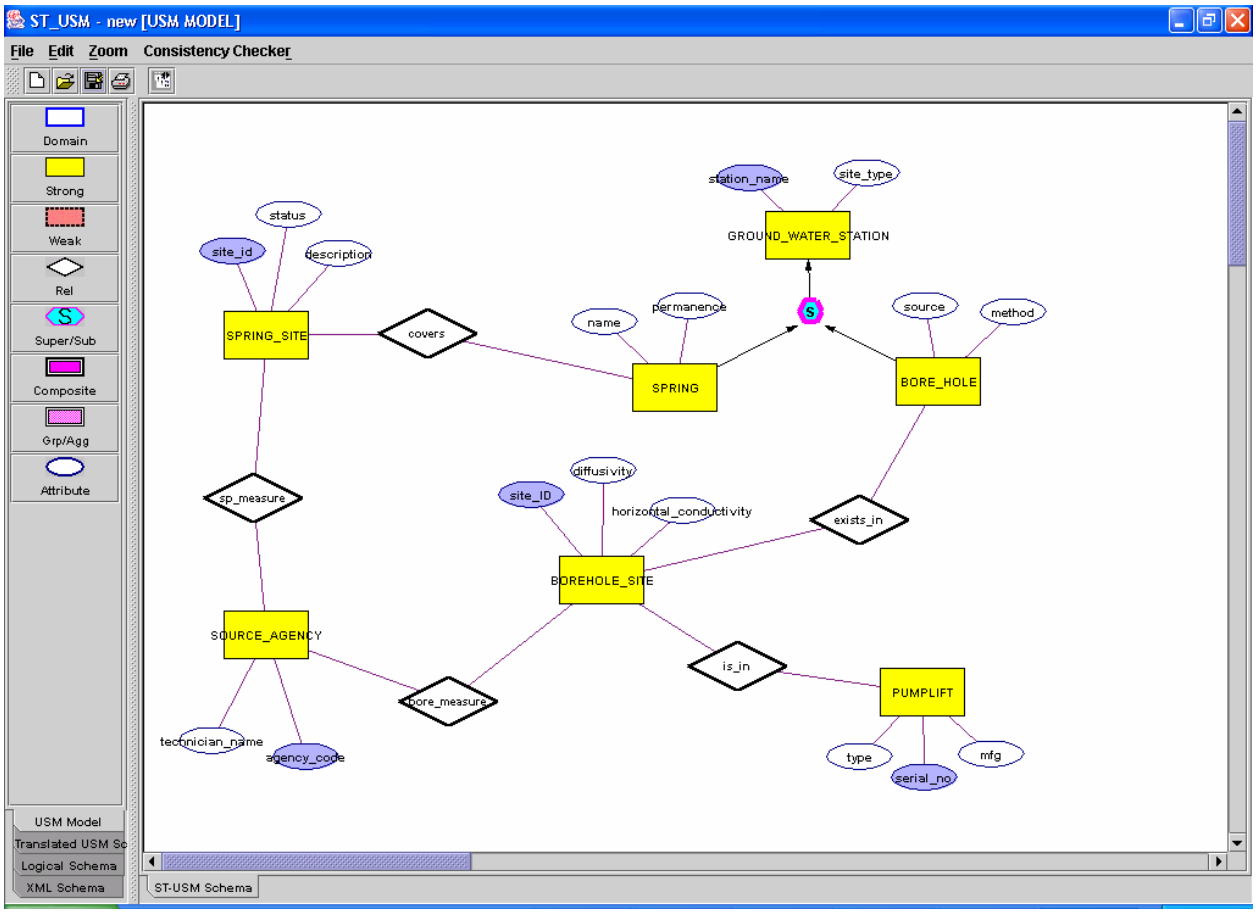

Figure 3: USM Schema

A composite relationship defines a new class called the *composite class* that has another entity set (or subsets of an entity set) as its members. A composite relationship is similar to the "power set grouping" in [52], in that they both represent a set whose members are subsets of the base entity set, $S(E)$. Each member from a composite class—referred to as a *composite*—is a subclass from some other class called the *base class*. Note that the base class is both a *subclass* and a *subtype* of a composite class. A grouping relationship defines a new class, called a *grouping class*, whose members are physically or logically made up of members or sets of members from some other entity class(es), called *component classes*. The grouping establishes a "part-of" or "property-of" relationship.

Having briefly overviewed the USM, we next show how to elicit semantics of "what" is important for the USGS application described in Section 2. Figure 3 shows an example of a core USM Schema developed by the database analyst. This schema includes entity classes like SPRING_SITE,

BORE_HOLE_SITE, SOURCE_AGENCY, PUMPLIFT, SPRING, BORE_HOLE, GROUND_WATER_STATION and their various attributes.

Entity classes are created by using the constructs on the "USM Model" panel on the left side (specifically, the "Strong" rectangles). For example, the relationship sp_measures (the diamond "Rel" on the USM Model side panel) between SPRING_SITE and SOURCE_AGENCY relates an entity of SPRING_SITE with that of SOURCE_AGENCY. Each entity class has associated attributes, e.g., PUMPLIFT has properties like type and manufacturer (mfg) and a key attribute serial_no. SPRING and BORE_HOLE have certain common properties that can be abstracted as GROUND_WATER_STATION. Properties such as station_name and site_type are common to both SPRING and BORE_HOLE. On the other hand, the attributes like permanence and name are specific to SPRING, and source and method are specific to BORE_HOLE.

Developing a schema like the one shown in Figure 3, which includes abstractions like entity class, attributes, relationships, superclass/subclass, grouping and composite, is supported by typical design-support environments. Note that developing such a schema—even without space and time aspects—is an involved task and the resulting non-temporal and non-spatial schema can contain tens of entity types and hundreds of attributes. For example, a small fragment of the (non-temporal and non-spatial) schema for the USGS application includes 18 entity classes and 92 attributes [33].

We next show how a design-support environment that supports conventional conceptual modeling can be augmented with spatio-temporal annotations.

4.2 Annotating the Core Schema

Via annotations, we exemplify a supplementary level of abstraction that "naturally" extends the semantics of a conventional conceptual model to capture the spatio-temporal data semantics. The conceptual model that captures spatio-temporal semantics via annotations is referred to as ST-USM. We describe the syntax related to annotations and then show how DISTIL—based on the analysts' inputs—automatically creates annotation phrases within the schema. Note that these annotation phrases succinctly encapsulate the spatio-temporal aspects of the application.

As shown in the Appendix, the overall structure of an *annotation phrase* has the following components.

⟨temporal annotation⟩ // ⟨geospatial annotation⟩ // ⟨time-varying spatial annotation⟩

The temporal annotations, spatial annotations and time-varying spatial annotations are each separated by a double forward slash (//).

The temporal annotation first specifies the existence time (or valid time) followed by the transaction time. The temporal annotation for existence time and transaction time is segregated by a forward slash (/).

Any of these aspects can be specified as not being relevant to the associated conceptual construct by using "-". The valid time or existence time can be modeled as an event (E) or a state (S) and has an associated temporal granularity. For example, "S(min)/T//" associated with PUMPLIFT would denote that PUMPLIFT exists in a bitemporal space; the temporal granularity of the states (S) is minute (min). Additionally, we also need to capture transaction time (T) associated with PUMPLIFT. In this example, the granularity associated with transaction time is not specified as it is system-defined.

The temporal annotation described above can be specified using a dialog panel (e.g., Figure 4). In the pop-up box, the database analyst can specify if the application needs organize data based on time.

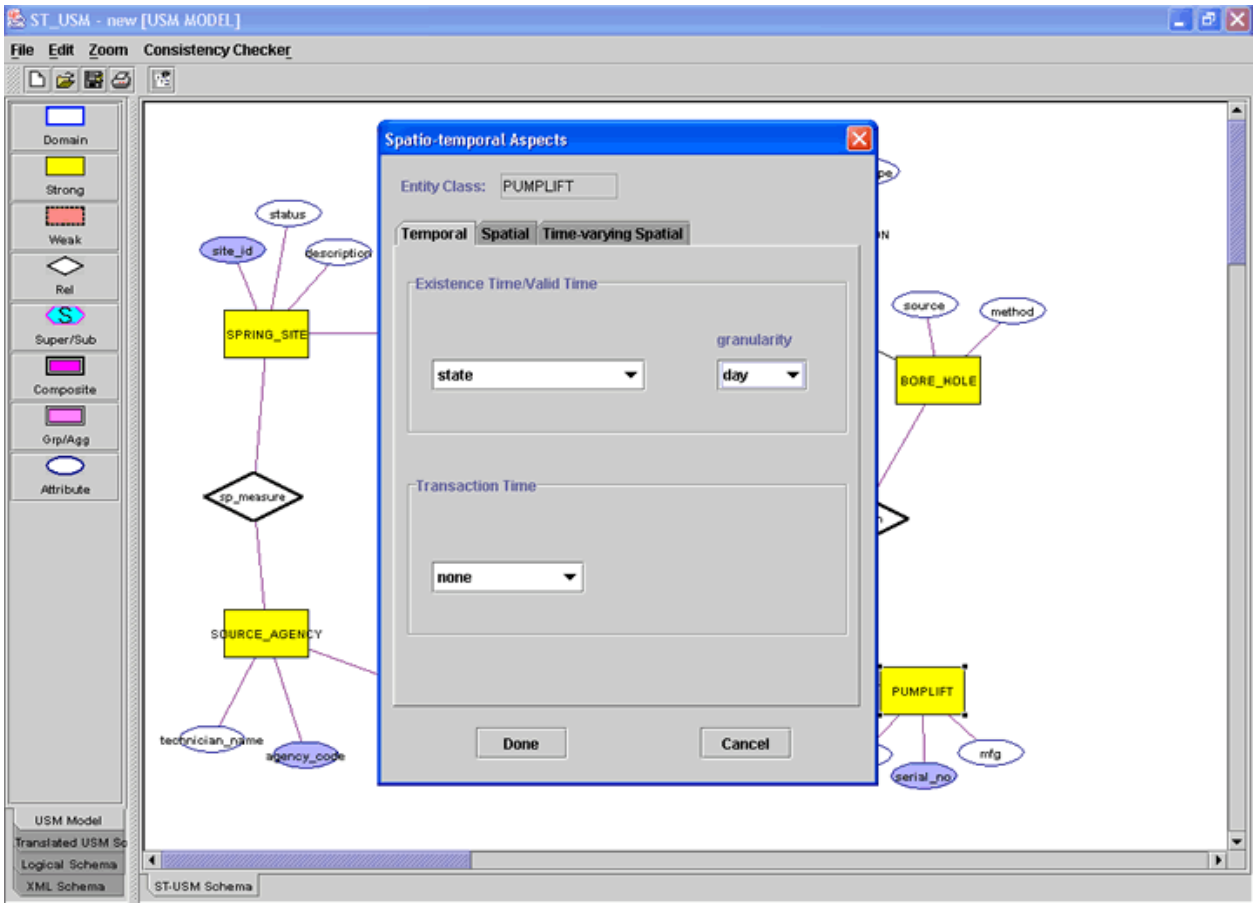

Figure 4: Specifying Temporal Aspects

Accordingly, valid time and/or transaction time may be pertinent for an application. The valid time may be represented as an event or state and has an associated temporal granularity. On the other hand, granularity associated with transaction time does not need to be specified as it is system-defined. For example, Figure 4 shows how the database analyst can enter temporal details which would result in an annotation phrase "S(day)/-//" for PUMPLIFT. This annotation succinctly describes that the lifespan of a pumplift need to be represented as a state (S) with temporal granularity of day.

The spatial annotation includes geometry and position in *x*-, *y*- and *z*-dimension; each dimension is segregated by a forward slash (/). For example, "// P(dms-sec) / P(dms-sec) / -" for SPRING_SITE describes a geometry of points (P) in the *x*-*y* plane. The associated horizontal spatial granularity is dms-sec.

