Differentiating Telic-Atelic Data Semantics in Conceptual Design

Differentiating Telic-Atelic Data Semantics in Conceptual Design

Vijay Khatri, Operations & Decision Technologies, Indiana University, Bloomington IN 47405
Sudha Ram, Management Information Systems, University of Arizona, Tucson AZ 85721
Richard T. Snodgrass, Computer Science, University of Arizona, Tucson AZ 85721
Paolo Terenziani, Computer Science, Universita degli Studi di Torino, Italy

Abstract: Time provides context for all our experiences, cognition, and coordinated collective action. Prior research in linguistics, AI and temporal databases suggests the need to differentiate between temporal facts with goal-related semantics (i.e., *telic*) from those are intrinsically devoid of culmination (i.e., *atelic*). To differentiate between telic/atelic data semantics in conceptual database design, we apply an annotation-based approach to the Unifying Semantic Model—a conventional conceptual model—to propose an annotation-based temporal conceptual model. Our temporal conceptual design approach involves: 1) capturing "what" semantics using a conventional conceptual model; 2) employing annotations to differentiate between telic/atelic data semantics that help capture "when" semantics; 3) specifying temporal constraints, specifically non-sequenced semantics, in the temporal data dictionary as metadata.

Our proposed approach provides a mechanism to "wrap" telic/atelic temporal semantics using temporal annotations. We also show how these semantics can be "unwrapped" using constructs of the conventional conceptual model and axioms in first-order logic. Via what we refer to as the "semantics of composition," i.e., semantics implied by the interaction of annotations, we illustrate the consequences of explicating telic/atelic data semantics during temporal conceptual design. Because the formally-defined (using BNF) temporal annotations (i.e., when) are orthogonal to entity classes, attributes and/or relationships (i.e., what), the proposed approach requires minimal updates to the conventional conceptual model. The annotation-based approach is not only more expressive and provides a mechanism to capture telic/atelic semantics but is also upward-compatible, i.e., it does not invalidate the extant (non-historical) legacy schemas, thereby, protecting the investments in existing schemas.

Differentiating Telic-Atelic Data Semantics in Conceptual Design[1]

Vijay Khatri, Operations & Decision Technologies, Indiana University, Bloomington IN 47405
Sudha Ram, Management Information Systems, University of Arizona, Tucson AZ 85721
Richard T. Snodgrass, Computer Science, University of Arizona, Tucson AZ 85721
Paolo Terenziani, Computer Science, Universita degli Studi di Torino, Italy

1. Introduction

The differentiation between telic and atelic data semantics—dating back to Aristotle—is rooted in linguistics (Vendler 1967) and cognitive science (Bloom et al. 1980). Based on temporal properties, linguists classify natural language sentences into different aktionsart classes (also called *aspectual classes*), thus, distinguishing between facts with goal-related semantics (referred to as *telic*) from those that are devoid of culmination (referred to as *atelic*). In this paper, we suggest that the telic/atelic distinction is important in designing *temporal applications* such as accounting, portfolio management, personnel management, and inventory management.

Conceptual database design is widely recognized as an important step in the development of database applications (Batini et al. 1992; Elmasri and Navathe 2006; Silbershatz et al. 1997) including *temporal applications*, i.e., database applications that need to organize data by time, described above. During conceptual database design, a *conceptual model* provides a notation and formalism that can be used to construct a high-level representation of the real world—referred to as a *conceptual schema*—independent of the implementation details. Since data semantics provides a "connection from a database to the real world outside the database" (Sheth 1995) and a conceptual model provides a mechanism to capture the data semantics, data semantics that are captured during conceptual database design provide a mapping from a representation (i.e., conceptual schema) to some concepts in the "real world" and is the basis of any information system. A 2001 CSC survey found that organizing and utilizing data is one of the top-three issues for organizations worldwide (CSC 2001). Although conceptual modeling of data-related requirements is a small phase within the overall organization and utilization of data, it has a greater impact than any other phase (Witt and Simsion 2004) and impacts system development costs, system flexibility, and the ability to meet users' requirements (Moody et al. 1998).

---

[1] This work was supported in part by the National Science Foundation under grants IIS-0415101 and IIS-0455993, EIA-0080123 and by partial support from a grant from Microsoft Corporation.

Extant approaches to conceptual design of temporal applications (see, for example, Gregersen and Jensen 1998; Khatri et al. 2004) do not provide a mechanism to distinguish between *telic* and *atelic* data semantics. While prior research (Terenziani and Snodgrass 2004) has examined the implications of the telic/atelic dichotomy on the temporal aspects of the logical (specifically, relational) data model and algebra, this dichotomy has until now not been considered within the conceptual model. Other work (Khatri et al. 2004; Khatri et al. 2002) has considered conceptual modeling for temporal and geospatial data and applied an annotation-based approach to the *U*nifying *S*emantic *M*odel (USM) (Ram 1995)—an extended version of the *E*ntity-*R*elationship (ER) Model (Chen 1976)—to propose the geo*S*patio-*T*emporal *U*nifying *S*emantic *M*odel (ST USM). That work likewise did not consider the telic/atelic dichotomy. By proposing an approach to help capture telic/atelic data semantics for temporal applications, this paper extends prior research in temporal conceptual design. To the best of our knowledge, this is the first work to incorporate this dichotomy in a conceptual model.

To delineate the scope of our work, we state our assumptions.

- In this research we employ annotations to capture telic/atelic data semantics. The objective of this research is not syntax (e.g., textual annotations), but rather temporal semantics that need to be captured during conceptual design. Prior research in temporal conceptual design has presented several different ways of rendering temporal data semantics via annotations. For example, temporal data semantics can be represented textually (see, for example, Khatri et al. 2004) or graphically (see, for example, Zimanyi et al. 1997); they can be represented in the schema (see, for example, Khatri et al. 2004) or outside the schema (see, for example, Snodgrass 1999). However, the objective of this research is to explicate the telic and atelic data semantics and not delve into how "best" to syntactically render the telic/atelic data semantics.
- In our research on conceptual modeling, we also do not focus on how semantics are *physically* implemented, as that is an involved topic in its own right and has been considered elsewhere in the context of the relational model (Terenziani and Snodgrass 2004).
- Facts can interact with time in two orthogonal ways resulting in transaction time and valid time (Snodgrass and Ahn 1986). While *valid time* denotes when the fact is true in the real world and implies the storage of histories related to facts, *transaction time* links an object to the time it is current in the database and implies the storage of versions of a database object (Jensen et al. 1998). In this paper, we

focus only on valid time because transaction time is intrinsically atelic.

- While database schema can evolve with time and *schema versioning* (Roddick 1995; Roddick and Snodgrass 1995) is an important area of research, we do not focus on schema versioning. Instead, we focus on the semantics of data, as captured in a single schema.
- In this work we differentiate between *data* and *query semantics*, and focus on the former. Note that data has its own semantics—independently of any query language and query operators—and that queries are merely an operational way of making such semantics explicit. However, data has semantics even if *no* query is asked. Because conceptual models provide a mechanism to capture data semantics, we focus on capturing (telic/atelic) data semantics.

In summary, this paper focuses on differentiating telic/atelic data semantics during conceptual modeling and provides an approach (more accurately, annotation-based approach) to capture telic data semantics in a temporal conceptual model.

The rest of this paper is organized as follows. The following section first provides a context for this research in areas such as linguistics, artificial intelligence and temporal databases, and then motivates the need for a temporal conceptual model. In Section 3, we motivate the specific research question addressed in this paper through an example that illustrates the differences between telic and atelic data semantics. In Section 4, we present our proposed annotation-based approach for capturing the telic-atelic dichotomy in a conceptual model; this approach focuses first on "what" (e.g., using USM) is important for an application in the real world, then considers "when" semantics, and finally captures the temporal constraints. The basis of a modeling language is the semantics (or meaning) of its constructs; Section 5 formally defines the meaning of telic/atelic temporal entity classes, attributes, and relationships. We further describe the semantics of composition, i.e., the semantics implied by the interaction of annotations, and provide a mechanism to capture temporal constraints. We evaluate our proposed approach in Section 6 and conclude by highlighting the contributions of this research.

2. Background

After providing the context for telic/atelic data semantics in linguistics, artificial intelligence and temporal databases, we describe problems in capturing temporal semantics during "conventional" conceptual design.

2.1 Telic/Atelic Distinction

The distinction between *telic* and *atelic* dates back to Aristotle, who first noticed that facts can be partitioned into two main classes depending on whether they are goal-oriented (an example of a telic fact is "Bob built a house;" *telos* means goal in Greek) or not (an example of an atelic fact is "Bob is asleep;" in Greek "*a*" is used as a prefix to denote negation). Further, such a distinction has played a major role in areas such as linguistics, cognitive science, and lately, artificial intelligence and temporal databases, as we now review.

2.1.1 Linguistics

As a useable communication medium, language is generally thought of as the raw material from which data is created. The distinction between telic and atelic facts has been widely explored in linguistics (see, for example, Webber 1988) and cognitive science (see, for example, Bloom et al. 1980). Within the linguistic community, it is commonly agreed that sentences can be classified into different *aktionsart* classes (also called *aspectual* classes (Webber 1988)) depending on their linguistic behavior or their semantic properties. The ontologic basis of this classification into stative (e.g., "built a house") and kinesis (e.g., "asleep") sentences is rooted in causation and consequence: for a kinesis sentence, there is a "culmination" after which a consequent state ensues (Moens and Steedman 1988). While in the example of "built a house," there is a clearly defined "culmination," that in the example of "asleep" is not. Prior research employs the following semantic criteria to distinguish between stative (or atelic) and kinesis (or telic) statements (Dowty 1986):

- A sentence φ is stative iff it follows from the truth of φ at an interval I that φ is true at all subintervals of I, e.g., if Bob was asleep from 1:00 to 2:00 PM, then he was asleep at all subintervals of this interval; asleep is, thus, stative. [2]
- A sentence φ is kinesis iff it follows from the truth of φ at an interval I that φ is false at all subintervals of I, e.g., if Bob built a house from June 1 to September 1, then it is false that he built a house in any subinterval of this interval; built a house is, thus, kinesis.

The linguistic community agrees that although all base facts can be classified as telic/atelic, a telic-to-atelic

[2] This property is referred to as *downward inheritance* by the AI community (Shoham 1987); see also the next section.

(or atelic-to-telic) coercion can always be performed using explicit linguistic tools; e.g., a present perfect sentence (Verkuyl 1972) can always be converted into a progressive form (cf., Dowty 1986; Moens and Steedman 1988; Vendler 1967). For instance, although "Bob built a house from June 1 to September 1" is a telic fact and one cannot infer that "Bob built a house on July 1," one can correctly assert that "Bob was building a house on July 1" because a progressive form has been used in the telic-to-atelic coercion.

2.1.2 Artificial Intelligence

Since "one of the most crucial problems in any computer system that involves representing the world is the representation of time" (Allen 1991), the treatment of the telic/atelic dichotomy has had a significant impact on the artificial intelligence (AI) literature. While the ontologic basis for the telic/atelic distinction is rooted in causation and consequence, the AI literature (see, for example, Shoham 1987) distinguishes between telic/atelic facts on the basis of *upward* and *downward inheritance*; note that upward inheritance has been adapted from the linguistic literature (property of stative sentence) described in the prior section:

- The *downward inheritance* property implies that one can infer from the temporal fact *f* that holds at valid time *t* (where *t* is a time period) that *t* holds in any sub-period (and sub-point) of *t*.
- The *upward inheritance* property implies that one can infer from the temporal data *f* that holds at two consecutive or overlapping time periods t_1 and t_2 that *f* holds in the union time period $t_1 \cup t_2$.

While atelic facts exhibit upward and downward inheritance, telic facts do not.

In AI, the telic/atelic dichotomy (using a different terminology) was first explicitly dealt with in Allen's reified logic (Allen 1984), which models a general formal ontology to deal with time and causation and includes the famous Interval Algebra (Allen 1983); note that Allen's seminal Interval Algebra is a subpart of Allen's general logical framework.[3] Recently, AI approaches in the development of formal ontologies (Guarino and Poli 1995) pay specific attention to the telic/atelic dichotomy. A relevant example is the ontology devised within the CYC project, which aims at encoding "the hundreds of millions of facts and heuristics that comprise human consensus reality" (Lenat and Guha 1994).

---

[3] In AI, there is a long and still ongoing debate on whether it is better to model reality as a sequence of different *states* (*atelic*-based representation; consider, for example, the Situation Calculus (McCarthy 1968)), or as a sequence of different *events* (*telic*-based representation; consider, for example, the Event Calculus (Kowalski and Sergot 1986)). A discussion on the relative merits of the two approaches would lead us far away the main goals of this paper. Following Allen's seminal approach (Allen 1984), most AI researchers generally agree that state- and event-based representation of reality is complementary, and that one needs a *flexible* approach in which *both ways* can be adopted (see, for example, Allen 1984; Franconi et al. 1993; Galton 1991; Terenziani and Torasso 1994).

On the other hand, the treatment of the telic/atelic dichotomy has been considered by the database community only recently.

2.1.3 Temporal Databases

Current database management systems (DBMSs) offer little built-in query language support for temporal data management other than that for user-defined time.[4] The change proposal to SQL3 that includes temporal semantics is referred to as SQL/Temporal (Snodgrass et al. 1996a; Snodgrass et al. 1996b); recently, the telic/atelic distinction has been defined in a relational data model and algebra (Terenziani and Snodgrass 2004).