Figure 5 shows how the database analyst can enter spatial details that result in an annotation phrase "//P(dms-sec)/P(dms-sec)/-" for SPRING_SITE. This implies that the SPRING_SITE needs to be represented as a point ("P") on the *x*-*y* plane. Additionally, the associated granularity in the *x*-*y* plane is degree (deg).

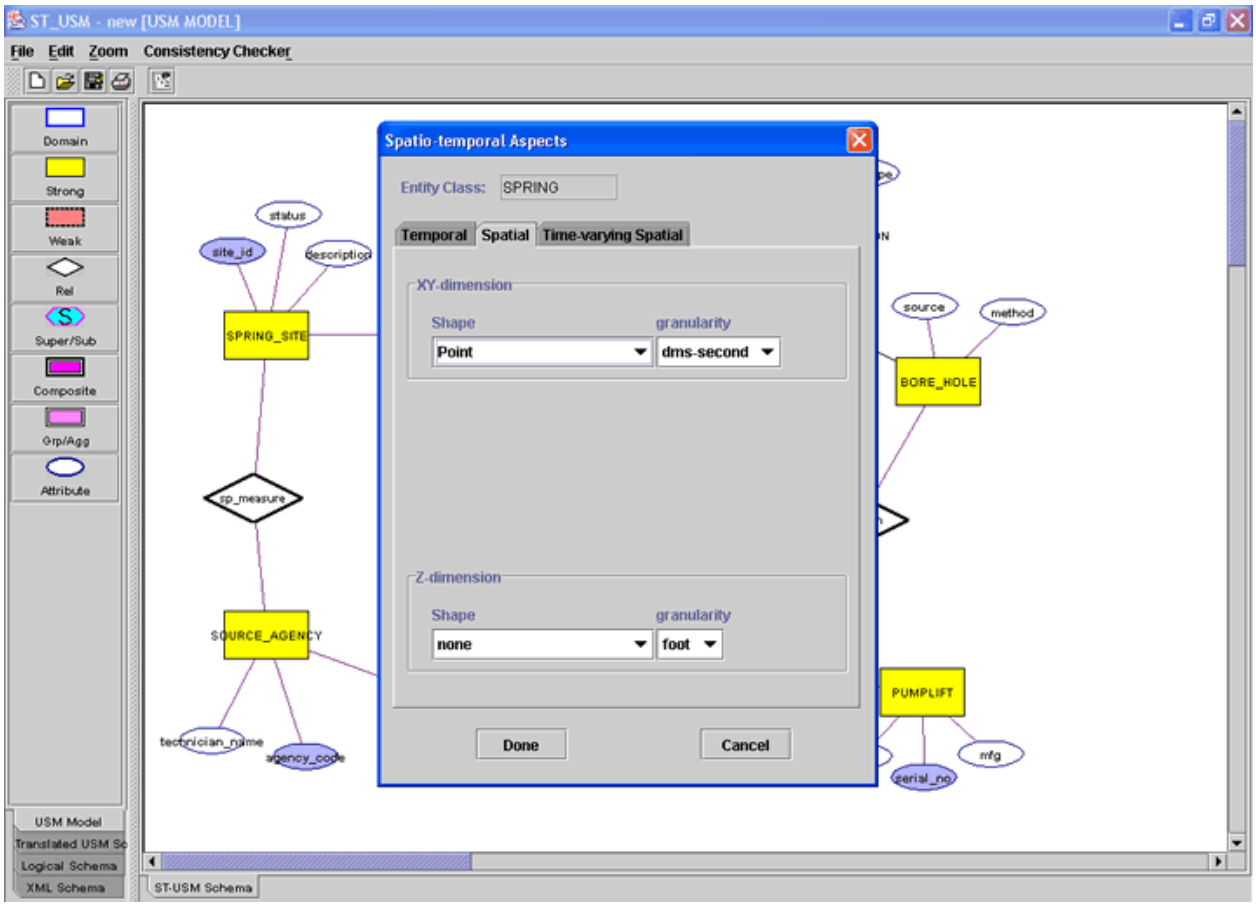

Figure 5: Specifying Spatial Aspects

The interaction between an object and space-time can result in change in the shape and/or change in the position of an object. A time-varying spatial annotation can be specified only if spatial and temporal annotation have already been specified. For example, a class of moving car tracked by satellite may be represented by an annotation phrase "E(sec)/-// P(deg)/ P(deg)/-//Pos@xy," which denotes a time-varying position while the shape is time-invariant. The geometry is a point (P) in an *x*-*y* plane with a spatial granularity of degree. The position changes in the *x*-*y* plane (Pos@xy) over time and each geometry is valid for time granules (E, i.e., event) measured in second.

Figure 6 shows the dialog panel for specifying the time-varying spatial aspects of an application. The four options are: neither shape or position is changing, shape is changing, position is changing, and both shape and position are changing over time. Additionally the dimension over which the change is happening can be specified.

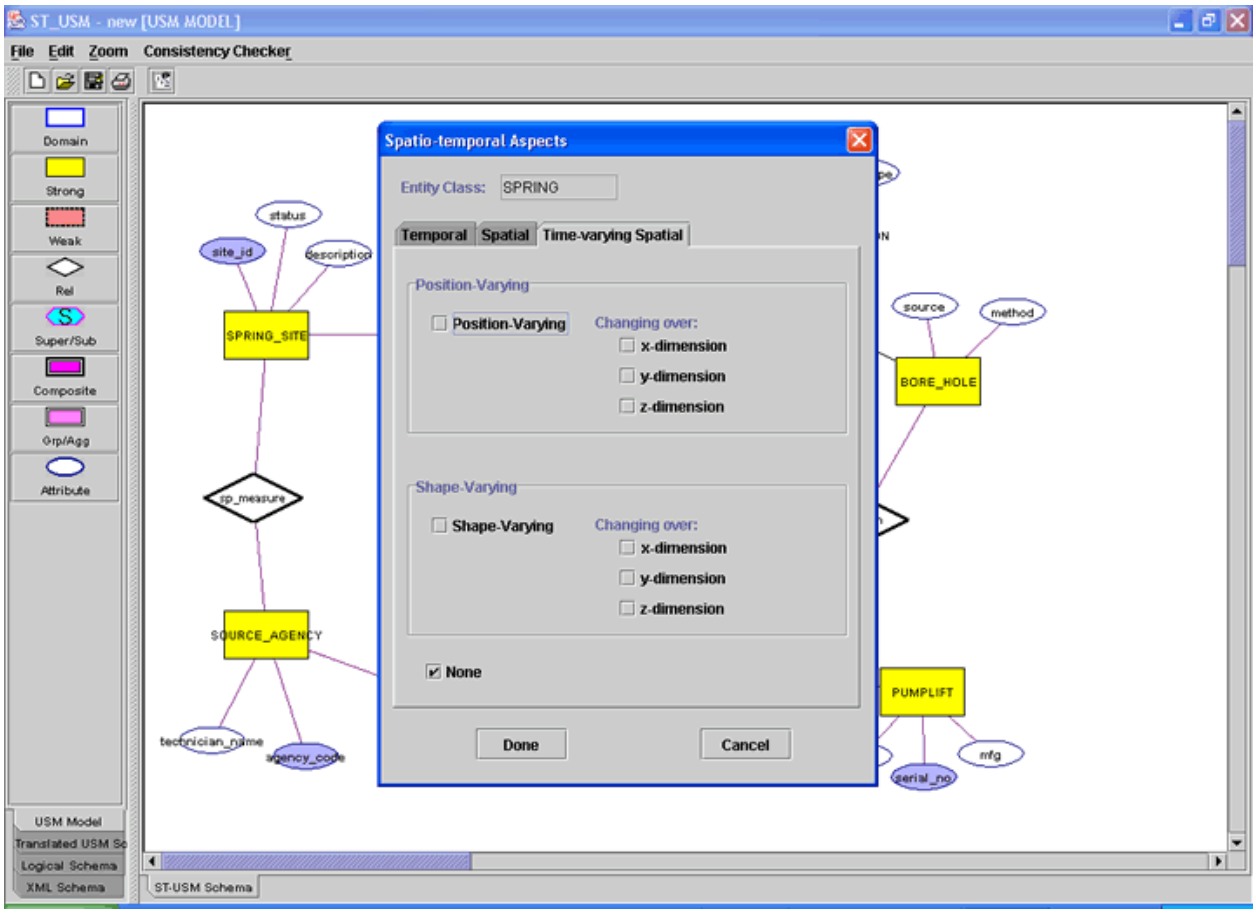

Figure 6: Specifying Time-Varying Spatial Aspects

In summary, for each construct in the core USM schema the database analyst, in consultation with the users, considers whether temporality and spatiality are important for the application. The database analyst asks questions like: Do you want to store the history or only the current value of this fact? Do you want to capture valid time or transaction time, or both? What is the associated temporal granularity? Is it important to store the geographical reference for the objects? What is the geographical shape of the objects? What is the associated spatial granularity? Can the spatial shape/position for these objects change over time? Accordingly, the database analyst enters the details using the dialog panel as shown in Figure 4, Figure 5 and Figure 6.