Using an example, we summarize the approach proposed in Terenziani and Snodgrass (2004). Figure 1 illustrates an example of a relation CONTRACT that includes attributes such as ID, task, budget and the time over which the contract was accomplished (represented by two columns, start VT to provide the beginning valid time, and end VT) with year as the temporal granularity. From a linguistic perspective, the data associated with contract `C1` is telic because there is a "culmination" when the contract is accomplished, e.g., 2006 for the contract `C1`. Based on the AI/database operationalization of telic facts, upward and downward inheritance would not apply for contracts. For example, it is not the case that contract `C1` was accomplished in the year 2005 (downward inheritance). It is also not the case that the projection[5] of task, start VT, and end VT would suggest that there is a single contract for `Web design` for three years from 2004 to 2006 (upward inheritance). Note that erroneous application of upward inheritance to telic facts can cause *a non-recoverable loss of information* since different facts (with the same values for non-temporal attributes) would be erroneously "merged together."

Figure 1 also illustrates how ascribing telic or atelic semantics to the lifespan of a CONTRACT affects the "meaning" of the lifespan of a CONTRACT. Ascribing atelic data semantics (see Figure 1a) to the relation CONTRACT would imply that contracts that were accomplished in 2004-2005 were `C1`, `C2` and `C3`, while ascribing telic data semantics (see Figure 1b) would imply that only `C1` was accomplished in that time period.

As illustrated in Figure 1a, prior research (see, for example, Terenziani and Snodgrass 2004) suggests

---

[4] An uninterpreted time value is referred to as *user-defined time* (Snodgrass 1999).
[5] The *projection* operation selects certain columns from the table and discards others (Elmasri and Navathe 2006).

that temporal database approaches have been semantically "point-based," that is, they effectively (if not within the representation) associate time points to facts; as such prior approaches treat all temporal facts as atelic only. Recently, a two-sorted data model has been proposed in which telic data can be stored in telic relations (which are relations *not* supporting the upward and downward inheritance properties) and atelic data in atelic relations (relations supporting the upward and downward inheritance properties) (Terenziani and Snodgrass 2004). At the level of relational algebra, this approach also provides the linguistic flexibility to switch from a telic to an atelic view, and vice versa. While stored data is either telic or atelic, additional telic-to-atelic (or atelic-to-telic) coercion functions, which are the counterpart of linguistic coercion operators (see the discussion in 2.1.1), have been provided *at the query level*. (As the present paper is concerned with conceptual design and thus with the telic-atelic dichotomy within stored data, we do not consider such query coercion functions further.)

ID	Task	budget	start VT	end VT
C1	SAP implementation	500,000	2002	2006
C2	Web design	20,000	2004	2005
C3	Web design	75,000	2005	2006

CONTRACT

a) **Ascribe atelic semantics to the lifespan of CONTRACT:**
2002 → <C1, SAP implementation, 500,000>
2003 → <C1, SAP implementation, 500,000>
2004 → <C1, SAP implementation, 500,000>
2004 → <C2, Web design, 20,000>
2005 → <C1, SAP implementation, 500,000>
2005 → <C2, Web design, 20,000>
2005 → <C3, Web design, 75,000>
2006 → <C1, SAP implementation, 500,000>
2006 → <C3, Web design, 75,000>

b) **Ascribe telic semantics to the lifespan of CONTRACT:**
[2002-2006] → <C1, SAP implementation, 500,000>
[2004-2005] → <C2, Web design, 20,000>
[2005-2006] → <C3, Web design, 75,000>

Figure 1: Example of temporal semantics for a CONTRACT relation

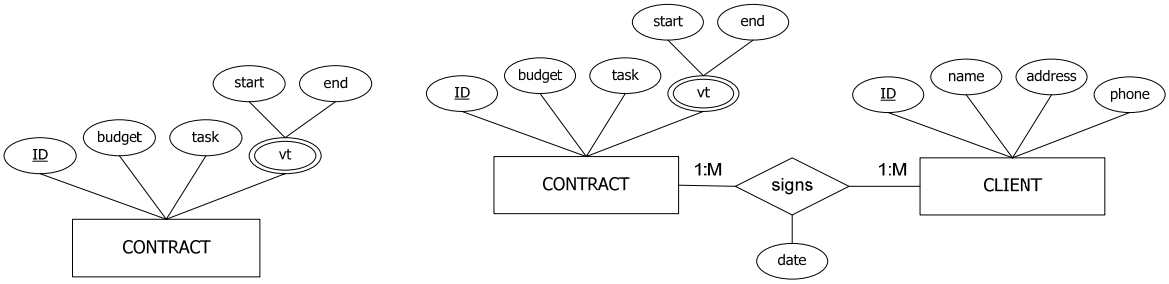
Considering the significance of temporal data semantics (including telic/atelic) in a logical model, we continue with our example of contracts to illustrate the problems with capturing these data semantics during "conventional" conceptual design.

2.2 Capturing Temporal Data Semantics during Conceptual Design

To motivate the need for a temporal conceptual model, let us continue with the example above. Let us

assume that we want to capture the lifespan when a CONTRACT exists; see Figure 2a. Besides other attributes of CONTRACT such as ID (identifier shown by underlining the attribute), task and budget, the only way in which one can capture the existence of a CONTRACT in a conventional conceptual model (e.g., the ER Model (Chen 1976), USM (Ram 1995)) is by using a multi-valued composite attribute vt, with component attributes start and end. Since a conventional conceptual model does not provide a mechanism to capture the telic semantics of vt, the database analyst is left to discover, design and implement these semantics in an *ad-hoc* manner. For example, because the semantics of vt are "not" captured, a user of the database could enter the start and end value (of vt) as "`Jan 5 2007`" and "`Jan 2 2007`," respectively, i.e., end.vt < start.vt, which is conceptually impossible. Such database values do not represent those in the "real world," i.e., the database will not be a reflection of the "real world."

Let us continue with the CONTRACT example; see Figure 2b. Let us assume that the application requires capturing the relationship between CONTRACT and CLIENT; CLIENT has attributes such as ID, name, address and phone. To capture the date on which a CLIENT signs a CONTRACT, a date attribute is added to the relationship signs. However, such a schema has unclear semantics: Do the cardinalities imply that a CLIENT can sign multiple CONTRACTs on a certain date or that a CLIENT can sign multiple CONTRACTs during the lifespan of the CONTRACT?



a) CONTRACT b) CONTRACT having a relationship (signs) with CLIENT

Figure 2: Two examples of a conceptual schema with support for user-defined time only

In summary, attributes such as date, start.vt,,and end.vt (in Figure 2) do not adequately capture temporal data semantics. Because conventional conceptual models (e.g., the ER Model (Chen 1976) and USM (Ram 1995)) do not provide a mechanism to capture temporal data semantics, temporal conceptual models have been proposed; the readers are referred to Gregersen and Jensen (1999) for a survey on temporal conceptual models.

3. Desiderata

A precursor to augmenting a conventional conceptual model is identifying the conceptual modeling requirements that need to be met. We first present an example that motivates the need to capture telic/atelic data semantics. Next, we outline criteria for augmenting a conventional conceptual model.

3.1 Motivating Example

To motivate this research, we provide an example below. As shown in Figure 3, let us assume that a project has three contracts, `C1`, `C2` and `C3`, that are accomplished in time `[2002-2006]`, `[2004-2005]`, and `[2005-2006]`, respectively. Similarly, let us assume that three contractors, `CTOR1`, `CTOR2` and `CTOR3`, exist over time {`2002`, `2003`, `2004`, `2005`, `2006`}, {`2004`, `2005`}, and {`2005`, `2006`}, respectively. Contracts are *telic* facts, which involve a specific goal or culmination and do not satisfy the property of upward and downward inheritance. Thus, if a contract (say `C1`) has been accomplished in a given period of time [`2002-2006`], it cannot be inferred that it has been accomplished in any other time period (say, `[2004-2005]`); that is, the contract `C1` has been accomplished in the whole period starting from 2002 and ending in 2006, and in no other period. On the other hand, existence of contractors is construed as atelic, since it does not involve any specific goal or culmination and satisfies the property of upward and downward inheritance. As a consequence, if a contractor existed for a period of time, s/he also existed in the sub-parts of the time period. For example, while `CTOR1` existed as a contractor from 2002-2006, one can appropriately infer that s/he existed (as a contractor) during 2004-2005.

For compliance purposes, let us assume that the organization wants to audit only the contracts that were accomplished during the time period 2004-2005; similarly, let us assume that the same organization wants to audit the contractors that existed during the time period 2004-2005. (See the shaded region in Figure 3.)

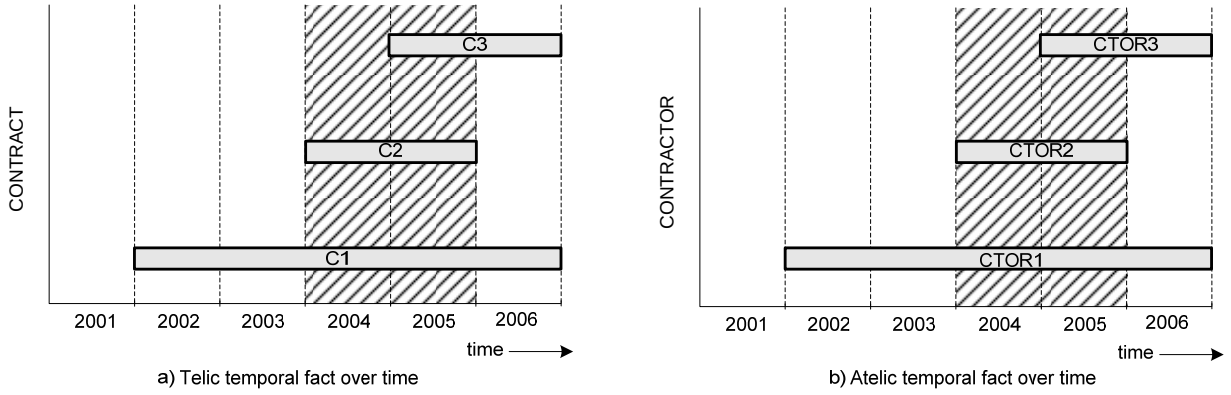


Figure 3: Motivating example of telic and atelic facts

Based on telic/atelic interpretation of the modeled data, Table 1 shows the results of the contracts/contractors that should be audited. As is evident from the presented results, there would be significant problems if there is no mechanism to differentiate between telic and atelic facts. For example, if all facts are interpreted as telic (row 1 of Table 1), then two contractors (`CTOR1` and `CTOR3`) will *not* be erroneously audited; such oversight has legal ramifications for the organization. On the other hand, if all facts are interpreted as atelic (row 2 of Table 1), then two contracts (`C1` and `C3`) will be erroneously audited, thus, wasting resources in the organization.

	Contracts that were accomplished in the time period `2004-2005`	Contractors that existed in the time period `2004-2005`
Ascribe Telic Semantics	`C2` (*correct*)	`CTOR2` (*incorrect*)
Ascribe Atelic Semantics	`C1`, `C2`, `C3` (*incorrect*)	`CTOR1`, `CTOR2`, `CTOR3` (*correct*)

Table 1: Audit results based on telic and atelic interpretation of data

Considering that telic/atelic semantics need to be captured during temporal conceptual design, we next describe criteria for augmenting a conventional conceptual model.

3.2 Evaluation Criteria

Prior research suggests that conceptual models should possess qualities such as expressiveness, simplicity, minimality and formality (Batini et al. 1992). While the availability of a large variety of constructs for a comprehensive representation of the "real world" ensures *expressiveness* (see also Wand and Weber 1993), understandability that is based on human cognition (see, for example, Bodart et al. 2001; Kim et al. 2000; Parsons and Wand 1997; Siau and Tan 2005) enables *simplicity*. *Minimality* implies that no concept can be

expressed through composition of other concepts. *Formality* ensures that each construct of the model presents a unique, precise and well-defined interpretation. Because annotation-based temporal conceptual models (see, for example, Gregersen and Jensen 1998; Khatri et al. 2004; Zimanyi et al. 1997) require "augmenting" a conventional conceptual model, another requirement that is pertinent is upward compatibility. *Upward compatibility* (Böhlen et al. 2000) refers to the ability to render a conventional conceptual model temporal without impacting or negating the semantics of the non-historical "legacy schema," i.e., the schema developed using a conventional conceptual model; such a property of a temporal conceptual model protects investments in non-historical existing (or legacy) schemas. Upward compatibility requires that with the addition of temporal aspects, the syntax and semantics of the conventional conceptual model (for example, the ER Model (Chen 1976)) remain unaltered.

4. Annotation-based Approach

Prior research in *temporal conceptual design* divides temporal conceptual modeling into two phases: 1) first capture the "current" reality using a conventional conceptual model without considering the temporal aspects; and subsequently 2) annotate the schema with the temporal data semantics of the application (Khatri et al. 2004; Khatri et al. 2002). Based on human associative memory (see, Anderson and Bower 1973) that segregates "what" from "when," the annotation-based temporal conceptual design approach employs the generic problem solving approach of "divide and conquer." Instead of using additional constructs to represent the temporal aspects, this approach uses annotations to represent "when" on top of the initial abstraction, which represents "what." In the second phase, we suggest that a database analyst needs to distinguish between telic and atelic data semantics because, as we saw above, an inability to do so will lead to a database that is not an accurate reflection of the "real world."

In this paper, we adapt the annotation-based approach exemplified via ST USM (Khatri et al. 2004; Khatri et al. 2002). Via annotations, we provide a means to differentiate between telic and atelic data semantics. While prior research (Khatri et al. 2004; Khatri et al. 2002) employed annotations to help capture atelic semantics, in this paper we extend the annotations to include telic semantics.