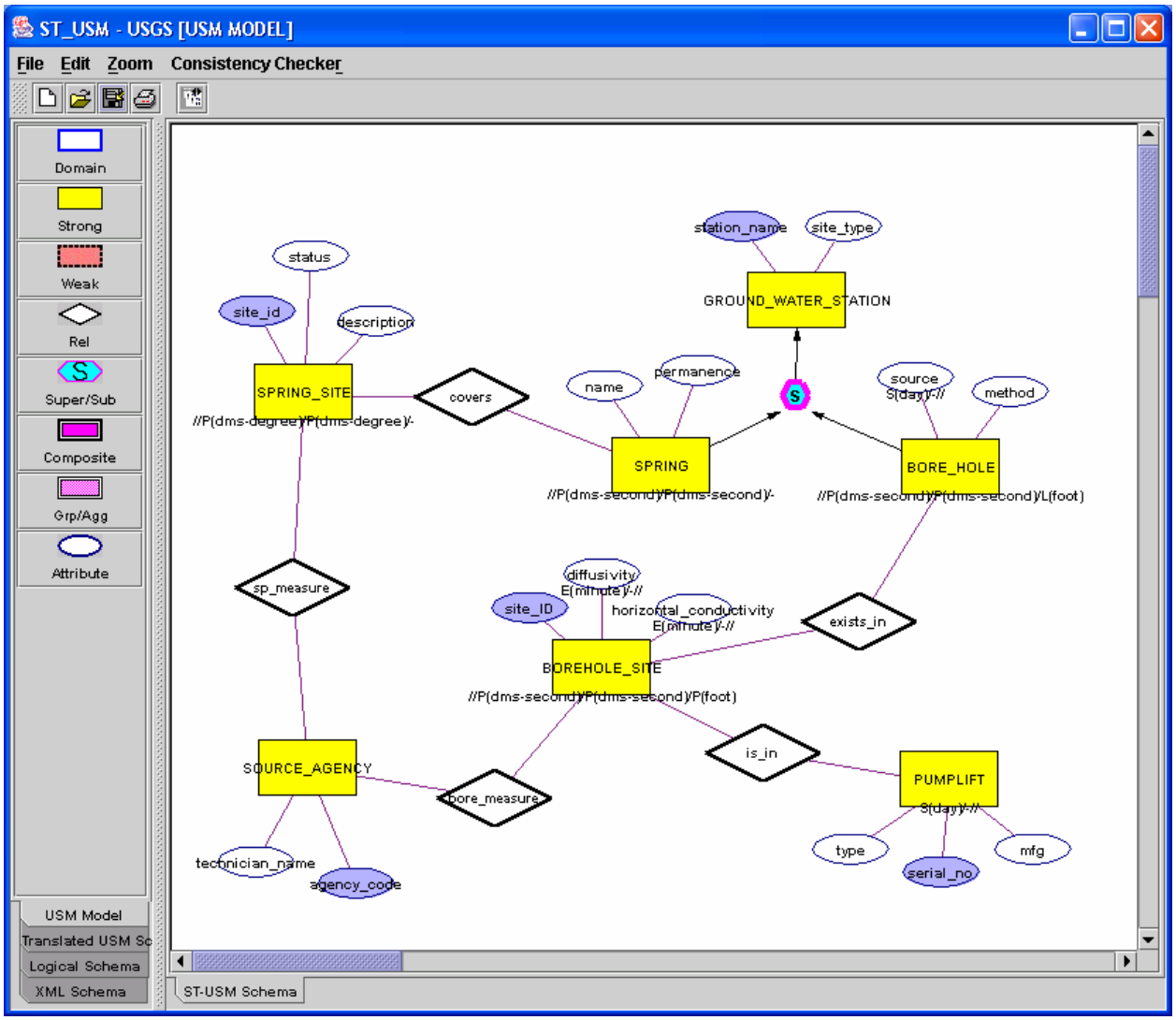

Figure 7: Annotated Schema

The schema shown in Figure 7 is automatically annotated according to the information filled into the pop-up boxes. The database analyst can annotate each entity class (e.g., SPRING_SITE, BOREHOLE_SITE, SPRING, BORE_HOLE, PUMPLIFT), relationship and attribute (e.g., diffusivity, horizontal_conductivity, source). Once the database analyst has made the annotated schema, the requirements so collected can be established with other users. Note how the annotated schema encapsulates the spatio-temporal semantics for an application and can be employed to verify (with the user) if the spatio-temporal semantics have been captured in the schema.

Our annotation-based approach is *upward compatible* [9, 60], i.e., our approach renders conventional conceptual schemas spatio-temporal without affecting the legacy schemas. Upward compatibility requires that the syntax and semantics of the traditional conceptual model, e.g., [16, 49, 55], remain unaltered. The schema shown in Figure 7 includes both "un"annotated constructs (e.g., SOURCE_AGENCY), where space

and time may not pertinent/important for the application and annotated constructs (e.g., PUMPLIFT), where annotation phrase exemplifies another level for abstraction that represents the time- and space-related semantics.

4.3 Validating Consistency in a Spatio-Temporal Conceptual Schema

Once the spatio-temporal aspects of the application have been elicited and captured in the schema, the database analyst needs to check if the spatio-temporal aspects are consistent. Using an example of a temporal relationship, we show how an inconsistent schema is identified by the design-support environment.

A temporal relationship implies that we want to keep track of the evolution of the interaction between temporal entities in a relationship. For example, if bore_measure were a temporal relationship between two temporal entity classes BOREHOLE_SITE and SOURCE_AGENCY, it would imply that an application might include queries like "In the last six months, what are the various source agencies associated with the borehole site 12345."

A temporal/non-temporal relationship may be associated temporal/non-temporal entities. As shown in Table 1, a temporal relationship and the participating entities include four possibilities. A temporal relationship can be defined only when all the participating entities are also temporal.

	Non-temporal Relationship	**Temporal Relationship**
Non-temporal Entity Class	Currently valid relationship between currently valid entities (*1*)	N/A (*2*)
Temporal/ Time-varying Spatial Entity Class	Currently valid relationship among temporal (time-varying spatial) entities (*3*)	Temporal relationship among temporal (time-varying spatial) entities (*4*)

Table 1: The semantics of temporal/non-temporal relationship/entity class combinations

Note that this is an implication of the semantics of a relationship in a conventional conceptual model where relationships can only be defined between entities that exist. If the participating entity classes are temporal but the relationship is not (cell *3* of Table 1) the entities participating in the relationship should be valid *now*. A temporal relationship between non-temporal entities (cell *2* of Table 1) is not legal in ST USM, as that would imply existence of a relationship even when the associated entities do not exist (since the existence time of entities is unknown). If the database analyst does specify a temporal relationship between non-temporal entity classes, the design-support environment should flag this as an error.

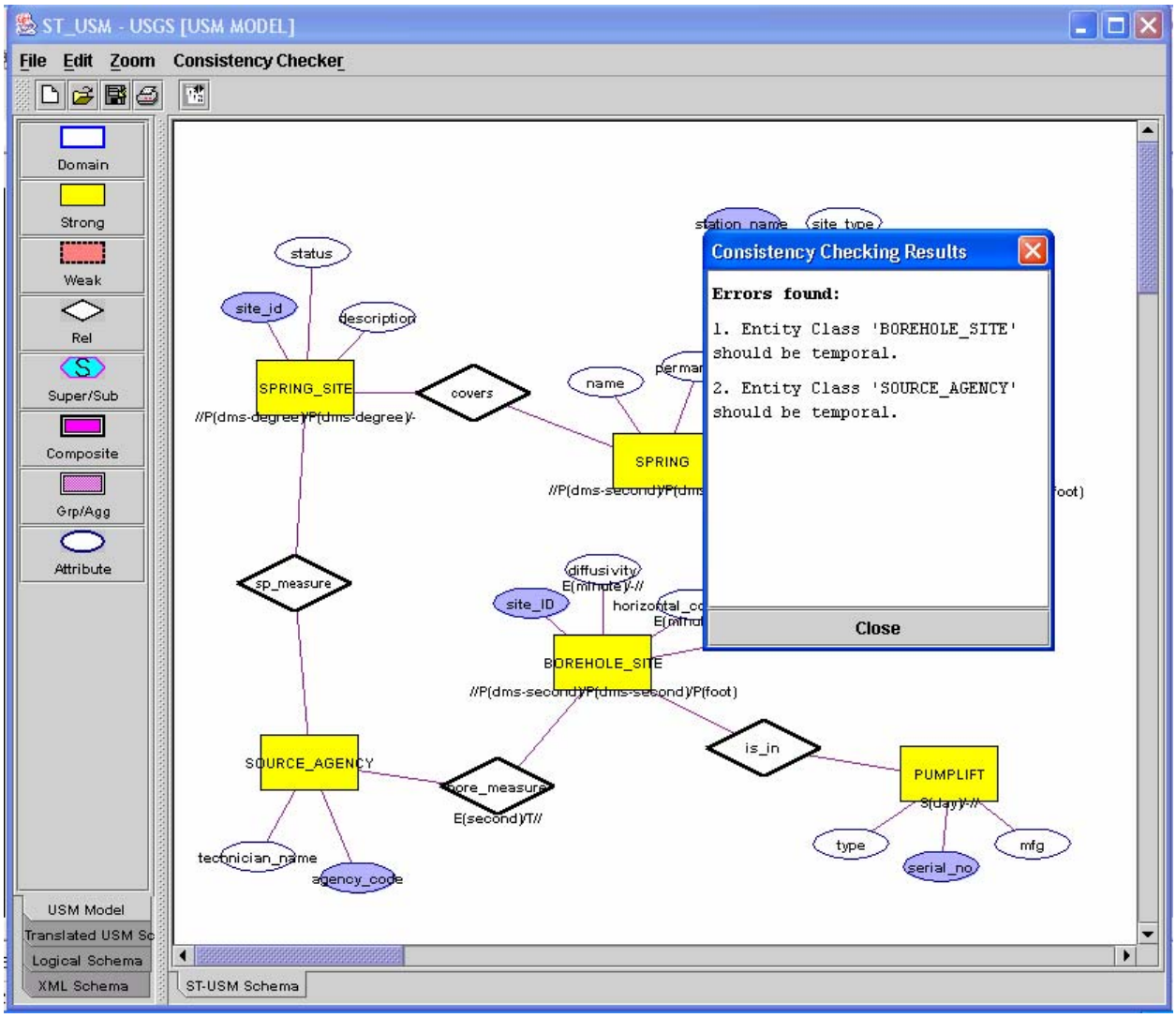

Figure 8: Semantic Error Log

If the database analyst were to specify a relationship (e.g., bore_measure) as temporal when the participating entity classes are not temporal, an error log is shown to the database analyst. For example, Figure 8 shows an error log that advises the database analyst that a temporal relationship like bore_measure implies that the participating entity classes (i.e., BOREHOLE_SITE and SOURCE_AGENCY) should also be temporal. The consistency checker in DISTIL ensures that the captured temporal and spatial data semantics—as specified by the database analyst—are always consistent. If they are not, a semantic error log like that in Figure 8 is shown to the database analyst. If the developed schema is found to be consistent, the consistency checker gives the message "Consistency checker found no errors."

The consistency checks supported by the design-support environment are based on *snapshot reducibility* [9, 56]. For incorporating spatio-temporal extension, snapshot reducibility implies "natural" generalization of the syntax and semantics of extant conventional conceptual models (e.g., [16, 49, 55]).

As shown in the example above, snapshot reducibility guides whether, e.g., temporal relationship can be defined between non-temporal entity classes. Table 1 showed how snapshot reducibility can be operationalized for a spatio-temporal conceptual model, and Figure 8 demonstrated how snapshot reducibility can be embedded into a design-support environment.

4.4 Semantics of the Annotated Schema

Our spatio-temporal design methodology uses annotations to capture the semantics of (temporal and spatial) sequenced statements. With annotations, our approach naturally extends the semantics of a conventional conceptual model (with implicit snapshot semantics), thereby inducing sequenced spatio-temporal semantics. For example, in a conventional conceptual model a *key attribute* [16] uniquely identifies an entity (at a point in time). A *temporal key* [58] implies uniqueness *at each point in time*. As may be evident, the semantics of a temporal key here are implied by the semantics of a key in a conventional conceptual model. Similarly, in a *temporal relationship*, the temporal element associated with a temporal relationship is constrained to be a subset of the intersection of the lifespan of the participating entities. Again, the semantics of a temporal relationship are implied from that of a (conventional) relationship where a relationship can be defined among entities that exist.

Annotations provide a mechanism to encapsulate concepts like history, lifespan and geo-referencing while hiding the concepts which have well-known semantics (e.g., valid time, point, line, region). In this section, we describe how the encapsulated semantics can be explicated. Note that explication of the temporal and spatial data semantics—resulting in what is referred to as a translated schema—is based on time and space ontology described in Section 3. Using an example of a temporal entity class, we show detailed semantics of an annotated abstraction. Details related to other abstractions like attribute, interaction relationship, subclass, composite class and grouping class are similar to those of a temporal entity class and described elsewhere [33].

A temporal entity class implies that the membership of an entity in the entity set is time-varying. We assume that a temporal entity class (as contrasted with entities of that class) exists during the entire modeled time. Thus, the existence time represents the lifespan of an entity and defines the time when facts associated with an entity can be true in the miniworld. Similarly, we can capture the transaction time associated with an entity, which may be important for applications requiring traceability. When an entity class is defined as temporal, it implies that the application would have queries like “What is the average monthly power consumption by all pumplifts over their installed existence?” and “What are the pumplifts that were installed before 1995 and are operational now?”