Our overall approach and guiding principle is schematically shown in Figure 4 below. USM—a conventional conceptual model that helps to capture the data semantics related to entity classes, attributes and relationships—instantiates the non-temporal data semantics (cf. Section 4.1). Annotations are used to

instantiate both telic and atelic data semantics (cf. Section 4.2 for syntax and Section 5 for semantics). Finally, we suggest that temporal non-sequenced constraints, which do not treat time specially or reference all points of time, need to be captured in a temporal data dictionary (that is, as metadata).

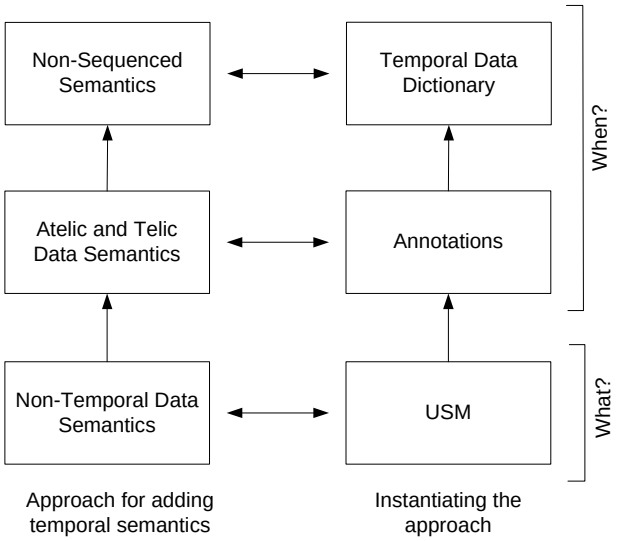


Figure 4: Inducing temporal data semantics

4.1 The Unifying Semantic Model

We summarize below how to represent "what" data semantics that can be captured using a conventional conceptual model (see, for example, Batini et al. 1992; Chen 1976; Ram 1995; Silbershatz et al. 1997), specifically USM (Ram 1995), utilizing the contract application (outlined in the prior sections) as an example.

USM includes concepts such as entity class, attribute, and relationship. While the representation of real world objects are referred to by the term *entities*, their characteristics are referred to as *attributes*. While an *entity class*, sometimes referred to as an *entity type*, is a collection of entities that have the same attributes (Elmasri and Navathe 2006), the set of instantiations of an entity class is referred to as an *entity set*. So, $E(e)$ represents an entity e of an entity class E and a set of entities of an entity class is represented as $S(E)$. As shown in Figure 5, CONTRACTOR is an entity class with attributes such as ID, name, and phone. A relationship connects members of one entity class to members of one or more entity classes. For example, executes is a relationship between CONTRACTOR and CONTRACT. The cardinalities on the schema imply that, for example, a CONTRACTOR executes a minimum of 0 and a maximum of many (0:M) CONTRACTs and that a given CONTRACT is executed by exactly one CONTRACTOR (1:1).

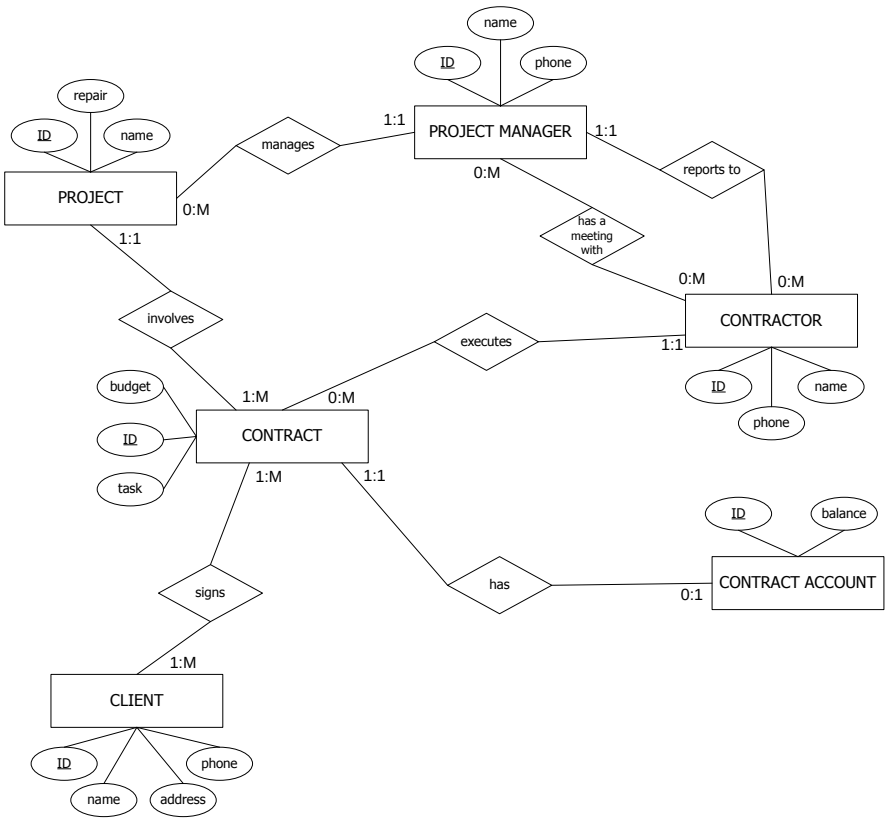


Figure 5: USM Schema for a contract application

Figure 5 shows a USM schema for our application, which includes entity classes, attributes and relationships. At this point, because the temporal aspects have not yet been considered, the schema may be referred to as non-temporal. We will later extend this schema to a temporal USM schema. But before doing so, we need to describe the basis for telic/atelic data semantics and the annotations that are employed to capture these semantics.

4.2 Associating Facts with Time

According to Klein (1994), "each finite verb obligatorily includes temporal information—it expresses tense, aspect, or both…the expression of time is a consequence of the way in which languages is structured," thus, suggesting that "all" objects, their properties and relationships are embedded in time. In the following, we summarize extant definitions associated with how facts can interact with time (Jensen et al. 1998; Snodgrass and Ahn 1986; Snodgrass et al. 1996a; Snodgrass et al. 1996b). Next, we continue the motivating example to show the need for differentiating between telic and atelic data semantics. We present an enhanced annotation syntax that would help capture the telic data semantics and then apply the

annotation-based approach to our motivating example.

4.2.1 Temporal Primitives

The pair (T, ≤) is used to denote *time domain*, 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 (Jensen and Snodgrass 1997). For example, a discrete time domain is represented by (**Z**, ≤) where instants are isomorphic to integers, implying that every instant has a unique successor.[6] Additionally, time is assumed to be bounded at both ends, i.e., the past and the future (Snodgrass 1995).

The time between two instants, e.g., from Jaunary 1, 2008 to May 31, 2008, is referred to as a *time period*. An unanchored contiguous portion of the time line, e.g., one day, is called a *time interval*. An interval is relative while an instant is absolute (Snodgrass 1999). A non-decomposable time interval of fixed minimal duration, say, microsecond, is referred to as a *chronon*. *A temporal granularity*, a measure of the time datum (Bettini et al. 1998; Bettini et al. 2000), is intrinsic to temporal data and provides a mechanism to hide details that are not known or not pertinent for an application. Some examples of temporal granularities are Gregorian day (or day) and business week.

4.2.2 The Telic/Atelic Distinction

For valid time, prior research has operationalized the presence or absence of goal-related semantics via the properties of upward and downward inheritance (Terenziani and Snodgrass 2004). Consistent with prior database literature, we define *atelic facts* as ones that are characterized by properties of downward and upward inheritance. On the other hand, *telic facts* are defined as facts for which neither property holds. While upward and downward inheritance holds for the lifespan of a CONTRACTOR that for CONTRACT does not (in Figure 5); thus, the former is atelic and the latter is telic. In summary, whereas time periods are primitive non-decomposable units for telic valid time, those for atelic valid time are time points.

The telic/atelic dichotomy (and, in particular, the issues on whether upward and downward inheritance hold or not) seems irrelevant in the context of instantaneous facts (usually called *achievements* in the

[6] An *instant* is a time point on the time line.

linguistic literature (Vendler 1967) and *events* in the temporal database area (Jensen et al. 1998)), i.e., facts that inherently occur over a "point" in time. Events such as signing of contracts can be naturally associated to time points.

4.2.3 The Sequenced and Non-Sequenced Distinction

As the telic/atelic distinction concerns whether downward and upward inheritance applies, it is specific to facts represented in the conceptual schema. There is another useful distinction, sequenced and non-sequenced (Snodgrass 1999), which concerns the entity class-attribute and entity class-relationship pairs. A temporal constraint is defined as *sequenced* if it is applied independently "at each point in time." On the other hand, a *non-sequenced* constraint either does not treat time specially or references all points of time.

The entity class-attribute (or entity class-relationship) pair can be considered to be sequenced if the interaction between the two occurs at every instant of the existence of each of the participants. For example, balance (an attribute of CONTRACT ACCOUNT) may be time-varying. As each CONTRACT ACCOUNT has a balance at *each instant of time,* this interaction is sequenced. This term reflects a view of time as a "sequence" of instants (Böhlen et al. 1996) with the values of the balance aligning exactly in time instants with the existence (valid) time of the CONTRACT ACCOUNT. Similarly, PROJECT MANAGER and PROJECT play a role in the relationship manages, wherein a PROJECT MANAGER manages a PROJECT at *each point in time*. Thus, the entity class and relationship pair, PROJECT MANAGER-manages and PROJECT-manages, is sequenced.

The entity class-attribute (and entity class-relationship) pair is considered *non-sequenced* if there is a constraint on the interaction that does not involve every point in time, such as *before, meets* (these are a few of the Allen's predicates (Allen 1983)) or involves some more complex constraint between the time periods or temporal elements of the participants. As an example, when a CLIENT signs a CONTRACT, the signing would happen "before" the start of the contract; there is no instant-by-instant correspondence between the two (signs and CONTRACT). Similarly, the signing of a CONTRACT by a CLIENT happens "during" the existence of a CLIENT. These non-sequenced constraints can be defined using thirteen Allen's predicates (Allen 1983): before, before^{-1}, meets, meets^{-1}, overlaps, overlaps^{-1}, finished, finished^{-1}, during, during^{-1}, starts, starts^{-1} and equals. As shown in Figure 6, before(x, y) suggests that the time period of x ends before the start of the time period of y. The converse of before is before^{-1}(y, x) that suggests that the time period of

the start of *y* is *after* that of the end of *x*.

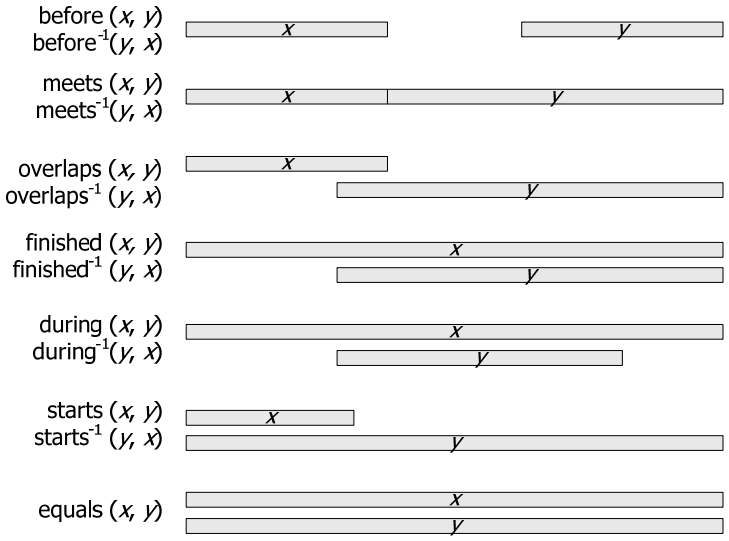


Figure 6: Allen's predicates; adapted from (Allen 1983)

4.2.4 Continuing the Example

We continue with the contract example to illustrate the concepts related to telic/atelic data semantics. Several aspects of the contract application need to be organized with respect to time. For example, two key entity classes of interest in the application are CONTRACT and CONTRACTOR. Both these entity classes need to be referenced with time. While the existence of a CONTRACT entails culmination (and does not satisfy upward and downward inheritance), that with a CONTRACTOR does not. The lifespan of a CONTRACT—with temporal granularity of day—needs to be represented as telic, while that of a CONTRACTOR—with the granularity of day—needs to be represented as atelic. Some of the attributes of a CONTRACT are tasks that are agreed upon in the CONTRACT and budget allocated for the CONTRACT. Before the start of a CONTRACT, a CLIENT signs it at a "point" (event) in time (i.e., day).

To financially manage a CONTRACT, each CONTRACT has a CONTRACT ACCOUNT. The CONTRACT ACCOUNT has a lifespan (atelic temporal) with a temporal granularity of day. To encourage a portfolio-based approach, each CONTRACT is organized such that it is involved with exactly one (1:1) PROJECT (telic temporal with granularity of day). Each PROJECT in turn is managed by a PROJECT MANAGER. A CONTRACTOR reports to a PROJECT MANAGER and the period of time during which this reporting relationship holds needs to be captured (as atelic time). Additionally, to effectively manage a PROJECT, PROJECT MANAGERs meet with CONTRACTORs. Because meetings have a culmination, the relationship has a meeting with between a CONTRACTOR and a PROJECT MANAGER needs to be represented as telic

temporal with a temporal granularity of minutes.