Figure 9 illustrates the representation of existence time expressed as state (S) with day as the temporal granularity name. Based on the users’ requirements, the database analyst simply annotates PUMPLIFT with

"S(day)/-//" and does not need to contend with the complexity of the underlying semantics or the associated temporal constraints. Figure 9 also shows the semantics of a temporal entity class in ST USM via a mapping using the concepts of a conventional conceptual model, which we refer to as a *translated USM schema*. This mapping from ST USM to (translated) USM is *snapshot equivalent*; i.e., the two schemas (ST USM and translated USM) represent the same information content over snapshots taken at all times. In order to express the semantics of a temporal entity class, we need to specify a TEMPORAL_GRANULARITY in which the evolution of a temporal object is embedded. The relationship PUMPLIFT_has_ET associates an entity with a corresponding TEMPORAL_GRANULARITY. Each TEMPORAL_GRANULARITY is uniquely identified by a granularity_name, shown by the underlined attribute. An extent is the smallest time interval that includes the image of a granularity and is expressed by two indexes, minimum and maximum. Each anchor_gran is a recursive relationship (i.e., a relationship where an entity from the same entity class can play different roles) such that each participating granularity optionally has an anchor (0:1) and each granularity is an anchor for 0 to many (i.e., 0:M) other granularities. The anchor of a granularity *TG* is the first index of a strictly finer granularity that corresponds to the origin of this granularity, i.e., *TG*(0). All granularities except the bottom granularity have an associated anchor. A finer-than and a coarser-than relationship between granularities are denoted by a recursive relationship groups_into, where one entity plays the role of finer-than and the other the role of coarser-than. The relationships anchor_gran together with groups_into helps create a *granularity graph* [14], which can help a user choose the level of detail associated with facts. Details related to granularities and indeterminacy is presented elsewhere [34].

A temporal entity with existence time may have a set of event_instants or state_periods associated with it depending on whether a temporal entity is represented as an event or a state. A time period of PUMPLIFT is represented with indexes begin and end of state_periods. A double-lined ellipse in USM denotes a multi-valued attribute. For example, state_periods is represented as a multi-valued attribute and represents a set of state periods (i.e., a temporal element) associated with an entity. Additionally, eight constraints described below will be generated in the translated USM schema for PUMPLIFT. These constraints are implicit in the ST USM schema but are explicit in the translated USM schema. Constraints 4.4.1 and 4.4.2 are based on the definition of a temporal entity; i.e., a temporal entity has an associated temporal granularity and an associated temporal element. Constraints 4.4.3–4.4.5 are based on the definition of temporal granularity, and these constraints need to be generated once for the entire schema. Constraints 4.4.6–4.4.8 are based on the definition of a temporal element. In these definitions, we assume a *closed-open representation* [58], i.e., the begin index is contained in the period while the index corresponding to end is not.

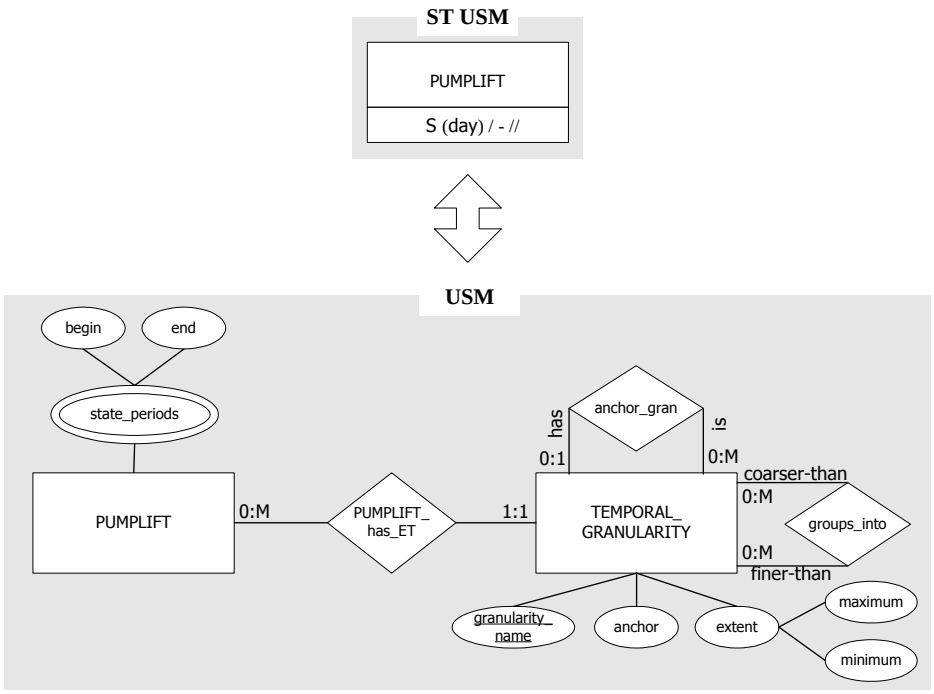

Figure 9: Temporal Entity Class in ST USM and its semantics in USM

For example, an instant for a temporal element may be represented by [17, 18). In this example, begin index (i.e., 17) is inclusive in the instant while end index (i.e., 18) is not.

Constraint 4.4.1: The existence time for all the entities of PUMPLIFT have the same associated granularity.
$\forall\, e \in S$(PUMPLIFT),
e. PUMPLIFT_has_ET.TEMPORAL_GRANULARITY (granularity_name) = day

Constraint 4.4.2: For every entity of PUMPLIFT, there exists an associated temporal element with well-formed periods.
$\forall\, e \in S$(PUMPLIFT), $\exists p \in e$(state_periods), p.begin < p.end

Constraint 4.4.3: Each TEMPORAL_GRANULARITY has a lower and an upper bound referred to as minimum and maximum; these bounds are well-formed.
$\forall\, e \in S$(TEMPORAL_GRANULARITY), e(extent.minimum) < e(extent.maximum)

Constraint 4.4.4: All the granularities, except one, have an anchor. The bottom granularity is allowed not to have an anchor.
$\forall\, e_1 \in S$(TEMPORAL_GRANULARITY), $\neg$ has(e_1.anchor_gran) $\Rightarrow$
$\neg\, (\exists\, e_2 \in S$(TEMPORAL_GRANULARITY) $\wedge\, e_1 \neq e_2 \wedge \neg$ has(e_2.anchor_gran))

Constraint 4.4.5: For a temporal granularity, if an anchor does not exist then that is the bottom granularity not having any granularity finer than it; in other words, it cannot take the role of coarser-than in the relationship groups-into.
$\forall\, e \in S$(TEMPORAL_GRANULARITY), $\neg$ has(e.anchor_gran) $\Rightarrow \neg$ coarser-than(e.groups_into)

Constraint 4.4.6: State periods of an entity of PUMPLIFT are well-formed.
$\forall\, e \in S$(PUMPLIFT), $\forall p \in e$(state_periods), p.begin < p.end

Constraint 4.4.7: Temporal elements are well-formed. A temporal element is defined as a union of non-overlapping time intervals.
$\forall\, e \in S$(PUMPLIFT), $\forall\, p_1, p_2 \in e$(state_periods), p_1.begin < p_2.begin $\Rightarrow$ p_1.end $\leq$ p_2.begin

Constraint 4.4.8: The extent of a temporal granularity defines the upper and lower bounds for any temporal element. In other words, a temporal element cannot include an index that is larger than the corresponding extent.maximum or smaller than the corresponding extent.minimum.
$\forall\, e \in S$(PUMPLIFT),
$\forall\, p \in e$(state_periods), *e*. PUMPLIFT_has_ET.TEMPORAL_GRANULARITY (extent.minimum) $\leq$
p.begin < p.end $\leq$ *e*. PUMPLIFT_has_ET.TEMPORAL_GRANULARITY (extent.maximum)

As may be evident, a straightforward annotation phrase (specifically "S(day)/- ") represents relatively involved semantics described in this section. We next show how the encapsulated spatio-temporal semantics can be explicated using DISTIL.

To view detailed explicit semantics associated with the annotated schema, the database analyst clicks on the "Translated USM Schema" tab to obtain a translated USM schema corresponding to the ST-USM schema (Figure 10).

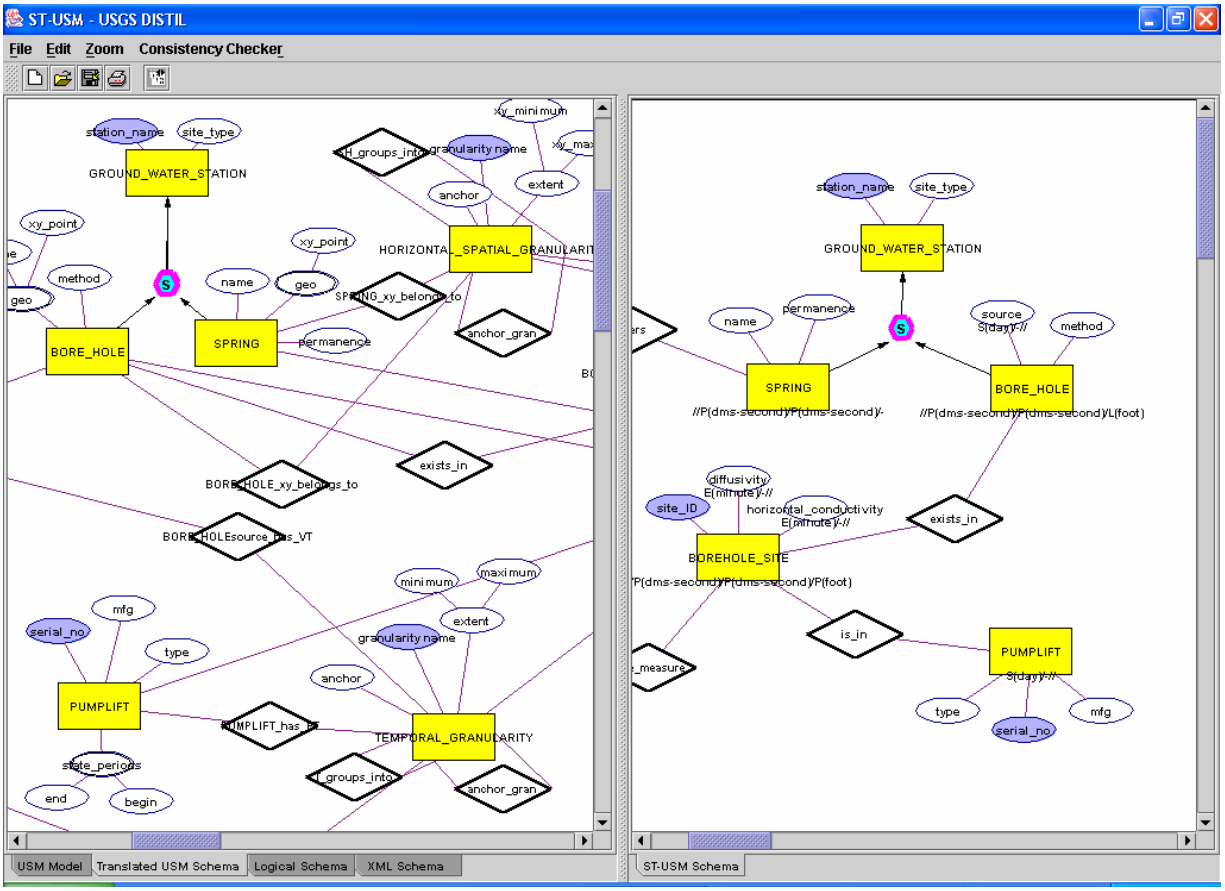

Figure 10: Semantics of the Annotated Schema

We have embedded translation rules (e.g., Figure 9) into DISTIL, to help translate the annotated constructs to an equivalent USM schema with explicit representation of the associated spatiality and temporality. In Figure 10, the portion of the ST-USM schema on the right has been translated into the conventional USM schema on the left. For example, the semantics associated with a temporal entity class PUMPLIFT includes an entity class TEMPORAL_GRANULARITY, which specifies the temporal granularity in which PUMPLIFT is embedded. The relationship PUMPLIFT_has_ET relates an entity from PUMPLIFT with a corresponding temporal granularity. A multi-valued attribute state_periods (with components begin and end) is added to the entity PUMPLIFT because PUMPLIFT lifespan was modeled as state. A multi-valued

attribute implies that each PUMPLIFT can have many associated state_periods. Similarly, other constructs of the annotated ST-USM schema are converted to a translated USM schema using the embedded rules in DISTIL.

Note how annotation phrases succinctly encapsulate the spatio-temporal data semantics on the conceptual schema. The translated schema that is shown in Figure 10 includes only a fragment of the entire schema, which includes 12 entity classes, 51 attributes, 22 relationships and 42 constraints. Thus, a relatively straightforward annotated schema (e.g., Figure 7) encapsulates rich-semantics explicated in the translated schema. In this section, we described how our *formal* ontology-based approach helps elicit—at a conceptual level—the spatio-temporal data semantics, e.g., *event* and *state* [29], *valid time* and *transaction time* [59], *existence time* [24], *temporal granularities* [7, 8, 14], *shape* and *position* [13], *spatial resolution* [75, 76], and *change in position* and/or *shape* over time [48, 68]. These formalized semantics are the basis for the development to the representational schema, which we describe next.

4.5 Logical Schema

While conceptual models provide a mechanism to that is close to the way users perceive the data, physical models provide concepts how data is stored in the computer. A *representational model*, e.g., relational model, provides concepts that may be understood by users and are not too far removed from the way data is organized in the computer [16]. Mapping rules that provide correspondences between conceptual and representational model constructs are applied in logical design and result in the development of a *logical schema*.

The mappings to a logical schema depend on the type of representational model used. Standard SQL (Structured Query Language), the basis for a relational schema, does not include time support except for user-defined time. As a result, over 50 temporal query languages have been proposed [29], most of which are a result of extending SQL for the temporal domain. While change proposals to SQL3 that includes temporal semantics is referred to as SQL/Temporal [60, 61], the Open GIS Consortium has proposed extensions to SQL that supports simple geospatial collections that is referred to as SQL/OGIS [40]. For each of these languages, mapping rules may be developed to convert an ST USM schema to a logical schema in that language. As illustrated in Figure 2, there are two mapping rules, which depend on the logical model used: a non-temporal logical model (e.g., SQL) or a temporal or spatio-temporal conceptual model (e.g., SQL/Temporal, SQL/OGIS). None of the existing DBMS products yet support a spatio-temporal logical model. So, we do not discuss this latter case further, other than to note that such a mapping could be developed as an extension of a previously-developed mapping from ER to SQL/Temporal [58]. Instead, we briefly describe a mapping from a spatio-temporal conceptual schema to a logical schema that does not support space and time semantics, utilizing additional columns to denote

the spatial and temporal extent of each row. This mapping is more complex than one to a spatio-temporal logical data model, because there are no special spatio-temporal constructs, such as temporally-sequenced primary and foreign keys, provided by the logical model that can be exploited in the mapping.

The translation from conceptual schema to logical schema is similar to the one described in standard database textbooks [16]. Each constraint described in the previous section will be translated to assertion. For example, we have a sequenced constraint for BOREHOLE: there can only be one value for source at any point in time. This can be enforced with the following assertion [58].

```
CREATE ASSERTION source_assertion
CHECK (NOT EXISTS (SELECT *
                   FROM BOREHOLE AS I1
                   WHERE 1 < (SELECT COUNT(source)
                   FROM BOREHOLE AS I2
                   WHERE I1.source = I2.source
                            AND I1.BEGIN < I2.END
                            AND I2.BEGIN < I1.END))
         AND NOT EXISTS (SELECT *
                   FROM BOREHOLE AS I
                   WHERE I.source IS NULL)
);
```

This assertion can be implemented using a trigger in Oracle [58].

```
CREATE OR REPLACE TRIGGER source_TRIGGER
AFTER INSERT OR UPDATE ON BOREHOLE
DECLARE
         valid INTEGER;
BEGIN
         SELECT 1
         INTO valid
         FROM DUAL
         WHERE NOT EXISTS (SELECT *
                   FROM BOREHOLE  I1, BOREHOLE I2
                   WHERE I1.source = I2.source
                            AND I1.BEGIN < I2.END
                            AND I2.BEGIN < I1.END
                            AND I1.rowed <> I2.rowed )
                   AND NOT EXISTS (SELECT *
                            FROM BOREHOLE AS I
                            WHERE I.source IS NULL);
EXCEPTION
         WHEN NO_DATA_FOUND THEN
         RAISE_APPLICATION_ERROR (-20001, 'source can have one value at any point in time');
END;
```

Once the users' spatio-temporal requirements have been captured and validated, the database analyst can click the "Logical Schema" tab to get the logical schema (Figure 11). This schema includes the name of the table, attributes in the table, primary key and foreign key (if any). For example, PUMPLIFT table includes four attributes: serial_no, type, mfg and granularity_name. The primary key (PK) is serial_no and the foreign key (FK) is TEMPORAL_GRANULARITY.granularity_name.

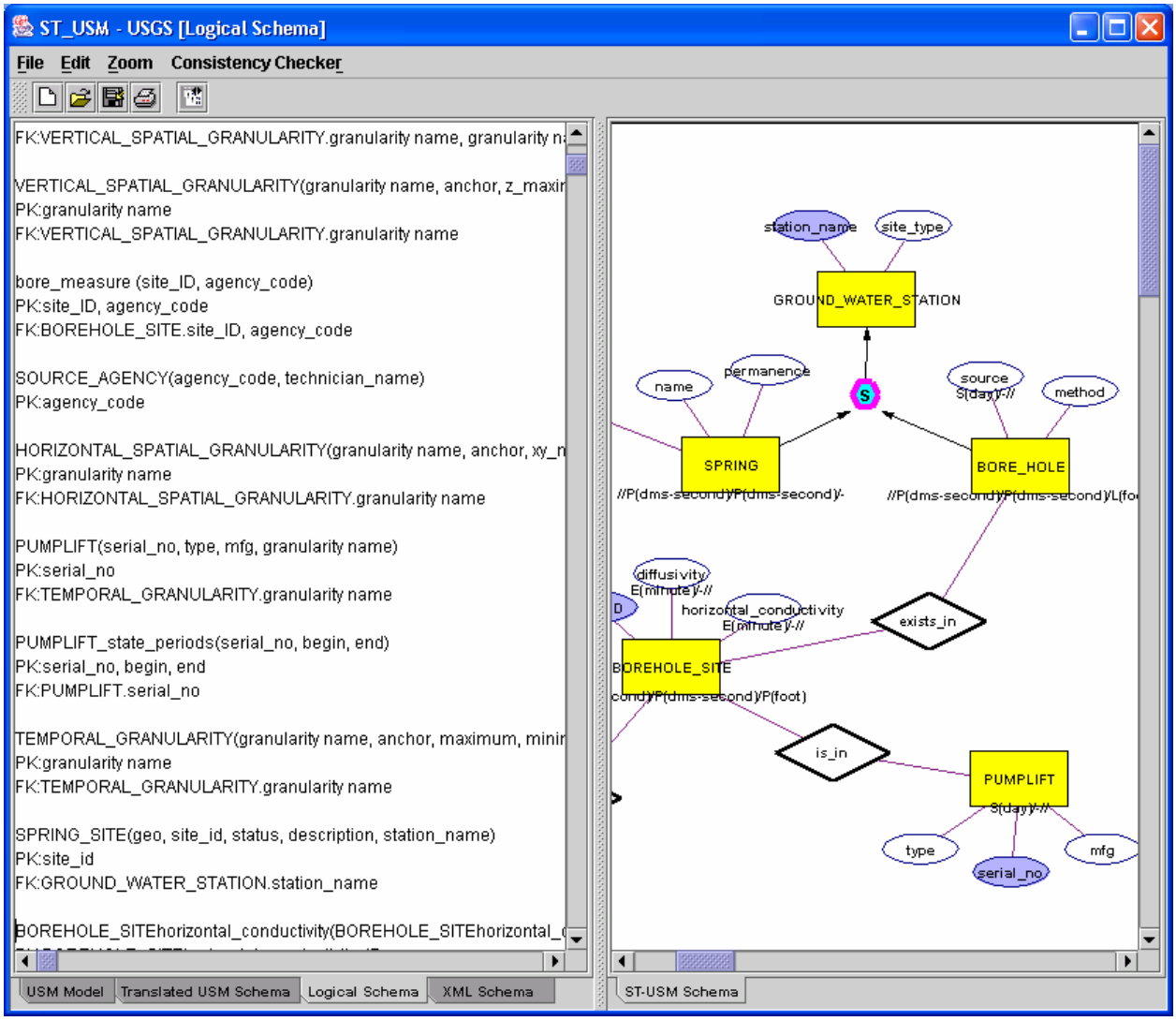

Figure 11: Logical Schema

While the tool described above gives a textual description of the logical schema, this can be tailored for specific DBMSs, such as Oracle. For example, Oracle Spatial [41, 42] includes a pre-defined object type SDO_GEOMETRY and a table for SPRING can be defined as shown below.

```
CREATE TABLE SPRING (
                station_name VARCHAR2(30) PRIMARY KEY,
                name VARCHAR2(100),
                permanence NUMBER (5,2),
                geo MDSYS.SDO_GEOMETRY);
```

We described how different mapping rules can be employed to translate a given conceptual schema to a logical schema. These mapping rules depend on the specific logical model and on the DBMS under consideration. We demonstrated how a methodical approach ensures that the spatio-temporal semantics elicited during conceptual design are embedded in the subsequent logical schema. A design-support

environment like DISTIL can automate this translation, thus, ensuring that the rich spatio-temporal semantics captured during conceptual design are not "lost" during translation to the logical schema.

4.6 Encoding the Schema as an XML Document

XML provides a standard format for exchanging conceptual schema among diverse design-support environments. If the schemas are generated by different design-support environments—with different syntax but same semantics—the XML document can enable sharing of the conceptual schema across platforms. Using an example of PUMPLIFT, we show the translation rules to an XML document.

While the first part of the XML document shown in Figure 12 describes the semantics of conventional conceptual schema (e.g., USM), the second part corresponds to the spatio-temporal annotations.

```
<?xml version="1.0" ?>
- <ST_USM xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.cs.arizona.edu/timecenter/STUSM/usm.xsd">
  - <simple_class name="PUMPLIFT">
      <normal_attribute name="serial_no" datatype="INTEGER" mandatory="yes" />
      <normal_attribute name="type" datatype="STRING" mandatory="no" />
      <normal_attribute name="mfg" datatype="STRING" mandatory="no" />
    - <primarykey_constraint>
        <normal_attribute name="serial_no" />
      </primarykey_constraint>
    </simple_class>
  - <ST_ANNOTATION>
    - <simple_class name="PUMPLIFT">
        <state granularity="day" />
      </simple_class>
    </ST_ANNOTATION>
</ST_USM>
```

Core Schema: "What" Semantics

Annotations: "When"/ "Where" Semantics

Figure 12: XML Document for a fragment of ST-USM Schema

The second part of the XML document—between ⟨ST_ANNOTATION⟩ and ⟨/ST_ANNOTATION⟩—specifies the spatio-temporal data semantics. In this case, the temporal entity class PUMPLIFT is represented as a state having a granularity of day. Note how the orthogonality of spatio-temporal semantics in a conceptual schema (e.g., Figure 7) renders an XML document where the conventional semantics (i.e., "what") are segregated from the "when/where" semantics.

Having, described the translation from conceptual schema to XML document, we next show how a design-support environment supports such a translation. The database analyst can click on "XML Schema" tab to obtain the XML document corresponding to the ST USM Schema. This XML file can be saved ("Save XML File") and is useful for sharing schemas among database analysts using different design-support environments.