In summary, capturing data semantics related to, e.g., PROJECT MANAGER, PROJECT, CONTRACTOR, CONTRACT ACCOUNT, CONTRACT, CLIENT requires a proposed temporal conceptual model to: 1) allow a data analyst to model non-temporal aspects of the application in a straightforward manner; 2) provide a mechanism to differentiate between atelic-telic temporal aspects of the application; 3) provide a common framework that applies to entity classes, attributes and relationships for expressing the structure of temporal data; 4) include a mechanism to represent multiple granularities in a conceptual schema (e.g., day, minute); and 5) enable a data analyst to specify non-sequenced temporal constraints.

4.2.5 Annotation Syntax

Annotations provide a common framework for specifying the temporal data semantics associated with entity classes, attributes, and relationships.

⟨annotation⟩	::=	ϵ | ⟨temporal annotation⟩ **/ /**
⟨temporal annotation⟩	::=	ϵ | ⟨valid time⟩ / ⟨transaction time⟩
⟨valid time⟩	::=	⟨state⟩ (⟨g_t⟩) | ⟨event⟩ (⟨g_t⟩) | -
⟨state⟩	::=	⟨telic state⟩ | ⟨atelic state⟩
⟨transaction time⟩	::=	T | Transaction | -
⟨telic accomplishment⟩	::=	Acc | Accomplishment
⟨atelic state⟩	::=	S | State | Atelic State
⟨event⟩	::=	E | Event
⟨g_t⟩	::=	⟨day⟩ | ⟨hour⟩ | ⟨minute⟩ | ⟨second⟩ | ⟨user defined⟩
⟨day⟩	::=	day
⟨hour⟩	::=	hr | hour
⟨minute⟩	::=	min | minute
⟨second⟩	::=	sec | second

Figure 7: Annotation syntax in BNF

As shown in Figure 7 above, the overall structure of an *annotation phrase* is "⟨temporal annotation⟩//". The temporal annotation first specifies the valid time (or existence time/lifespan) followed by the transaction time. The temporal annotation for valid 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 with "-". The valid time can be modeled as an event (E), an atelic state (S) or a telic accomplishment (Acc) and has an associated temporal granularity. For example, "S(day)/-//" associated with a CONTRACTOR denotes that a CONTRACTOR exists (i.e., has an associated existence time/lifespan or valid time), and the temporal granularity of the atelic states (S) is day. Additionally, we do not need to

capture transaction time (hence, "-") associated with the CONTRACTOR. Similarly, annotation phrase "E(day)/-//" associated with signs implies that a contract was signed on a day. PROJECT.repair is telic temporal attribute having "Acc(day)/-//" as an associated annotation phrase because repair has culmination.

In summary, annotations provide a succinct mechanism for denoting a complex assemblage of semantics: telic or atelic, event or state or accomplishment, valid or transaction time along with the associated granularities. Note that an annotation phrase can be consistently applied to entity classes, attributes and relationships. Also note that this approach does not propose new "constructs" for capturing temporal data semantics. Instead, the annotation-based approach "tweaks" the semantics of concepts in a conventional conceptual model, e.g., USM, using annotations.

4.2.6 The Example Revisited

Figure 8 shows the annotated temporal schema for the contract application. Note however that our annotation-based approach is not specific to USM and can be applied to any conventional conceptual model (Chen 1976; Elmasri and Navathe 2006). Note also that this annotated schema is very similar to the non-temporal USM schema in Figure 2, which is without annotations.

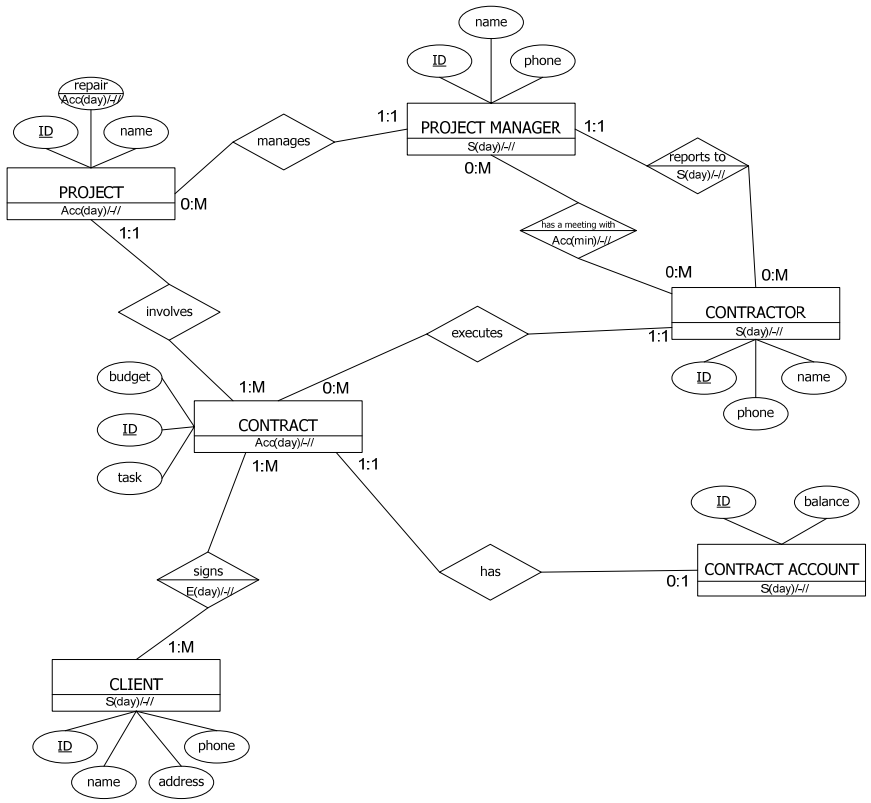


Figure 8: An annotated temporal schema for the contract application

To develop the annotated schema (e.g., Figure 8), which is based on a non-temporal schema (e.g., Figure 5), the data analyst asks (the user) questions such as: Do you want to capture history (lifespan) or only current values of the facts (objects)? Does the fact need to be modeled as an event or state (i.e., durative facts)? Does upward and downward inheritance hold for the durative facts? Accordingly, the data analyst annotates the schema. Note how Figure 8 augments the schema shown in Figure 5 with temporal annotations. Some entity classes, attributes and relationships are represented as non-temporal. This does not imply that they are non-temporal in reality, but rather that one is not interested to deal with its temporal features (if any) in the application or the "miniworld."

4.2.7 Specifying Non-sequenced Semantics

As shown in Figure 4, our approach for adding temporal semantics proceeds in three steps. Figure 5 provides an example of "what" semantics, while Figure 8 is an example of "when" (along with "what") semantics using annotations. We next turn to the third step, which involves capturing the non-sequenced semantics.

Besides the semantics that are implied by annotations, a data analyst can optionally define other explicit temporal constraints that capture the temporal relationship; such constraints are based on Allen's predicates (Allen 1983) such as before, meets, overlap, finished, during, starts, and equal. See Table 2 for an example of non-sequenced constraints in a temporal data dictionary.

Attribute/ Relationship	Entity class(es)	Non-sequenced constraint	Comment
signs	CONTRACT	before	Signing of the contract must happen before the lifespan of the contract.
has a meeting with	CONTRACTOR	during	The time period of a meeting must be during the lifespan of a contractor.
has a meeting with	PROJECT MANAGER	during	The time period of a meeting must be during the lifespan of a project manager.
reports to	CONTRACTOR	during	The time period of the reporting relationship must be during the lifespan of a contractor.
reports to	PROJECT MANAGER	during	The time period of the reporting relationship must be during the lifespan of a project manager.
repair	PROJECT	during	The time period of a repair must occur during the lifespan of a project.

Table 2: Specifying non-sequenced constraints

A constraint can be specified between any two annotated constructs, e.g., between the temporal relationship signs and the temporal entity class CONTRACT or between the temporal entity class PROJECT and the temporal attribute repair. As shown in Table 2, the constraint before between CONTRACT ("Acc(day)/-//")

and signs ("E(day)/-//") implies that the event signs must happen "before" the lifespan of the CONTRACT. Note that the table does not directly differentiate between before and before^{-1} (or *after*); this is indicated in the comment column instead.

Having discussed how annotation phrases and non-sequenced temporal constraints can be specified, we next examine the semantics of annotations as they apply to entity classes, attributes and relationships.

5. Explicating Temporal Semantics

We specify here the semantics of telic/atelic annotations on entity classes, attributes and relationships using the conventional USM; we do so using an approach that employs inheritance.

In the following, we do not include details related to atelic abstractions as they are described elsewhere (Khatri et al. 2001; Khatri et al. 2002). However, we do discuss some characteristics of the atelic semantics to bring out interesting differences between the telic and atelic semantics.

5.1 Semantics of Annotations

As shown in Figure 9 below, VT_EVENTUALITY and TT_EVENTUALITY are superclasses in our superclass/subclass hierarchy that are associated with valid time and transaction time, respectively. Each VT_EVENTUALITY and TT_EVENTUALITY is associated with a TEMPORAL_GRANULARITY. TEMPORAL_GRANULARITY (described in detail in Khatri et al. (2004)) includes the recursive relationships groups_into and anchor_gran, which helps create the *granularity graph* (Dyreson et al. 2000).

The valid time eventuality[7] (VT_EVENTUALITY) has three subclasses that represent events (VT_EVENT), atelic state (VT_STATE) and telic state (VT_ACCOMPLISHMENT). On the other hand, transaction time eventuality (TT_EVENTUALITY) has one subclass (TT_STATE) that indicates that transaction time is always atelic (i.e., there is *no* TT_ACCOMPLISHMENT subclass). We adopt maximal convex time periods to represent the valid time of atelic facts (Dowty 1986; Terenziani and Snodgrass 2004), with the intended meaning that the fact holds at each point (chronon) within the maximal time period. Each TT_STATE holds in a set of maximal convex time periods, where each time period is represented with indexes begin and end. While VT_EVENT occurs in time points, VT_STATE holds over maximal_periods (representing a convex maximal set of time points).

[7] Following the linguistic literature (see, for example, Vendler 1967), we use the term *eventuality* for all aktionsart statements.

The association of telic facts to time must be dealt with in a different way with respect to atelic facts. Convex maximal periods, interpreted as convex sets of time points, cannot be used since upward and downward inheritance would be inappropriately enforced. On the other hand, telic semantics in which time periods are conceived as atomic entities, which cannot be divided or merged together and time periods *can* overlap, need to be associated with these facts. Hence, a VT_ACCOMPLISHMENT holds over telic_periods (representing atomic indivisible entities). Via formal axioms, we explicate the inherent semantics that are associated with various temporal concepts. These axioms are predicates that are implied by the data analyst who annotates a conceptual schema.

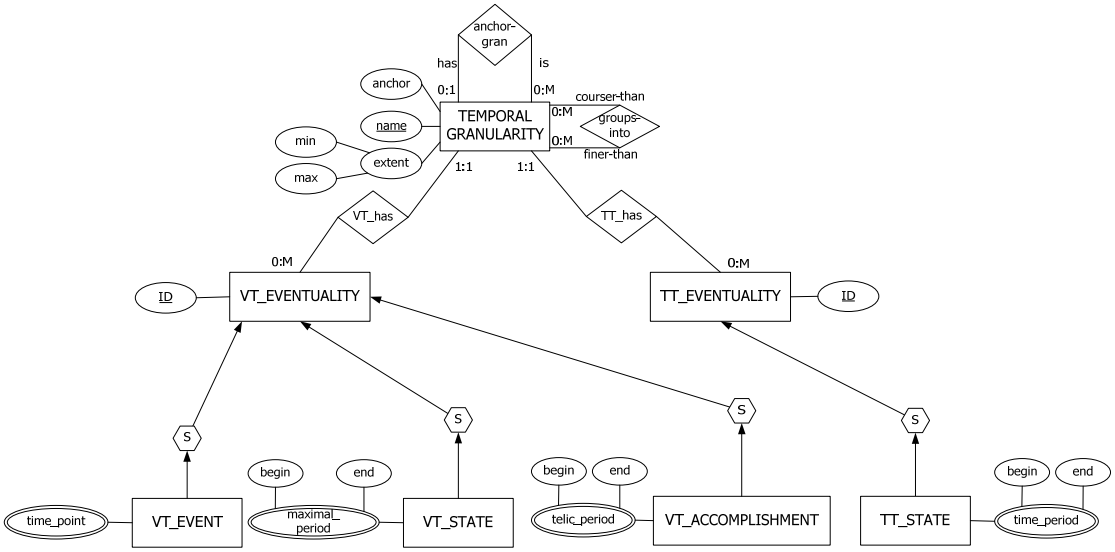


Figure 9: Semantics of temporal annotations

In these predicates, the values of attributes of instances of classes are denoted functionally. So if *e* is a specific instance of VT_ACCOMPLISHMENT, then VT_ACCOMPLISHMENT(*e*, telic_period) denotes the value of the telic_period attribute of *e*.