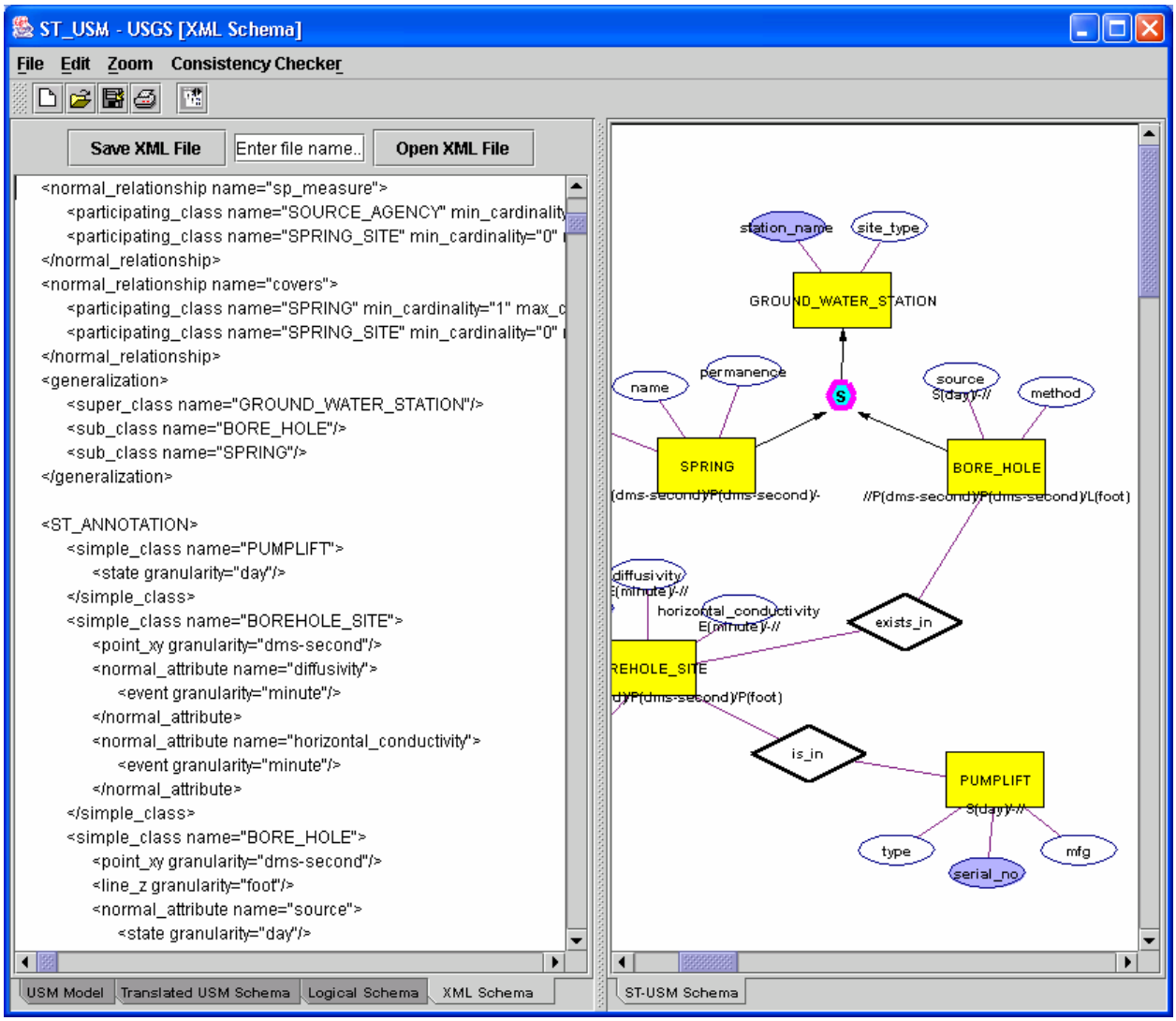

Figure 13: XML Document

We showed how a spatio-temporal conceptual schema can be "exported" to an XML document. The XML document can be "imported" into another design-support environment, with possibly different syntax, but same underlying semantics, i.e., based on abstractions like classification, association, generalization/specialization and aggregation. We illustrated how the "when" and "where" semantics can be kept orthogonal to the "what" semantics even in the XML document (cf. Figure 12).

5 Architecture

We describe the underlying architecture of DISTIL (Figure 14) that enables the development of spatio-temporal conceptual schemas. Figure 14 also explicates our spatio-temporal conceptual design approach that advocates first capturing "what" is important for the application (during *Conventional Conceptual*

Design) and only then augmenting the schema with the "when/where" semantics (during *Spatio-Temporal Conceptual Design*).

As illustrated in Figure 14, the database analyst first develops a *Core USM Schema* during Conventional Conceptual Design using *USM Schema Designer*. The USM Schema Designer allows the database analyst to develop a core conceptual schema (e.g., Figure 3) that captures *current reality* without considering *temporal* or *spatial aspects*. We have adapted the USM Schema Designer from the data modeling module of CREAM [45].

Our Spatio-Temporal Conceptual Design includes annotating the Core USM Schema via *Annotation Designer* (e.g., Figure 4, Figure 5) resulting in the *ST-USM Schema* (e.g., Figure 7) and validating the consistency of the captured spatio-temporal semantics via *Consistency Checker*. If any inconsistencies are detected in the spatio-temporal annotations, they are shown via the *Semantic Error Log* (e.g., Figure 8). This architecture also supports the translation of existing USM schemas that have spatio-temporal semantics incorporated in an ad-hoc manner (perhaps because they were designed with CASE tools that did not have such support). The user loads such a schema in as a Core USM Schema. For each time-varying entity class, relationship, and attribute, the user can annotate that semantic object, then manually remove the ad hoc modeling constructs. For example, if the original schema had used a ternary relationship to model a time-varying binary relationship (with one of the entity classes being a time value), the relationship could be designated as time-varying with an annotation, then the time value entity class and its connection to the relationship removed, leaving a simpler schema with the same abstract semantics. The user could then add more detail to the annotation, such as granularity, indeterminacy, kind of time; it is doubtful that all of these details were in the original schema. Because the resulting schema is at a higher level of abstraction, different logical schemas could then be generated, under user control.

A consistent *ST-USM Schema* (e.g., Figure 7) can next be converted to a *Translated USM Schema* (e.g., Figure 10) through the *Semantic Mapper*. While the spatio-temporal semantics are encapsulated in the ST-USM schema, these semantics are explicated in the translated USM schema. While the consistent ST-USM schema can be employed for eliciting and validating (with the user) the spatio-temporal requirements for an application, the translated USM schema is useful for translation to a logical schema that is interpretable by a DBMS.

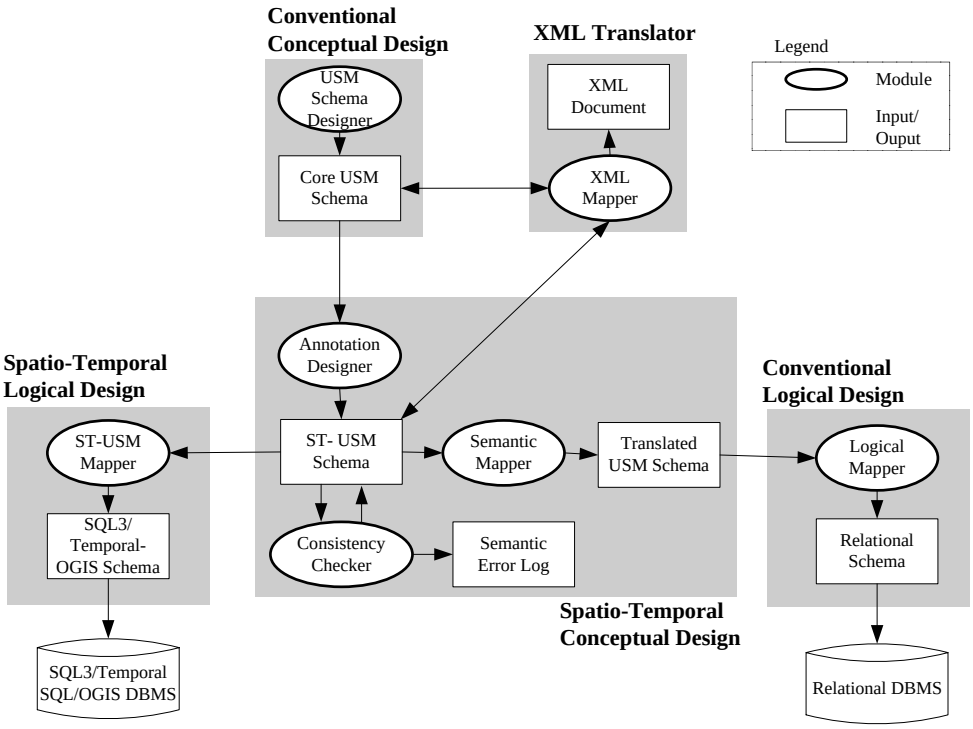

Figure 14: DISTIL Architecture

As shown in Figure 14, our proposed Spatio-Temporal Conceptual Design implemented via DISTIL integrates with Conventional Conceptual Design and the translated USM Schema merges again with the *Conventional Logical Design.* The *Logical Mapper* in Conventional Logical Design includes rules to convert a Translated USM Schema to *Relational Schema* with spatial support that can be implemented in a relational DBMS. The *XML Mapper* converts a USM or ST USM Schema to an *XML Document*. The Schema in XML format can be shared by distributed teams which may be, possibly, using different design-support environment with different modeling formalism (i.e., syntax) but the same underlying semantics. In the future, it would be useful to incorporate the *ST-USM Logical Mapper*, which would translate the ST-USM schema to an *SQL3/Temporal-OGIS Schema* that can be implemented in a spatio-temporal DBMS.

DISTIL has been implemented using Java 2 (JDK 1.2) and Oracle 8.1.6. A dialog panel in DISTIL (cf., Figure 4, Figure 5, Figure 6) is created to elicit the spatial and temporal semantics associated with an entity class, a relationship or an attribute. When a user clicks an icon of a persistent object (e.g., an entity class), a dialog panel pops up and allows the user to input the spatial and temporal information of that persistent object. The “annotation” class is a Java Bean implemented to capture the spatial and temporal aspects associated with a persistent object that becomes a property of the persistent object. This annotation class also summarizes the spatial and temporal information into a simple annotation string. This string is displayed on the drawing canvas and stored in the database. An application can keep track

of the spatial and temporal information of an object class simply by looking up the annotation string that accompanies the object.

Having described the design-support environment, we evaluate DISTIL along with our proposed underlying approach.

6 Evaluation

Batini et al. [5] posit that conceptual models should possess the following qualities: *expressiveness, simplicity*, *minimality* and *formality*.

Expressiveness refers to the availability of a large variety of concepts for a more comprehensive representation of the real world. Wand et al. [72] propose that "conceptual modeling can be anchored in the models of human knowledge" and that ontology be employed as the basis a proposed formalism. We propose intuitive ontology-based dialog panels in DISTIL, which comprehensively capture the semantics related to space and time. These pop-up boxes automatically annotate the schema, thus, helping represent the spatio-temporal data semantics on the conceptual schema.

One of the conflicting goals related to expressiveness is simplicity, which requires that augmenting an existing design-support environment should require minimal. As shown in Table 2, the additional number of classes (and lines of code) required to augment an existing design-support environment (i.e., [45]) with the spatio-temporal semantics is relatively modest.

Module	**Classes**	**Lines of Code (kLOC)**
USM Schema Designer	118	34.1
Annotation Designer	6	2.2
Semantic Mapper	3	1.4
Logical Mapper	2	1.0
XML Mapper	5	2.4

Table 2: Number of Classes and Lines of Code for Modules of DISTIL

The annotation designer and semantic mapper entailed a 10.6% increase in the lines of code. Additionally, adding the spatio-temporal semantics via annotations did not entail any changes to the existing code. Thus, orthogonality of a conceptual framework (i.e., ST USM) is mirrored with orthogonality of an implementation structure (i.e., DISTIL). Additionally, to incorporate annotations the changes to the database schema were minimal. The XML document encoding an annotated ST USM Schema captures spatio-temporal semantics orthogonal to conventional semantics. All this implies that our approach is straightforward from the perspective of repository (database) design and application development.

Minimality ensures that no concept can be expressed through composition of other concepts. Because of orthogonality in the annotated ST-USM schema and the corresponding XML document, minimality is also supported.

Formality specifies that the model must present a unique, precise and well-defined interpretation. Wand et al. [73] posit that effective use of conceptual modeling constructs requires that their meanings be defined "rigorously." The syntax and semantics of the underlying ST-USM is formally defined using BNF (cf. Appendix) and first-order logic [33], respectively.

Snapshot reducibility ensures that the semantics of a spatio-temporal conceptual model are understandable in terms of the semantics of the conventional conceptual model, thus, helping ensure minimum additional investment in database analyst training.

Upward compatibility allows the legacy and spatio-temporal schemas to co-exist. Upward compatibility requires that the syntax and semantics of the traditional conceptual model, e.g., [16, 49, 55], remain unaltered. Our proposed approach has not altered the syntax or semantics of extant models.

With our annotation-based approach, we claim to have achieved expressiveness and formality along with simplicity in spatio-temporal conceptual modeling.

7 Related Work

Given the need to capture spatio-temporal data semantics, various formalisms have been proposed. However, the extant spatio-temporal database design methodologies—e.g., [26, 27, 53]—do not integrate naturally into extant conventional conceptual design. While some approaches that add new constructs (e.g., [19, 65, 66]) have changed the syntax of conventional conceptual models, some other approaches (e.g., [17, 18, 65, 66]) have even resorted to changing the semantics of conventional conceptual models.

With the growth of geographic applications, recently design tools [6, 36, 44] to support modeling of spatio-temporal databases have been proposed. Perceptory [6] and GeoOOA [36] focus on capturing the semantics associated with geometry of the spatial objects, where spatiality of entities is defined by a relationship link with geometry, e.g., point, line and region. A visual schema editor based on MADS [6, 36, 44] aids in capturing some spatio-temporal semantics. However, none of these tools provide a mechanism to capture semantics related to granularity and indeterminacy. Granularities related to facts need to be captured during conceptual design because under-specifying granularities can restrict an application and affect relative ordering of events. Additionally, none of the extant design-support environments help validates the consistency of the schemas. As far as we know, there does not exist a tool that helps translate a conceptual schema to an XML document. Because extant tools do not embed a generic spatio-temporal database design methodology that integrates naturally with conventional conceptual design, methodologies proposed via these tools cannot be easily adopted by existing CASE tools that support conventional conceptual design.

Our approach to spatio-temporal conceptual modeling is comprehensive and captures various aspects related to temporality and spatiality, e.g., valid time, transaction time, events, states, position, geometry,

shape, granularities and indeterminacy. Additionally, our proposed annotation-based approach used in DISTIL integrates into existing database design methodologies. Annotating the schema is intuitive from the perspective of database analysts and users as it corresponds to the way human beings perceive spatio-temporal objects. Moreover, incorporating annotation—via pop-up boxes—into an existing CASE tool is also straightforward to implement. Additionally, we show how schemas developed via distributed design tools can be shared using XML document.

8 Conclusion

We described an approach to supporting spatio-temporal conceptual database design and described a proof-of-concept spatio-temporal conceptual modeling design environment called DISTIL. DISTIL enables capturing "what" is pertinent for a database application along with the "when/where" semantics. This ST-USM schema—developed using DISTIL—can be used as communication vehicle: it can also be used to decide if all the spatio-temporal requirements of the user have been captured and whether the requirements are conflicting. Schemas developed via DISTIL can be saved as XML documents, which can be used for schema exchange among database analysts. Via DISTIL, we demonstrated how augmenting an existing CASE tool with our proposed the annotation-based approach is straightforward to implement. An empirical study that evaluated our spatio-temporal conceptual modeling approach on comprehension and ease of use is outside the scope of this paper, and described in detail elsewhere [33].

In the future, we plan to incorporate the *ST-USM Logical Mapper*, which would translate the ST-USM schema to an *SQL3/Temporal Logical Schema* [60, 61] that can be implemented in a temporal DBMS. We want to consider the details of tailoring the mapping for specific DBMSs, e.g., Oracle. Additionally, we want to investigate generating triggers in the logical schema that are the temporal and spatial equivalents of the non-temporal constructs in USM.

9 Acknowledgements

We are grateful to Arvind Gupta, Jun Liu, Mihir Shah and Xingtao Wang for their help in development of various parts of DISTIL. This research was supported in part by NASA grant 314401, and NSF grants IIS-0100436 and EIA-0080123.

10 References

[1] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, No. 11, pp. 832-843, 1983.

[2] K. K. Al-Taha, R. T. Snodgrass, and M. D. Soo, "Bibliography on Spatiotemporal Databases," *International Journal of Geographical Information Systems*, vol. 8, No. 1, pp. 95-103, 1994.

[3] J. R. Anderson and G. H. Bower, *Human Associative Memory*. Washington, D.C.: V. H. Winston & Sons, 1973.

[4] R. D. Banker and R. J. Kauffman, "Reuse and Productivity in Integrated Computer-Aided Software Engineering," *MIS Quarterly*, vol. 15, No. 3, pp. 375-401, 1991.
[5] C. Batini, S. Ceri, and S. B. Navathe, *Conceptual Database Design: An Entity-Relationship Approach*: Benjamin/Cummings Publishing Company, 1992.
[6] Y. Bedard, "Visual Modeling of Spatial Databases: Towards Spatial PVL and UML," *Geomatica*, vol. 53, No. 2, pp. 169-186, 1999.
[7] C. Bettini, C. E. Dyreson, W. S. Evans, R. T. Snodgrass, and X. S. Wang, "A Glossary of Time Granularity Concepts," in *Temporal Databases: Research and Practice*, O. Etzion, S. Jajodia, and S. Sripada, Eds.: Springer-Verlag, 1998, pp. 406-413.
[8] C. Bettini, S. Jajodia, and S. X. Wang, *Time Granularities in Databases, Data Mining and Temporal Reasoning*. Berlin: Springer-Verlag, 2000.
[9] M. H. Böhlen, C. S. Jensen, and R. T. Snodgrass, "Temporal Statement Modifiers," *ACM Transactions on Database Systems*, vol. 25, No. 4, pp. 407-456, 2000.
[10] J. C. Brancheau, B. D. Janz, and J. C. Wetherbe, "Key Issues in Information Systems Management" 1994-1995 SIM Delphi Results," *Management Information Systems Quarterly*, vol. 20, No. 2, pp. 225-242, 1996.
[11] P. P. Chen, "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transactions of Database Systems*, vol. 1, No. 1, pp. 9-36, 1976.
[12] F. A. D'Agnese, C. C. Faunt, A. K. Turner, and M. C. Hill, "Hydrogeologic evaluation and numerical simulation of the Death Valley Regional ground-water flow system, Nevada and California," U.S. Geological Survey Water Resources 96-4300, 1997.
[13] B. David, M. V. D. Herrewegen, and F. Salge, "Conceptual Models for Geometry and Quality of Geographic Information," in *Geographic Objects With Indeterminate Boundaries*, P. A. Burrough and A. Frank, Eds.: Taylor & Francis, 1996, pp. 352.
[14] C. E. Dyreson, W. S. Evans, H. Lin, and R. T. Snodgrass, "Efficiently Supporting Temporal Granularities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, No. 4, pp. 568-587, 2000.
[15] C. E. Dyreson and R. T. Snodgrass, "Supporting Valid-Time Indeterminacy," *ACM Transactions on Database Systems*, vol. 23, No. 1, pp. 1-57, 1998.
[16] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, Second ed. Redwood City, CA: Benjamin/ Cummings Publishing Co., 1994.
[17] R. Elmasri, G. Wuu, and V. Kouramajian, "A Temporal Model and Query Language for EER Databases," in *Temporal Databases: Theory, Design and Implementation*, A. Tansel, Ed.: Benjamin/ Cummings, 1993, pp. 212-229.
[18] R. Elmasri and G. T. J. Wuu, "The Temporal Model and Query Language for ER databases," 6th International Conference on Data Engineering, pp. 76-83, 1990.
[19] S. Ferg, "Modeling the Time Dimension in an Entity-Relationship Diagram," 4th International Conference on the Entity-Relationship Approach, pp. 280-286, 1985.
[20] P. N. Finlay and A. C. Mitchell, "Perceptions of the Benefits from Introduction of CASE: An Empirical Study," *MIS Quarterly*, vol. 18, No. 4, pp. 353-370, 1994.
[21] D. Finnigan, E. A. Kemp, and D. Mehandjiska, "Towards an Ideal CASE Tool," International Conference on Software Methods and Tools, pp. 189-197, 2000.
[22] A. U. Frank, "Spatial Concepts, Geometric Data Models, and Geometric Data Structures," *Computers and Geosciences*, vol. 18, No. 4, pp. 409-417, 1992.
[23] M. F. Goodchild, "Geographic Information Systems," in *Ten Geographic Ideas that Changed the World*, S. Hanson, Ed. New Brunswick, New Jersey: Rutgers University Press, 1997.
[24] H. Gregersen and C. Jensen, "Conceptual Modeling of Time-Varying Information," TIMECENTER Technical Report TR-35, September 10 1998.
[25] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, No. 2, pp. 199-220, 1993.
[26] O. Gunther and W. F. Riekert, "The Design of GODOT: An Object-Oriented Geographical Information System," *IEEE Data Engineering Bulletin*, vol. 16, No. 3, pp. 4-9, 1993.

[27] T. Hadzilacos and N. Tryfona, "An Extended Entity-Relationship Model for Geographic Applications," *SIGMOD Record*, vol. 26, No. 3, pp. 24-29, 1997.
[28] S. Jarzabek and R. Huang, "The Case for User-Centered CASE Tools," *Communications of the ACM*, vol. 41, No. 8, pp. 93-99, 1998.
[29] C. S. Jensen, C. E. Dyreson, M. Bohlen, J. Clifford, R. Elmasri, S. K. Gadia, F. Grandi, P. Hayes, S. Jajodia, W. Kafer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peresi, B. Pernici, J. F. Roddick, N. L. Sarda, M. R. Scalas, A. Segev, R. T. Snodgrass, M. D. Soo, A. Tansel, R. Tiberio, and G. Wiederhold, "A Consensus Glossary of Temporal Database Concepts-February 1998 Version," in *Temporal Databases: Research and Practice*, O. Etzion, S. Jajodia, and S. Sripada, Eds.: Springer-Verlag, 1998.
[30] C. S. Jensen and R. T. Snodgrass, "Semantics of Time-Varying Information," *Information Systems*, vol. 21, No. 4, pp. 311-352, 1996.
[31] C. S. Jensen and R. T. Snodgrass, "Semantics of Time-Varying Attributes and their use for Temporal Database Design," TimeCenter Technical Report, Tucson, Arizona TR-1, January 29 1997.
[32] C. S. Jensen and R. T. Snodgrass, "Temporal Data Management," *Transactions of Knowledge and Data Engineering*, vol. 11, No. 1, pp. 36-44, 1999.
[33] V. Khatri, "Bridging the Spatio-Temporal Semantic Gap: A Theoretical Framework, Evaluation and A Prototype System," in *Management Information Systems Department Dissertation*. Tucson, AZ: University of Arizona, 2002, pp. 320.
[34] V. Khatri, S. Ram, and R. T. Snodgrass, "Supporting User-defined Granularities and Indeterminacy in a Spatiotemporal Conceptual Model," *Annals of Mathematics and Artificial Intelligence*, vol. 36, No. 1-2, pp. 195-232, 2002.
[35] N. Kline, " An Update of the Temporal Database Bibliography," *SIGMOD Record*, vol. 22, No. 4, pp. 66-80, 1993.
[36] G. Kosters, B.-U. Pagel, and H.-W. Six, "Object Oriented Requirements Engineering for GIS-Applications," International Conference on ACM Geographical Information System, pp. 61-68, 1995.
[37] D. M. Mark and A. U. Frank, "Experiential and Formal Models of Geographic Space," *Environment and Planning*, vol. 23, No. 1, pp. 3-24, 1996.
[38] L. E. McKenzie, "Bibliography: Temporal Databases," *SIGMOD Record*, vol. 15, No. 4, pp. 40-52, 1986.
[39] J. L. Mennis, D. J. Peuquet, and L. Qian, "A Conceptual Framework for Incorporating Cognitive Principles into Geographical Database Representation," *International Journal of Geographic Information Science*, vol. 14, No. 6, pp. 501-520, 2000.
[40] OGIS, "OpenGIS Simple Feature Specification for SQL," Open GIS Consortium, Inc. 99-049, May 5 1999.
[41] Oracle, "Coordinate Systems User's Guide," Oracle Inc., Release 8.1.6 April 28 2000.
[42] Oracle, "Oracle8i Documentation, Release 8.1.6," Oracle, Accessed on, http://technet.oracle.com/docs/products/oracle8i/ doc_index.htm, December 1999.
[43] W. J. Orlikowski, "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development," *MIS Quarterly*, vol. 17, No. 3, pp. 309-340, 1993.
[44] C. Parent, S. Spaccapietra, and E. Zimanyi, "Spatio-temporal conceptual models: Data structures + space + time," 7th ACM Symposium on Advances in Geographic Information Systems, Kansas City, USA1999.
[45] J. Park, "Facilitating Interoperability among Heterogeneous Geographic Database Systems: A Theoretical Framework, A Prototype System and Evaluation," in *Management Information Systems Department Dissertation*. Tucson: University of Arizona, 1999, pp. 507.
[46] D. Parkes and N. Thrift, *Time, spaces and places*. New York: John Wiley & Sons, 1980.
[47] D. J. Peuquet, "It's about Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems," *Annals of the Association of American Geographers*, vol. 83, No. 3, pp. 441-461, 1994.