Axiom 1: The following axioms characterize the maximal_period and telic_period attributes. Note how the well-formed telic period (i.e., a) below) differs from that of an atelic period (i.e., b) and c) below).

a) Telic periods (i.e., the valid times of VT_ACCOMPLISHMENTs) are well-formed.
$\forall e \in S$(VT_ACCOMPLISHMENT), $\forall p \in$ VT_ACCOMPLISHMENT(e, telic_period), begin(e, p) < end(e, p)

b) Maximal temporal periods (i.e., the valid time of VT_STATEs) are well-formed.
$\forall e \in S$(VT_STATE), $\forall p \in$ VT_STATE(e, maximal_ period), begin(e, p) < end (e, p)

c) Maximal temporal periods (i.e., the valid time of VT_STATEs) cannot overlap in time. Such a constraint does ***not*** hold for telic periods.
$\forall e \in S$(VT_STATE),
$\forall p_1, p_2 \in$ VT_STATE(e, maximal_ period), begin(e, p_1) < begin(e, p_2) $\Rightarrow$ end (e, p_1) < begin (e, p_2)

Axiom 2: Upward and downward inheritance holds for atelic state (VT_STATE) but not for telic state (VT_ACCOMPLISHMENT). In order to state the properties of VT_STATE and VT_ACCOMPLISHMENT, we introduce "hold" as an inferred attribute that relates each VT_EVENTUALITY to its valid time.

$\forall e \in S$(VT_EVENTUALITY),
VT_EVENT(e, time_point)$\Rightarrow$ VT_EVENTUALITY(e, hold) = to_element (VT_EVENT(e, time_point)) $\wedge$
VT_STATE (e, maximal_period) $\Rightarrow$ VT_EVENTUALITY(e, hold) = VT_STATE(e, maximal_period) $\wedge$
VT_ACCOMPLISHMENT (e, telic_period) $\Rightarrow$
VT_EVENTUALITY(e, hold) = VT_ACCOMPLISHMENT(e, telic_period)

where to_element is a function that takes as input a time point and coerces it to a one-chronon period.

The *VT_Down* and *VT_Up* axioms in the following specify that the properties of downward and upward inheritance that holds on VT_STATE.

a) VT_Down

$\forall e,\ p_1, p_2,$ VT_STATE(e) $\wedge$ VT_STATE.hold(e, p_1) $\wedge\ p_2 \subseteq\ p_1 \Rightarrow$ VT_STATE.hold(e, p_2)

b) VT_Up

$\forall e,\ p_1, p_2,$ VT_STATE(e) $\wedge$ VT_STATE.hold(e, p_1) $\wedge$ VT_STATE.hold(e, p_2) $\wedge$
(MEETS(p_1, p_2) $\vee$ MEETS^{-1}(p_1, p_2) $\vee$ OVERLAPS(p_1, p_2) $\vee$ OVERLAPS^{-1}(p_1, p_2)
$\vee$ FINISHED(p_1, p_2) $\vee$ FINISHED^{-1}(p_1, p_2) $\vee$ DURING(p_1, p_2)
$\vee$ DURING^{-1}(p_1, p_2) $\vee$ STARTS(p_1, p_2) $\vee$ STARTS^{-1}(p_1, p_2) $\vee$ EQUALS(p_1, p_2)
) $\Rightarrow$ VT_STATE.hold(e, $p_1 \cup p_2$)

where $\cup$ denotes set-union over maximal periods, i.e., over sets of time points and the temporal relations in the disjunction are relations of Allen's Interval Algebra (Allen 1983) that indicate that p_1 and p_2 are either contiguous or intersecting in time (i.e., they are not temporally disjoint); see Figure 6 for a schematic of Allen's predicates. The absence of analogous axioms for VT_ACCOMPLISHMENT means that downward and upward inheritance *does not* hold for accomplishments. Indeed, that is the central telic/atelic distinction we wish to capture.

5.2 ERA

In the following, we elucidate the semantics of an annotated temporal entity class, relationship and attribute, collectively referred to as ERA. Specifying the semantics of a telic/atelic fact is rather straightforward: the semantics of each annotated ERA can be obtained by classifying it as a subclass of an appropriate temporal entity class such as VT_EVENT, VT_STATE, VT_ACCOMPLISHMENT and TT_STATE; see Figure 9. The temporal semantics of the annotated ERA are thus automatically obtained through inheritance from such superclasses. The schema that explicitly shows the semantics of the annotations is referred to as the *translated USM schema*; see the bottom of Figure 10 for an example. Also, see Appendix A for an example of the translated USM schema (along with the automatically generated axioms) for a (shaded)

fragment of the annotated temporal schema.

5.2.1 Temporal Entity Class

We describe the semantics of a telic temporal entity class using the example of CONTRACT. As discussed above, CONTRACT is telic since neither downward nor upward inheritance applies. Figure 10 shows the semantics of the telic temporal entity class, CONTRACT. Note that the associated annotation for this entity class is "Acc(day)/-//".

The semantics of CONTRACT is explicated by relating it to the appropriate superclass, which is VT_ACCOMPLISHMENT. This implies that CONTRACT inherits all the properties (i.e., attributes and relationships) of its superclasses, VT_ACCOMPLISHMENT and VT_EVENTUALITY (i.e., only axiom 1a will apply). Similarly, the semantics of an atelic entity class (e.g., CONTRACTOR) can be obtained by classifying it as a subclass of VT_STATE; in this way, the upward and downward properties are obtained through inheritance and axioms 1b, 1c, and 2 apply. For instance, a contract `C1` with a lifespan `[2004-2006]` from VT_STATE(`C1`) and axiom 2 (first part), one gets VT_STATE (`C1`, hold) = `[2004-2006]`, which in conjunction with Axiom 2.a (VT_Down) and that `[2004,2004]` $\subseteq$ `[2004,2006]` implies that the contract `C1` was valid in 2004, i.e., VT_STATE (`C1`, hold) = `[2004-2004]`.

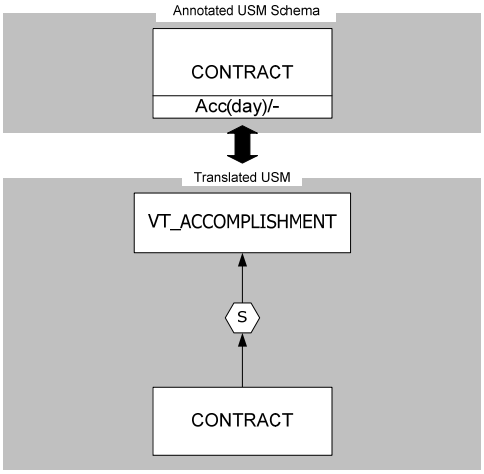


Figure 10: Semantics of telic temporal entity class

Bi-temporal entity classes would be entity classes that are a specialization of both VT_EVENTUALITY (VT_EVENT, VT_STATE or VT_ACCOMPLISHMENT) and TT_EVENTUALITY (TT_STATE). Compared with the previous approach (Khatri et al. 2001; Khatri et al. 2002), employing superclasses and subclasses to represent temporal data semantics is more parsimonious.

Finally, the following axiom related to granularity is similar to that in Khatri et al. (2002).

Axiom 3: All the entities in CONTRACT have the same associated temporal granularity (day).

$\forall e \in S$ (CONTRACT), CONTRACT.VT_ACCOMPLISHMENT.VT_EVENTUALITY.
VT_has.TEMPORAL_GRANULARITY(*e*, name) = day

5.2.2 Temporal Attribute

The semantics of temporal attributes can be explicated through the introduction of an auxiliary "constructed" entity class (in the translated USM schema), which is then associated with the superclass VT_STATE, VT_EVENT or VT_ACCOMPLISHMENT. The auxiliary constructed entity class is related to the entity class to which the attribute refers via an "auxiliary constructed relationship". The cardinalities to the original entity class is 1:1 for standard attributes, 1:N for multi-valued attributes, while the cardinality to the constructed entity class is 0:1 because optionality of an attribute is typically not shown on the conventional conceptual schemas (and is shown in the data dictionary). Additionally, axiom 1a and that similar to axiom 3 would apply.

Figure 11 shows the semantics of a telic temporal attribute PROJECT.repair in Figure 8. As shown in Figure 11, the auxiliary "constructed" entity class SEM_ATTR_repair_PROJECT is a subclass of VT_ACCOMPLISHMENT and is related to the entity class PROJECT by the "auxiliary constructed relationship" ATTR_OF_repair_PROJECT.

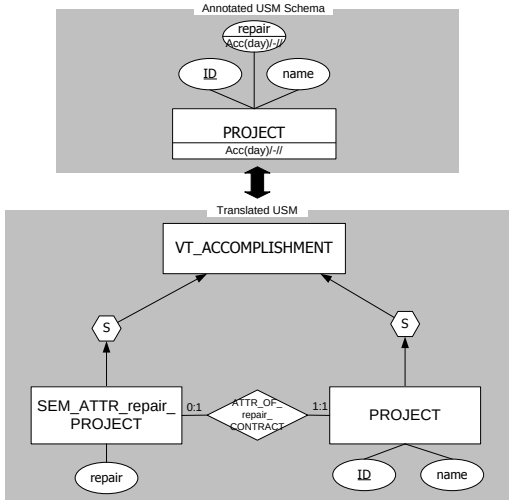


Figure 11: Semantics of telic temporal attribute

5.2.3 Temporal Relationship

The explication of the semantics of a temporal relationship is similar to that of an attribute described above.

For example, Figure 12 shows the semantics of a telic temporal relationship (meets_with). The semantics is explicated through the introduction of auxiliary "constructed" entity class, SEM_REL_meets_with.

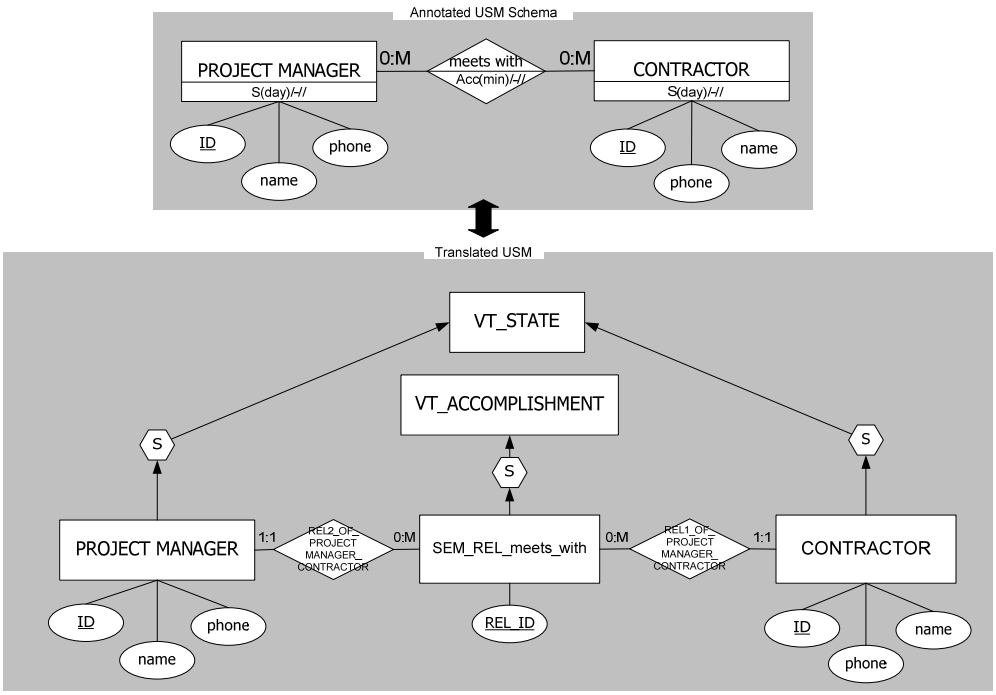


Figure 12: Semantics of telic temporal relationship

While the "constructed" entity class (SEM_REL_meets_with) inherits attributes and relationships from VT_ACCOMPLISHMENT superclass, PROJECT MANAGER and CONTRACTOR inherit attributes and relationships from VT_STATE because their associated annotations are "Acc(min)/-//" and "S(day)/-//", respectively. Additionally, axiom 1a and that similar to axiom 3 would apply.

5.3 Semantics of Composition

In Section 5.2, we discussed how the telic/atelic distinction can be captured via annotations; however, we presented the semantics of annotations as related to ERAs "in isolation." In this section, we take into account the semantics of the interactions (sequenced/non-sequenced) between a pair of ERAs, i.e., entity class-relationship and entity class-attribute. Thus, sequenced/non-sequenced semantics are orthogonal to the telic/atelic distinction.

As shown in Figure 8, signs (represented as an event) is a relationship between entity classes CLIENT (which is atelic) and CONTRACT (which is telic). While the relationship between signs and CLIENT is sequenced which implies that signing must be performed at one point during the existence of the client, that between signs and CONTRACT is non-sequenced (see Table 2) which signifies that signing must be

performed “before” the contract. This example illustrates an important feature of our approach: the interplay of semantics between ERAs can be captured in a pair-wise compositional way.

We assume that the sequenced constraint as the default; on the other hand non-sequenced constraints can be explicitly specified by the data analyst in the temporal data dictionary (see Table 2 for example). In the rest of this section, we first provide the default (sequenced) semantics with the understanding that a data analyst can override these semantics at any point by providing alternative constraints explicitly. Next, we take into account the non-sequenced semantics.

5.3.1 Sequenced Semantics

In the sequenced interpretation, an exact correspondence (for each chronon of time) is required between the pair of entity class-attribute or entity class-relationship.

As shown in Table 3, non-temporal, atelic and telic temporal attributes/relationships can interact with non-temporal, atelic, and telic temporal entity classes. Table 3 indicates that 16 cases must be taken into account; however, few general principles are adequate to summarize all the possible combinations. Note that the sequenced interpretation requires a chronon-by-chronon (or point-by-point) correspondence, which, in turn, requires an atelic interpretation of the involved ERAs. Therefore, telic ERAs must be coerced into atelic ones (column “coercion to atelic” in Table 3).