[48] D. Pfoser and N. Tryfona, "Requirements, Definitions, and Notations for Spatiotemporal Application Environments," 6th International Symposium on Advances in Geographic Information Systems, Washington, United States, pp. 124-130, 1998.
[49] S. Ram, "Intelligent Database Design using the Unifying Semantic Model," *Information and Management*, vol. 29, No. 4, pp. 191-206, 1995.
[50] P. Rigaux, M. O. Scholl, and A. Voisard, *Spatial Databases: With Application to GIS*: Morgan Kaufmann Publishers, 2001.
[51] D. Rumelhart, P. Lindsay, and D. Norman, "A process model for long-term memory," in *Organization of Memory*, E. Turving and W. Donaldson, Eds. New York: Academic Press, 1972.
[52] E. A. Rundensteiner, A. Bic, J. P. Gilbert, and M. Yin, "Set restrictions on semantic groupings," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, No. 2, pp. 193-204, 1994.
[53] S. Shekhar, M. Coyle, B. Goyal, D.-R. Liu, and S. Sarkar, "Data Models in Geographic Information Systems," *Communications of the ACM*, vol. 40, No. 4, pp. 103-111, 1997.
[54] A. Sheth, "Data Semantics: What, Where and How?," 6th IFIP Working Conference on Data Semantics (DS-6), Atlanta, Georgia, pp. 601-610, 1995.
[55] A. Silbershatz, H. Korth, and S. Sudarshan, *Database System Concepts*, Third Edition ed: WCB/ McGraw Hill, 1997.
[56] R. T. Snodgrass, "The Temporal Query Language TQuel," *ACM Transactions of Database Systems*, vol. 12, No. 2, pp. 247-298, 1987.
[57] R. T. Snodgrass, *The TSQL2 temporal query language*. Boston: Kluwer Academic Publishers, 1995.
[58] R. T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*. San Francisco: Morgan Kaufmann Series in Data Management Systems, 1999.
[59] R. T. Snodgrass and I. Ahn, "Temporal Databases," *IEEE Computer*, vol. 19, No. 9, pp. 35-42, 1986.
[60] R. T. Snodgrass, M. H. Böhlen, C. S. Jensen, and A. Steiner, "Adding Transaction Time to SQL/Temporal," ISO-ANSI SQL/Temporal Change Proposal, ANSI X3H2-96-152r ISO/IEC JTC1/SC21/WG3 DBL MCI-143, 1996.
[61] R. T. Snodgrass, M. H. Böhlen, C. S. Jensen, and A. Steiner, "Adding Valid Time to SQL/Temporal," ISO-ANSI SQL/Temporal Change Proposal, ANSI X3H2-96-151r ISO/IEC JTC1/SC21/WG3 DBL MCI-142, 1996.
[62] R. T. Snodgrass, M. H. Böhlen, C. S. Jensen, and A. Steiner, "Transitioning Temporal Support in TSQL2 to SQL3," in *Temporal Databases: Research and Practice*, O. Etzion, S. Jajodia, and S. M. Sripada, Eds. New York: Springer Verlag, 1997, pp. 150-194.
[63] M. D. Soo, "Bibliography on Temporal Databases," *SIGMOD Record*, vol. 20, No. 1, pp. 14-23, 1991.
[64] R. B. Stam and R. T. Snodgrass, "A Bibliography on Temporal Databases," *Data Engineering Bulletin*, vol. 11, No. 4, pp. 53-61, 1988.
[65] C. Theodoulidis, P. Loucupoulos, and B. Wangler, "A Conceptual Modeling Formalism for Temporal Database Applications," *Information Systems*, vol. 16, No. 4, pp. 401-416, 1991.
[66] C. Theodoulidis, B. Wanglet, and P. Loucopoulos, "The Entity Relationship Time Model," in *Conceptual Modeling, Databases and CASE: An Integrated View of Information Systems Development*. New York: John Wiley & Sons, 1992, pp. 87-115.
[67] M. Todd, R. L. Coleman, and J. M. Shimonek, "A CASE Perspective of System-Development: Woodmen Accident and Centel," *Journal of Systems Management*, vol. 42, No. 9, pp. 13-16, 1991.
[68] N. Tryfona and C. S. Jensen, "Conceptual Data Modeling for Spatiotemporal Applications," *Geoinformatica*, vol. 3, No. 3, pp. 245-268, 1999.
[69] V. J. Tsotras and A. Kumar, "Temporal Database Bibliography Update," *SIGMOD Record*, vol. 25, No. 1, pp. 41-51, 1996.
[70] J. W. van Roessel, "Design of a Spatial Data Structure using the Relational Normal Form," *International Journal of Geographical Information Systems*, vol. 1, No. 1, pp. 33-50, 1987.

[71] M. Wachowicz, *Object-Oriented Design for Temporal GIS*: Taylor and Francis, 1999.
[72] Y. Wand, D. E. Monarchi, J. Parsons, and C. C. Woo, "Theoretical Foundations for Conceptual Modeling in Information Systems Development," *Decision Support Systems*, vol. 15, No. 4, pp. 285-304, 1995.
[73] Y. Wand, V. C. Storey, and R. Weber, "On Ontological Analysis of the Relationship Construct in Conceptual Modeling," *ACM Transactions of Database Systems*, vol. 24, No. 4, pp. 494-528, 1999.
[74] W. A. Woods, "What's in a link: Foundations for semantic networks," in *Representation and understanding : studies in cognitive science*, D. G. Bobrow and A. Collins, Eds. New York: Academic Press, 1975, pp. 35-82.
[75] M. F. Worboys, "Computation with Imprecise Geospatial Data," *Computer, Environment and Urban Systems*, vol. 22, No. 2, pp. 85-106, 1998.
[76] M. F. Worboys, "Imprecision in Finite Resolution Spatial Data," *GeoInformatica*, vol. 2, No. 3, pp. 257-279, 1998.
[77] M. F. Worboys, H. M. Hearshaw, and D. J. Maguire, "Object-oriented data modeling for spatial databases," *International Journal of Geographical Information Systems*, vol. 4, No. 4, pp. 369-383, 1990.
[78] Y. Wu, S. Jajodia, and X. S. Wang, "Temporal Database Bibliography Update," in *Temporal Databases: Research and Practice*, O. Etzion, S. Jajodia, and S. M. Sripada, Eds.: Springer, 1997, pp. 338-366.

Appendix: Annotation Syntax in BNF

⟨annotation⟩ ::= ϵ | ⟨temporal annotation⟩ **/ /** ⟨spatial annotation⟩
| ⟨temporal annotation⟩ **/ /** ⟨spatial annotation⟩ **/ /** ⟨time-varying spatial annotation⟩

⟨temporal annotation⟩ ::= ϵ | ⟨valid time⟩ / ⟨transaction time⟩
⟨valid time⟩ ::= ⟨state⟩ (⟨g_t⟩) | ⟨indeterminate state⟩ (⟨g_t⟩) | ⟨event⟩ (⟨g_t⟩) | ⟨indeterminate event⟩(⟨g_t⟩) | -
⟨transaction time⟩ ::= T | -
⟨state⟩ ::= S | State
⟨indeterminate state⟩ ::= ⟨state⟩~ | ⟨state⟩+-
⟨event⟩ ::= E | Event
⟨indeterminate event⟩ ::= ⟨event⟩~ | ⟨event⟩+-

⟨spatial annotation⟩ ::= ϵ | ⟨horizontal geometry⟩ / ⟨vertical geometry⟩
⟨horizontal geometry⟩ ::= ⟨geometry⟩ (⟨g_{xy}⟩) / ⟨geometry⟩ (⟨g_{xy}⟩)
⟨vertical geometry⟩ ::= ⟨geometry⟩ (⟨g_z⟩) | -
⟨geometry⟩ ::= ⟨point⟩ | ⟨indeterminate point⟩ | ⟨line⟩ | ⟨indeterminate line⟩ | ⟨region⟩
| ⟨indeterminate region⟩ | ⟨user defined⟩ | -
⟨point⟩ ::= P | Point
⟨indeterminate point⟩ ::= ⟨point⟩~ | ⟨point⟩+-
⟨line⟩ ::= L | Line
⟨indeterminate line⟩ ::= ⟨line⟩~ | ⟨line⟩+-
⟨region⟩ ::= R | Region
⟨indeterminate region⟩ ::= ⟨region⟩~ | ⟨region⟩+-

⟨time-varying spatial annotation⟩ ::= ϵ | ⟨position varying⟩ | ⟨shape varying⟩ | ⟨position varying⟩ / ⟨shape varying⟩
⟨position varying⟩ ::= ⟨position⟩@⟨varying in dimension⟩
⟨shape varying⟩ ::= ⟨shape⟩@⟨varying in dimension⟩
⟨position⟩ ::= Pos | Position
⟨shape⟩ ::= Sh | Shape
⟨varying in dimension⟩ ::= x | y | z | xy | yz | xz | xyz

⟨g_t⟩ ::= ⟨day⟩ | ⟨hour⟩ | ⟨minute⟩ | ⟨second⟩ | ⟨user defined⟩
⟨day⟩ ::= day
⟨hour⟩ ::= hr | hour
⟨minute⟩ ::= min | minute
⟨second⟩ ::= sec | second
⟨g_{sxy}⟩ ::= ⟨mile⟩ | ⟨dms-degree⟩ | ⟨dms-minute⟩ | ⟨foot⟩ | ⟨user defined⟩
⟨g_{sz}⟩ ::= ⟨mile⟩ | ⟨foot⟩ | ⟨user defined⟩
⟨mile⟩ ::= mile
⟨dms-degree⟩ ::= dms-deg | dms-degree
⟨dms-minute⟩ ::= dms-min | dms-minute
⟨foot⟩ ::= ft | foot