- If the attribute/relationship (or entity class) is non-temporal and the entity class (or attribute/relationship) is temporal, the non-temporal attribute/relationship (or entity class) is coerced to be temporal, e.g., of type VT_STATE (for row 4 in Table 3); if the data analyst wanted something else, they should have specified it as so.
- The validity time of the entity class and of the attribute/relationship (i.e., of its corresponding constructed entity class in ST USM) must be same. If the attribute/relationship is an event, the sequenced interpretation requires that its validity time (a time point) must be *during* the validity time of the entity class.
- If the entity class is an event, the attribute/relationship cannot be VT_STATE or VT_ACCOMPLISHMENT within the sequenced interpretation. Such an interpretation demands that the validity time of attribute/relationship must be equal to or contained in the validity time of the entity class it is related to. Table 3 shows different cases of semantics of composition, where “E” and “A/R” refers to entity class

and attribute/relationship, respectively. The first column in this table is simply the row number. Sixteen constraints are required to fully cover all the possibilities. The second and third columns identify various combinations between different types of "A/R" and "E." So, for example, the row 10 considers either a non-temporal attribute of a state entity class or a state entity class participating in a non-temporal relationship. The fourth column lists a conversion which is necessarily implied by five of the combinations; we'll describe row 7 shortly. The final column gives some details of the semantics.

This table can be expressed by a set of constraints. We provide here the constraint corresponding to the row 10 of the table where: 1) the attribute/relationship (which has been specified as non-temporal) is inferred as temporal; and 2) the valid time of the attribute/relationship and that of the entity class must be equal. For each non-temporal attribute (e.g., name of the entity class PROJECT MANAGER), we first need an auxiliary entity (e.g., SEM_ATTR_PROJECT MANAGER_name) that is used to infer the ATTR_OF predicate.

$$\forall e_1 \in S(\text{PROJECT MANAGER}), \forall e_2 \in S(\text{name}), \text{SEM\_ATTR\_PROJECT MANAGER\_name}\ (e_1, e_2) \Rightarrow \text{ATTR\_OF}\ (e_2, e_1)$$

Then, the general constraint below is used to enforce the fact that the ATTR_OF predicate makes the (coerced) valid time of the (auxiliary entity representing the) attribute (or relationship) equal to that of the temporal entity it is associated with.

$$\forall e, \forall a, p_1,\ e.\text{VT\_STATE} \wedge (e.\text{VT\_STATE}(\text{hold}) = p_1) \wedge \text{ATTR\_OF}(a, e) \wedge (a.\text{annotations} \neq \text{Acc} \wedge a.\text{annotations} \neq \text{E} \wedge a.\text{annotations} \neq \text{S}) \Rightarrow a \in \text{S(VT\_STATE)} \wedge a.\text{VT\_STATE(hold)} = p_1$$

Similar constraint would be required to express each possible case in Table 3 below.

No	Type of attribute/relationship (*A*/*R*)	Type of entity class (*E*)	Coercion to atelic	Semantics
1	Non-temporal	Non-temporal	–	Conventional
2	VT_STATE	VT_STATE	–	
3	VT_STATE	VT_ACCOMPLISHMENT	*E*	
4	VT_STATE	Non-temporal	–	Infer *E* is temporal
5	VT_STATE	VT_EVENT		Disallowed
6	VT_ACCOMPLISHMENT	VT_STATE	*A/R*	
7	VT_ACCOMPLISHMENT	VT_ACCOMPLISHMENT	*E*, A/R	
8	VT_ACCOMPLISHMENT	Non-temporal	*A/R*	Infer *E* is temporal
9	VT_ACCOMPLISHMENT	VT_EVENT		Disallowed
10	Non-temporal	VT_STATE	–	Infer *A*/*R* is temporal
11	Non-temporal	VT_ACCOMPLISHMENT	*E*	Infer *A*/*R* is temporal
12	Non-temporal	VT_EVENT	–	Infer *A*/*R* is temporal
13	VT_EVENT	VT_STATE	–	
14	VT_EVENT	VT_ACCOMPLISHMENT	–	
15	VT_EVENT	Non-temporal	–	Infer *E* is temporal
16	VT_EVENT	VT_EVENT	–	

Table 3: Interaction of temporal/non-temporal attribute/relationship with temporal/non-temporal entity class

It is worth noticing that, based on row 7 of Table 3, if an attribute/relationship or entity class is telic (VT_ACCOMPLISHMENT), it is converted to atelic for the purpose of the equality constraint. This is needed for upward inheritance. For example, the PROJECT entity class is telic as is its attribute repair; see Figure 8. Each PROJECT, say `P1` with lifespan `[2004-2007]`, has associated repairs, say `R1` (`[2005-2006]`) and `R2` (`[2006-2007]`), which implies that the periods of the possibly several repairs, each at different times, are converted to atelic, collated together (`[2005-2007]`), and then compared for equality with the existence time of the PROJECT `P1`.

Note that there are subtle ramifications in considering non-temporal entity classes to be temporal for the sake of the sequenced semantics. For example, if the phone attribute of CONTRACTOR is represented as atelic state (with temporal granularity of day) and the CONTRACTOR is assumed non-temporal, CONTRACTOR will be considered to be an entity class whose lifespan (valid time) is equal to the validity period of phone; see row 4 of Table 3. Should CONTRACTOR have another atelic temporal attribute, that attribute would be required by this "composed" semantics to have a validity period equal to that of CONTRACTOR, thus, that of the phone attribute. By this linkage, which we term *compositionality*, the implication is that under the sequenced interpretation all temporal attributes must have identical validity time periods. While prior research (see, for example, Klein 1994) suggests that "all" objects, their properties and relationships are embedded in time, data analysts can *choose* (or *not choose*) to capture the associated temporality for an application. Thus, implicit semantics (described above) elucidate the temporality "assumptions" for an application.

Similarly, the budget and task (non-temporal attributes) associated with CONTRACT (a telic entity class) in a non-sequenced manner implies that the attributes are time-invariant. Hence, it can be assumed that the same budget and task were used for the entire time period(s) of existence of a particular CONTRACT. If, however, different tasks were employed during the lifespan of CONTRACT and the time periods of tasks (during the lifespan of CONTRACT) were pertinent for the application, then the task attribute needs to be represented as an atelic temporal attribute with a sequenced interpretation (row 3 of Table 3).

5.3.2 Non-Sequenced Semantics

As discussed in Section 4.7, the non-sequenced interpretation allows data analysts to specify constraints between ERAs. Besides the constraints that are *implied* by annotations, a data analyst can optionally define

other explicit temporal constraints that capture the temporal relationship; such constraints are based on Allen's predicates (Allen 1983) such as before and meets. The semantics of such non-sequenced constraints is just that of these predicates applied to the relevant timestamps.

5.4 Temporal Propagation

In Section 5.3 we showed that in both sequenced and non-sequenced interpretations, temporal constraints are imposed on the validity time of the related ERAs. Such pairwise temporal constraints propagate in a compositional way in the annotated schema. For example, consider the entity classes PROJECT and PROJECT MANAGER and the relationship manages between them. Based on the annotated schema shown in Figure 8, PROJECT, PROJECT MANAGER, manages need to be represented as telic, atelic and nontemporal, respectively. Moreover (by default), the relations between PROJECT and manages and between manages and PROJECT MANAGER are both *sequenced*. Consider PROJECT and manages, where rule 11 in Table 3 applies, so that PROJECT is coerced to atelic, manages is coerced to atelic and their validity time are imposed to be the same. Consider again PROJECT MANAGER and manages, where rule 10 in Table 3 applies, so that manages is coerced to atelic, and its validity time is imposed to be equal to the validity time of PROJECT MANAGER. As a consequence, it is inferred that the validity time of the two must be the same. Notice that such a constraint is *implicit* in the annotated schema, i.e., it is logically *implied* by the underlying semantics of ERAs and by the sequenced/non-sequenced interpretations of interactions between ERAs.

6. Evaluation

Based on the criteria presented in Section 3.2, we evaluate our proposed approach.

- *Expressiveness:* The ontologic basis for telic data semantics is rooted in the linguistic literature (Vendler 1967; Webber 1988) that differentiates statements that are stative from those that are kinesis. While the latter has clearly defined "culmination," the former does not. As far as valid time (Snodgrass and Ahn 1986) is concerned, they are of only two types facts: telic and atelic. Hence our proposed approach is expressive within the boundary that this research explores. Note that in conceptual modeling, which seeks to capture data semantics of the "real world," it is impossible to assert completeness in absolute sense. However, because the annotations are orthogonal to conceptual

modeling constructs, the proposed approach is straightforward to extend.

- *Simplicity:* Based on the general problem-solving approach of divide-and-conquer, the annotation-based approach divides conceptual design into two phases: first capture "what" semantics and then associate "when" semantics with "what" constructs. Prior research, based on cognitive fit (Vessey 1991) and human associative memory (Anderson and Bower 1973), suggests that annotations *in* the schema result in the matching of the external problem representation (schema) with internal task representation; on the other, annotations *outside* the schema result in a mismatch (Khatri et al. 2006b). Thus, prior research suggests that the annotated schemas should be straightforward to comprehend.
- *Minimality:* Since various conceptual modeling constructs are orthogonal to temporal annotations, the annotations are minimal and generic. For example, an annotation phrase "Acc(day)/-//" can be applied to an entity class, an attribute or a relationship.
- *Formality:* We defined the annotation syntax formally in BNF and have used first-order logic to define the temporal data semantics.
- *Upward compatibility:* Since our extension is a strict superset that is provided by adding non-mandatory semantics, the annotation-based temporal conceptual model is upward compatible with conventional conceptual model.

7. Implications

This research presents an approach for temporal conceptual design that helps capture telic/atelic data semantics: 1) first capture "what" semantics using a conventional conceptual model; 2) then employ annotations to differentiate between telic/atelic "when" semantics; 3) finally, specify temporal constraints, specifically non-sequenced semantics, in the temporal data dictionary. We showed how the annotated schema captures rich semantics and that the semantics of the annotated schema can be "unwrapped." For example, see the translated USM schema as well as the automatically generated axioms in Appendix A that explicate the semantics of a small fragment of the (shaded) annotated schema. In this section, we present the contributions of our research and conclude with the implications of our research for both future research and for practice.

With respect to prior approaches, and, in particular ST-USM (Khatri et al. 2004; Khatri et al. 2002), this work makes three key contributions. First, we include both telic/atelic and sequenced/non-sequenced

dichotomies, thus, enhancing the expressiveness of the prior approach. Both dichotomies (telic/atelic and sequenced/non-sequenced) are important in temporal databases. We showed that they are orthogonal since the telic/atelic distinction concerns the temporal properties of entity classes, attributes and relationships per se, while the sequenced/non-sequenced distinction concerns the entity class-attribute/ entity class-relationship pair. Notice that only atelic entity types and properties and sequenced relationships have been allowed in ST-USM (Khatri et al. 2004; Khatri et al. 2002) as well as the prior other approaches in the literature (see, for example, Parent et al. 1999; Tryfona and Jensen 1999; Tryfona and Jensen 2000).

Second, while our proposed approach is more expressive, it maintains the succinctness of the prior proposal (Khatri et al. 2004; Khatri et al. 2002) because of two features: compositionality and inheritance. To keep the semantics manageable and clear, they need to be modeled in a compositional way. In our approach, the semantics of the entire conceptual schema can be split into the semantics of each entity class/attribute/relationship alone and entity class-attribute/ entity class-relationship pair considered separately. (In passing, notice that such a compositional approach perfectly fits with the orthogonality of the telic/atelic and sequenced-non-sequenced distinctions. This enhances the clarity and the compactness of the proposed approach.)

Third, to further enhance compactness and clarity, we have used inheritance throughout. In particular, in the proposed approach, we specify the semantics of a taxonomy of few "basic" entity classes (e.g., VT_STATE, VT_ACCOMPLISHMENT, VT_EVENT), so that the semantics of specific entity classes and properties in a conceptual model is simply obtained by inheritance by classifying them along the taxonomy. In such a way, we provide a "direct" semantics to the "S", "Acc" and "E" annotations: this implies that the annotated entity class has to be classified as state, accomplishment and event, respectively, thus, inheriting the corresponding (implicit) temporal properties.

7.1 Implications for Research

Our research has several implications for researchers. First, we have briefly mentioned the notion of implicit constraints and temporal propagation. Future research should provide algorithms to propagate such constraints, e.g., in order make implied constraints explicit and/or to check the consistency of explicit and implicit constraints; see, for example, the survey (Vila 1994) that describes temporal constraints propagation algorithms for temporal constraints.

Second, we focused on a specific type of temporal constraints, i.e., non-sequenced temporal constraints. Future research should investigate temporal constraints in the context of cardinality constraints (see, for example, Currim 2004).

Third, the role of telic/atelic distinction in spatial data needs to be explored in the future. While several spatial conceptual models have been proposed (see, for example, Khatri et al. 2004; Parent et al. 1999; Tryfona and Jensen 1999; Tryfona and Jensen 2000), telic/atelic aspects have not been included in these models.

Fourth, while we presented a mechanism to capture non-sequenced temporal semantics, future research needs to similarly explore non-sequenced spatial semantics. Similar to temporal constraints that are based on Allen's predicates, spatial constraints that are based on topological constraints, such as meets, equals, inside and covers, can be specified (Nunes 1990). Additionally, the role of the telic/atelic distinction in spatial queries needs to be examined further; this distinction has been considered for temporal queries (Terenziani and Snodgrass 2004). Additionally, it would be helpful to determine which kinds of non-sequenced constraints are useful, and whether these constraints might propagate, as do the sequenced constraints (cf. compositionality).

Fifth, future research needs to explore how a design support environment can support the elicitation of telic/atelic data semantics. For example, in a design-support environment, clicking on a line could bring up a dialog box that would allow a constraint to be specified, which would then render that pair non-sequenced. Figure 14 in the Appendix B illustrates an example of specifying temporal annotation ("Acc(day)/- ") for CONTRACT and specifying non-sequenced temporal constraint (before) between CONTRACT and signs. Via a proof of concept design support environment, prior research has shown how the annotation-based approach can be embedded in extant design-support environment (Khatri et al. 2006a); such an environment integrates with existing database design methodologies and results in upward compatible conceptual as well as XML schemas.

Sixth, while prior research—based on schema understanding (see, for example, Bodart et al. 2001)—suggests that annotations should be placed *in* the schema (Khatri et al. 2006b) rather than *outside* the schema, further research should examine how the annotations should be rendered graphically in the schema. Further research should also evaluate the effect of annotations on the development (rather than schema understanding) of temporal conceptual schema (see, for example, Kim and March 1995).

7.2 Implications for Practice

Our research also has several implications for practice. We presented an approach for eliciting telic/atelic semantics that can be employed in conceptual design. We also provided a mechanism for capturing non-sequenced semantics. If there is no such mechanism to differentiate between telic/atelic semantics, or to model sequenced vs. non-sequenced interpretations, it can impact the ultimate use of the database.

Second, our proposed approach has not only immediate implications for the design of temporal databases, but also for the future when a temporal logical model is supported by the DBMS. While both the annotated schema and the translated USM schema can be employed in the design of temporal databases, the mapping to a logical schema depends on the representational model used. Because SQL (structured query language) does not provide time support except for user-defined time, over 50 temporal query languages (Jensen et al. 1998), including SQL/Temporal (Snodgrass et al. 1996a; Snodgrass et al. 1996b), have been proposed, most of which extend SQL to include temporal data semantics. While the mapping from a translated USM schema to a logical schema will be similar to the one proposed in standard textbooks (e.g., Elmasri and Navathe 2006), the one from an annotated USM schema to a temporal logical schema will depend on the temporal logical model itself; former approach has immediate implications, while the latter has implications for the future.

Third, because our proposed approach is upward compatible, it "naturally" augments conventional conceptual design. The natural augmentation implies that the proposed temporal conceptual design approach "generalizes" (rather than "specializes") the conventional conceptual design. For example, note how Table 3 extends the semantics of a conventional conceptual model; while the first row shows the semantics of (entity class-attribute/relationship) interaction in a conventional conceptual model, we provide fifteen other possible interactions that are possible because of capturing telic/atelic temporal data semantics. Also note how annotation syntax in Figure 7 includes ϵ (i.e., empty annotation), thus, implying that conventional conceptual design is a special case of the more general temporal conceptual design.

Fourth, the annotated approach could be used as the basis for augmenting existing CASE tools. Prior research suggests that such an approach would be straightforward to implement (see, for example, Khatri et al. 2006a).

8. Conclusion

In this paper, we differentiate facts with goal-related semantics (telic) from those that are intrinsically devoid of culmination (atelic). Using an example, we illustrated how failing to distinguish between telic and atelic semantics can affect the interpretation of the "content" of the database. Thus, capturing the meaning of the telic/atelic data, or, data semantics, can in turn affect the overall use of the database. In this work, we have extended prior annotation-based approach (Khatri et al. 2004; Khatri et al. 2002) to provide a mechanism to help capture telic/atelic data semantics. We also formally discuss the sequenced-non-sequenced semantics that can be captured during conceptual design, thus, significantly extending the types of temporal data semantics that can be captured during conceptual modeling.

References

Allen, J. F. (1983), "Maintaining knowledge about temporal intervals," *Communications of the ACM*, 26 (11), 832-43.

Allen, J. F. (1984), "Toward a General Theory of Action and Time," *Artificial Intelligence*, 23 (2), 123-54.

Allen, J.F. (1991), "Time and time again: The many ways to represent time," *International Journal of Intelligent Systems*, 6 (4), 341-56.

Anderson, John R. and Gordon H. Bower (1973), Human Associative Memory. Washington, D.C.: V. H. Winston & Sons.

Aristotle Categories. On Interpretation. Prior Analytics. Cambridge, MA Harvard University Press.

Batini, Carlo, Stefano Ceri, and Shamkant B. Navathe (1992), Conceptual Database Design: An Entity-Relationship Approach: Benjamin/Cummings Publishing Company.

Bettini, C., C. E. Dyreson, W. S. Evans, R. T. Snodgrass, and X. S. Wang (1998), "A Glossary of Time Granularity Concepts," in Temporal Databases: Research and Practice, O. Etzion and S. Jajodia and S. Sripada, Eds.: Springer-Verlag.

Bettini, Claudio, Sushil Jajodia, and Sean X. Wang (2000), Time Granularities in Databases, Data Mining and Temporal Reasoning. Berlin: Springer-Verlag.

Bloom, L., L. Lifter, and J. Hafitz (1980), "Semantics of Verbs and the Developments of Verb Inflection in Child Language," *Language*, 52 (2), 386-412.

Bodart, Francois, Arvind Patel, Marc Sim, and Ron Weber (2001), "Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests," *Information Systems Research*, 12 (4), 384-405.

Böhlen, Michael H., Christian S. Jensen, and Richard T. Snodgrass (2000), "Temporal Statement Modifiers," *ACM Transactions on Database Systems*, 25 (4), 407-56.

Böhlen, Michael H., Richard T. Snodgrass, and Michael D. Soo (1996), "Coalescing in Temporal Databases," in Proceedings of 22th International Conference on Very Large Data Bases, T. M. Vijayaraman and Alejandro P. Buchmann and C. Mohan and Nandlal L. Sarda (Eds.). Mumbai (Bombay), India: Morgan Kaufmann.

Chen, P. P. (1976), "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transactions of Database Systems*, 1 (1), 9-36.

CSC (2001), "Critical Issues of Information Systems Management: 14th Annual Survey of IS Management Issues."

Currim, Faiz (2004), "A Framework for Modeling Business Rules in Conceptual Database Design," University of Arizona PhD Dissertation.

Dowty, D. R. (1986), "The Effects of the Aspectual Class on the Temporal Structure of Discourse," *Linguistics and Philosophy*, 9 (1), 37-61.

Dyreson, Curtis E., William S. Evans, Hong Lin, and Richard T. Snodgrass (2000), "Efficiently Supporting Temporal Granularities," *IEEE Transactions on Knowledge and Data Engineering*, 12 (4), 568-87.

Elmasri, R. and S. B. Navathe (2006), Fundamentals of Database Systems (Sixth ed.). Boston, MA: Addison Wesley.

Franconi, E., A. Giorgi, and F. Pianesi (1993), "Tense and Aspect: A Mereological Approach " in 13th International Joint Conference on Artificial Intelligence Chambery, France

Galton, A. (1991), "A Critical Examination of Allen's Theory of Action and Time " Artificial Intelligence 42, 159–88.

Gregersen, H. and C.S. Jensen (1999), "Temporal Entity-Relationship Models-A Survey," *IEEE Transactions on Knowledge and Data Engineering*, 11 (3), 464-97.

Gregersen, Heidi and Christian Jensen (1998), "Conceptual Modeling of Time-Varying Information," TIMECENTER Technical Report.

Guarino, Nicola and Roberto Poli (1995), "The role of formal ontology in the information technology," *International Journal of Human Computer Studies*, 43 (5-6), 623-24.

Jensen, C. S. and R. T. Snodgrass (1997), "Semantics of Time-Varying Attributes and their use for Temporal Database Design." Tucson, Arizona: TimeCenter Technical Report.

Jensen, C.S., 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 (1998), "A Consensus Glossary of Temporal Database Concepts-February 1998 Version," in Temporal Databases: Research and Practice, O. Etzion and S. Jajodia and S. Sripada, Eds.: Springer-Verlag.

Khatri, Vijay, Sudha Ram, and Richard T. Snodgrass (2004), "Augmenting a Conceptual Model with Geospatiotemporal Annotations," *IEEE Transactions on Knowledge and Data Engineering*, 16 (11), 1324-38.

Khatri, Vijay, Sudha Ram, and Richard T. Snodgrass (2006a), "On Augmenting Database Design-Support Environments to Capture the Geo-Spatio-Temporal Data Semantics," *Information Systems*, 31 (2), 98-133

Khatri, Vijay, Sudha Ram, and Richard T. Snodgrass (2001), "ST-USM: Bridging the Semantic Gap with a Spatio-temporal Conceptual Model," TimeCenter Technical Report TR-64,.

Khatri, Vijay, Sudha Ram, and Richard T. Snodgrass (2002), "Supporting User-defined Granularities and Indeterminacy in a Spatiotemporal Conceptual Model," *Annals of Mathematics and Artificial Intelligence*, 36 (1-2), 195-232.

Khatri, Vijay, Iris Vessey, Sudha Ram, and V. Ramesh (2006b), "Cognitive Fit between Conceptual Schemas and Internal Problem Representations: The Case of Geospatio-Temporal Conceptual Schema Comprehension," *IEEE Transactions on Professional Communication*, 49 (2), 109-27.

Kim, Jinwoo, Jungpil Hahn, and Hyoungmee Hahn (2000), "How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning," *Information Systems Research*, 11 (3), 284-303.

Kim, Young-Gul and Salvatore T March (1995), "Comparing Data Modeling Formalisms," *Communications of the ACM*, 38 (6), 103-15.

Klein, Wolfgang (1994), Time in Language. London: Routledge.

Kowalski, R. and M. Sergot (1986), "A Logic-Based Calculus Of Events " New Generation Computing 4, 67–95.

Lenat, D. B. and R.V. Guha (1994), Building Large Knowledge-Based Systems: Representation and Inference in the Cyc project Addison Wesley Publishing Company.

McCarthy, J. (1968), "Programs With Common Sense," in Semantic Information Processing, L.M. Minsky, Ed. Cambridge, MA: MIT Press.

Moens, Marc and Mark Steedman (1988), "Temporal Ontology and Temporal Reference," *Computational Linguistics*, 14 (2), 15-28.

Moody, Daniel L., Graeme G. Shanks, and Peta Darke (1998), "Improving the Quality of Entity Relationship Models - Experience in Research and Practice," in 17th International Conference on Conceptual Modeling, Tok Wang Ling and Sudha Ram and Mong-Li Lee (Eds.). Singapore.

Nunes, Joan (1990), "Geographic Space as a Set of Concrete Geographical Entities," in NATO Advanced Study Institute on Cognitive and Linguistic Aspects of Geographic Space, David M. Mark and Andrew U. Frank (Eds.). Las Navas del Marques, Spain: Kluwer Academic Publishers.

Parent, C., S. Spaccapietra, and E. Zimanyi (1999), "Spatio-temporal conceptual models: Data structures + space + time," in 7th ACM Symposium on Advances in Geographic Information Systems. Kansas City, USA.

Parsons, Jeffrey and Yair Wand (1997), "Choosing classes in conceptual modeling," *Communication of the ACM*, 40 (6), 63-69.

Ram, S. (1995), "Intelligent Database Design using the Unifying Semantic Model," *Information and Management*, 29 (4), 191-206.

Roddick, J.F. (1995), "A Survey of Schema Versioning Issues for Database Systems," *Information and Software Technology*, 37 (7), 383-93.

Roddick, John F. and Richard T. Snodgrass (1995), "Schema Versioning," in The TSQL2 Temporal Query Language, Richard T. Snodgrass, Ed. Boston: Kluwer Academic Publishers.

Sheth, A. (1995), "Data Semantics: What, Where and How?," in 6th IFIP Working Conference on Data Semantics (DS-6), R. Meersman and L. Mark (Eds.). Atlanta, Georgia: Chapman and Hall.
Shoham, Y. (1987), "Temporal Logics in AI: Semantical and Ontological Considerations " *Artificial Intelligence* 33 (1), 89-104.
Siau, Keng and Xin Tan (2005), "Improving the quality of conceptual modeling using cognitive mapping techniques," *Data & Knowledge Engineering*, 55 (3), 343-65.
Silbershatz, A., H. Korth, and S. Sudarshan (1997), Database System Concepts (Third Edition ed.): WCB/ McGraw Hill.
Snodgrass, R. T., Michael H. Böhlen, Christian S. Jensen, and Andreas Steiner (1996a), "Adding Transaction Time to SQL/Temporal," ISO-ANSI SQL/Temporal Change Proposal.
Snodgrass, R. T., Michael H. Böhlen, Christian S. Jensen, and Andreas Steiner (1996b), "Adding Valid Time to SQL/Temporal," ISO-ANSI SQL/Temporal Change Proposal.
Snodgrass, Richard T. Ed. (1999), Developing Time-Oriented Database Applications in SQL. San Francisco: Morgan Kaufmann Series in Data Management Systems.
Snodgrass, Richard T. (1995), The TSQL2 temporal query language. Boston: Kluwer Academic Publishers.
Snodgrass, Richard T. (1999), Developing Time-Oriented Database Applications in SQL. San Francisco: Morgan Kaufmann Series in Data Management Systems.
Snodgrass, Richard T. and Ilsoo Ahn (1986), "Temporal Databases," IEEE Computer, 19 (9), 35-42.
Terenziani, Paolo and Richard T. Snodgrass (2004), "Reconciling Point-based and Interval-based Semantics in Temporal Relational Databases: A Proper Treatment of the Telic/Atelic Distinction," *IEEE Transactions of Knowledge and Data Engineering*, 16 (5), 540-51.
Terenziani, Paolo and Pietro Torasso (1994), "Towards an Integration of Time and Causation in a Hybrid Knowledge Representation Formalism," International Journal of Intelligent Systems, 9 (3), 303-38.
Tryfona, Nectaria and Christian S. Jensen (1999), "Conceptual Data Modeling for Spatiotemporal Applications," *Geoinformatica*, 3 (3), 245-68.
Tryfona, Nectaria and Christian S. Jensen (2000), "Using Abstractions for Spatio-Temporal Conceptual Modeling," in 2000 ACM Symposium on Applied Computing Applied Vol. 1. Como, Italy: ACM.
Vendler, Z. (1967), "Verbs and Times," in Linguistics in Philosophy. New York, NY: Cornell University Press.
Verkuyl, H.J. (1972), On the Compositional Nature of The Aspects Dorderect Riedel.
Vessey, Iris (1991), "Cognitive Fit: A Theory-based Analysis of Graphs Vs. Tables Literature," *Decision Sciences*, 22 (2), 219-40.
Vila, L. (1994), "A survey on temporal reasoning in artificial intelligence" *AI Communications*, 7, 4-28.
Wand, Yair and Ron Weber (1993), "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars," *Journal of Information Systems*, 3, 217-237.
Webber, B.L. (1988), "Special Issue on Tense and Aspect" Computational Linguistics 14 (2), 1-2.
Witt, G. C. and G. C. Simsion (2004), Data Modeling Essential: Analysis, Design and Innovation (Third ed.). San Francisco, CA: Morgan Kaufmann.
Zimanyi, E., C. Parent, S. Spaccapietra, and A. Pirotte (1997), "TERC+: A Temporal Conceptual Model," in International Symposium Digital Media Information Base.

Appendix

Appendix A

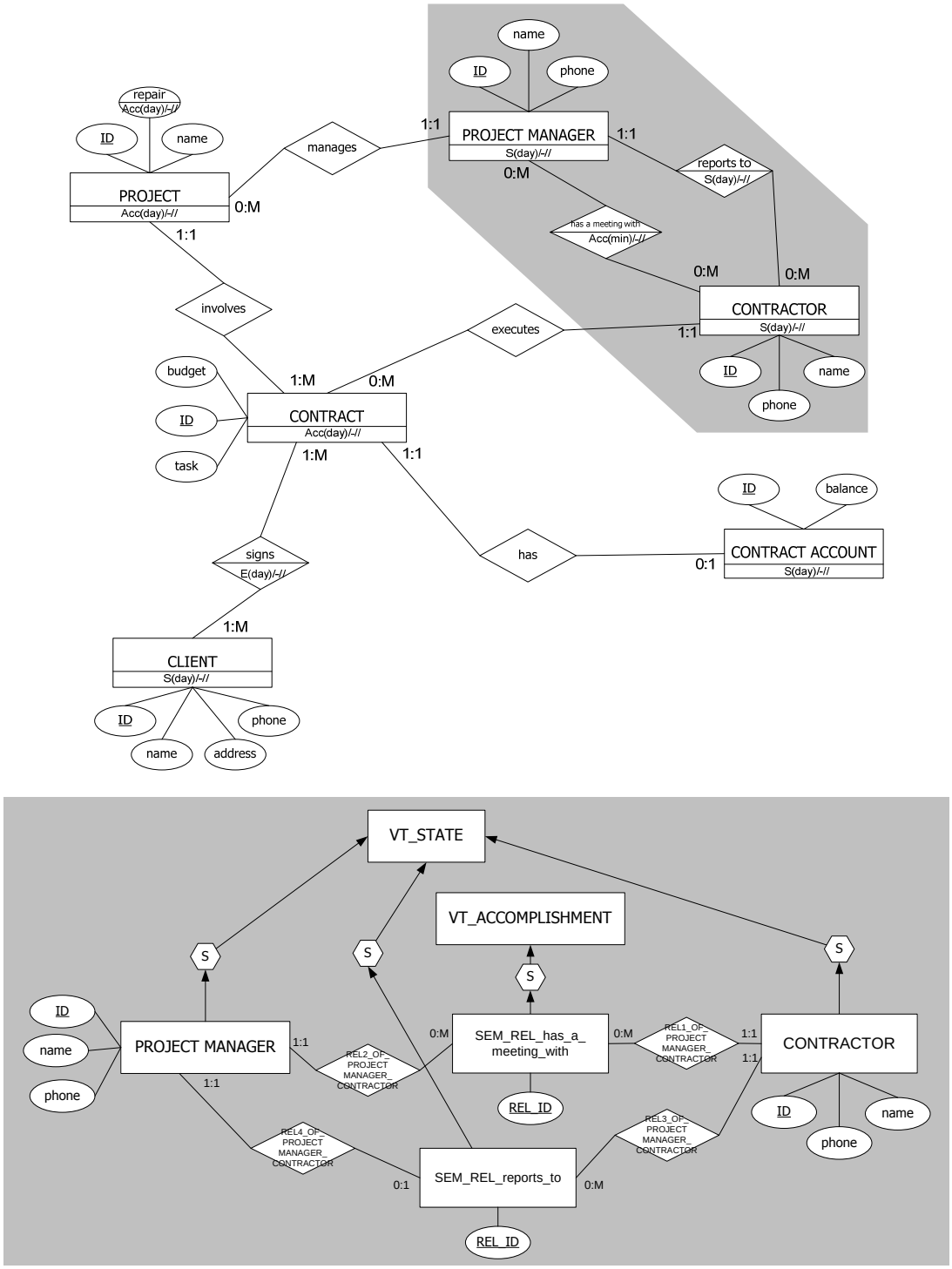


Figure 13: Semantics of the (shaded) fragment of the annotated schema for the CONTRACT application

To explicate the semantics of temporal annotations, the following axioms would be automatically generated as the shaded part of an annotated schema (top of Figure 13) is converted to a translated USM schema (bottom of Figure 13).

Axiom 1

a) $\forall e \in S$(VT_ACCOMPLISHMENT), $\forall p \in$ VT_ACCOMPLISHMENT(e, telic_period), begin(e, p) < end(e, p)

b) $\forall e \in S$(VT_STATE), $\forall p \in$ VT_STATE(e, maximal_ period), begin(e, p) < end (e, p)

c) $\forall e \in S$(VT_STATE), $\forall p_1, p_2 \in$ VT_STATE(e, maximal_ period), begin(e, p_1) < begin(e, p_2) $\Rightarrow$ end (e, p_1) < begin (e, p_2)

Axiom 2

$\forall e \in S$(VT_EVENTUALITY),

VT_EVENT(e, time_point)$\Rightarrow$ VT_EVENTUALITY(e, hold) = to_element (VT_EVENT(e, time_point)) $\wedge$
VT_STATE (e, maximal_period) $\Rightarrow$ VT_EVENTUALITY(e, hold) = VT_STATE(e, maximal_period) $\wedge$
VT_ACCOMPLISHMENT (e, telic_period) $\Rightarrow$
VT_EVENTUALITY(e, hold) = VT_ACCOMPLISHMENT(e, telic_period)

a) VT_Down

$\forall e,\ p_1, p_2$, VT_STATE(e) $\wedge$ VT_STATE.hold(e, p_1) $\wedge$ $p_2 \subseteq p_1 \Rightarrow$ VT_STATE.hold(e, p_2)

b) VT_Up

$\forall e,\ p_1, p_2$, VT_STATE(e) $\wedge$ VT_STATE.hold(e, p_1) $\wedge$ VT_STATE.hold(e, p_2) $\wedge$
(MEETS(p_1, p_2) $\vee$ MEETS^{-1}(p_1, p_2) $\vee$ OVERLAPS(p_1, p_2) $\vee$ OVERLAPS^{-1}(p_1, p_2)
$\vee$ FINISHED(p_1, p_2) $\vee$ FINISHED^{-1}(p_1, p_2) $\vee$ DURING(p_1, p_2)
$\vee$ DURING^{-1}(p_1, p_2) $\vee$ STARTS(p_1, p_2) $\vee$ STARTS^{-1}(p_1, p_2) $\vee$ EQUALS(p_1, p_2)
) $\Rightarrow$ VT_STATE.hold(e, $p_1 \cup p_2$)

Axiom 3

a) $\forall e \in S$ (CONTRACTOR), CONTRACTOR.VT_STATE.VT_EVENTUALITY.VT_has.TEMPORAL_GRANULARITY(e, name) = day

b) $\forall e \in S$ (PROJECT MANAGER), PROJECT MANAGER.VT_STATE.VT_EVENTUALITY.VT_has.TEMPORAL_GRANULARITY(e, name) = day

c) $\forall e \in S$ (has a meeting with), has a meeting with.VT_ACCOMPLISHMENT.VT_EVENTUALITY.VT_has.TEMPORAL_GRANULARITY(e, name) = minute

d) $\forall e \in S$ (reports to), reports to.VT_STATE.VT_EVENTUALITY.VT_has.TEMPORAL_GRANULARITY(e, name) = day

Axiom 4

a) $\forall e_1 \in S$(PROJECT MANAGER), $\forall e_2 \in S$(name), SEM_ATTR_PROJECT MANAGER_name (e_1, e_2) $\Rightarrow$ ATTR_OF (e_2, e_1)

b) $\forall e_1 \in S$(PROJECT MANAGER), $\forall e_2 \in S$(phone), SEM_ATTR_PROJECT MANAGER_phone (e_1, e_2) $\Rightarrow$ ATTR_OF (e_2, e_1)

c) $\forall e_1 \in S$(CONTRACTOR), $\forall e_2 \in S$(name), SEM_ATTR_CONTRACTOR_name (e_1, e_2) $\Rightarrow$ ATTR_OF (e_2, e_1)

d) $\forall e_1 \in S$(CONTRACTOR), $\forall e_2 \in S$(phone), SEM_ATTR_CONTRACTOR_phone (e_1, e_2) $\Rightarrow$ ATTR_OF (e_2, e_1)

e) $\forall e, \forall a, p_1,\ e$.VT_STATE $\wedge$ (e.VT_STATE(hold) = p_1) $\wedge$ ATTR_OF(a, e) $\wedge$
(a.annotations$\neq$Acc $\wedge$ a.annotations$\neq$E $\wedge$ a.annotations$\neq$S)
$\Rightarrow a \in$ S(VT_STATE) $\wedge$ a.VT_STATE(hold) = p_1

Appendix B

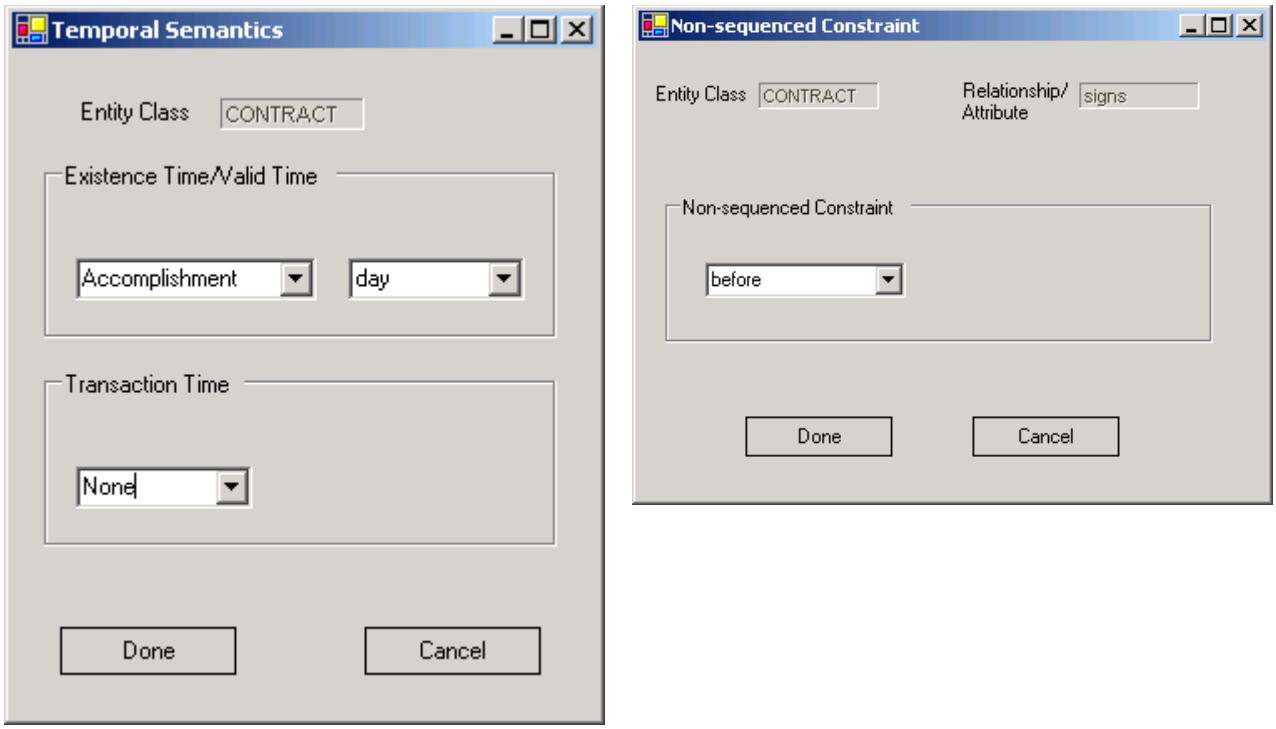


Figure 14: An example of specifying temporal annotations and non-sequenced temporal constraints