Modeling Time from a Conceptual Perspective

Stefano Spaccapietra
Database Laboratory
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland
Tel :+41 21 693 52 10
Stefano.Spaccapietra@epfl.ch

Christine Parent
HEC/INFORGE
Université de Lausanne
1015 Lausanne, Switzerland
christine@lbdsun.epfl.ch

Esteban Zimanyi
INFODOC CP 175/02
Université Libre de Bruxelles
1050 Bruxelles, Belgique
Tel :+32 2 650 39 50
ezimanyi@ulb.ac.be

1. ABSTRACT

Although many temporal models have been proposed in the literature, there is still need for a conceptual model capturing the essential semantics of time-varying information that is free of implementation concerns. This paper first discusses important criteria to be considered when assessing the "conceptual" quality of a temporal model. Then, it presents the main temporal features of MADS, a spatio-temporal conceptual model. The focus is on identifying issues that are either open or controversial, and discussing alternatives, if any. Finally, it is shown how the model may be implemented on top of either TSQL2 or a traditional, non-temporal data model.

1.1 Keywords

Temporal databases, conceptual modeling, data semantics

2. INTRODUCTION

In many application domains complex decision making involves analysis of spatial and temporal data. Thus, there is an increasing demand for strong computer support in spatio-temporal data management. In particular, the capability to plan for the future based on past experiences is essential. This calls for powerful temporal data modeling and temporal reasoning facilities. Many examples of desirable facilities may be found in the rich body of literature devoted to temporal databases. Bibliographies (e.g., [25]), surveys (e.g., [19]), and books [7][23] gathered more than a thousand papers. However, very little of the outcome had an impact on software providers, and current commercial DBMSs are only beginning to offer time-related functions, e.g., the temporal datablades of ILLUSTRA.

In [18] are identified some of the reasons behind this apparent failure in know-how transfer. The most evident drawback is the high number of approaches, making no single one emerge from the heap. It is also noted that "there seems to be a gap between the goals assumed by the temporal database community and the needs...", and "users could not say what a temporal database is, nor even begin to comprehend how it could be of service to their applications". A major conclusion was that "... the time-varying semantics is obscured in the representation schemes by other considerations of presentation and implementation", which lead the authors to "advocate a separation of concerns, i.e., adopting a very simple *conceptual* data model...".

One reason for this mismatch is that temporal research has mainly investigated issues at the logical or implementation levels, while user requirements are at the conceptual level. Focusing on conceptual models is indeed the right move towards a better dissemination of temporal databases.

Conceptual modeling offers two significant advantages with respect to relational or object-oriented design: (1) it allows to focus on the representation of application data and processes with minimal concern for technical constraints, and (2) its result (a conceptual schema of data and processes) is more stable than implementation-oriented schemas, which must be changed whenever the target platform changes.

Conceptual models also provide better support for visual user interfaces. Entity-relationship (ER) models, for instance, have been very successful with users. Recent work has shown that conceptual models also support direct manipulation techniques (e.g., point, click, and drag elements on a screen) to browse the database, and to express queries and updates without the burden of the complex syntax of a textual language [4][8]. Finally, conceptual modeling facilitates information exchange over the Internet and within heterogeneous distributed or federated databases. In such contexts, a conceptual model provides the best vehicle for a common understanding among partners with different technical and application backgrounds.

Several ER and object-oriented (OO) temporal models have been reviewed in [10][18][19], and other models have been proposed since (e.g., [2][22]). All of these models show weaknesses with respect to the goals of conceptual modeling. This paper proposes an approach focusing on well-known conceptual modeling principles, and where semantic criteria are used as a sound basis for comparing temporal models. Then, we propose a model, called MADS (for Modeling of Applications with Spatio-temporal features), which explicitly meets these criteria. The current version of MADS supports basic spatio-temporal features needed by the applications we have been involved with, mainly related to land management or utility networks. A case study allowed to quantify the benefits of using MADS with respect to a traditional ER model. Implementation of MADS in operational environments has shown that it can be used as a front-end to existing systems. A visual schema editor has been implemented allowing designers to define their MADS schemas through direct manipulation on a screen. A visual query language is planned to complement data definition with data manipulation facilities based on the same modeling paradigm.

Section 3 introduces the generic criteria on which we built the MADS model. We show that, although obvious, these criteria are not always satisfied in other proposals. The following sections discuss issues for defining a temporal conceptual model. Section 4 discusses timestamping of objects, attributes, and relationships. Section 5 develops dynamic temporal relationships. Section 6 relates on the implementation of our conceptual model. Finally, section 7 concludes by pointing at work in progress within a global spatio-temporal framework. In the paper we only consider valid time as this is the primary goal of the users we are working with.

3. CRITERIA FOR CONCEPTUAL MODELS AND RELATED WORK

This section recalls some overall goals that a conceptual model (CM) should satisfy. Such goals, which guided our design of the MADS model, may also serve to assess the conceptual quality of a given proposal. In [10] are identified other 19 design criteria, but these served a different, classification purpose.

Conceptual. A CM must provide a direct mapping between the perceived real world and its computer representation. Thus, it should be free from implementation-based limitations and from modeling tricks. This criterion alone eliminates a number of proposals. Unnecessary limitations include restraining relationships to be binary and attributes to be monovalued. Modeling tricks typically consist in introducing artificial object types (i.e., those not representing a real-world element) to cope with some modeling limitation. Known examples include liaison records, intersection classes, and objectified attributes. A spatio-temporal modeling trick is representing spatial or temporal features of application object types by linking them, using a relationship type, to an artificial space or time object type (e.g., Point, Line,..., Date, Timestamp,...). Some temporal proposals follow this approach and are therefore inadequate. It is worthwhile noting that showing these space and time object types on schemas induces an explosion in the number of links that will obscure any diagram representing a real application.

Powerful. How much expressive power a data model should support is a subject for an ever-lasting debate. Relying on general consensus seems therefore an adequate pragmatic rule. The current level of consensus (as illustrated by, e.g., ODMG's ODL [5] or UML [3]) includes support of object types, explicit relationship types, multivalued attributes, complex attributes (i.e., attributes composed of other attributes), is-a links, aggregation (part-of) links, and the associated integrity constraints. Although many more features have been proposed, past experience has shown that striving for the highest expressive power leads to unbearable complexity and eventually results in rejection of the model.

Simple. One of the major advantages of a CM is to allow users to get involved in the design of the application, which is known to be an essential success factor. Studies by ergonomists have assessed that for a model to be understandable by users with only minimal training, the number of concepts has to be kept small. Further, they have to have a clean definition, as close as possible to the real-world concepts they represent. Models with too sophisticated constructs or constructs whose semantics deviates from well-established ones are likely to be discarded by users.

Visual. Because most of the design work is based on schema diagrams, a CM has to be supported by clean, intuitive, visual notations. Few existing proposals adhere to this obvious principle. Some use abstract, unintuitive notations, others offer only a textual syntax, while in others the same semantics is conveyed by different notations depending on the context. In our proposal we simply use a clock icon to denote timestamping of constructs, whatever they are. A visual schema editor is also a must [10].

Formal. For specifications to be unambiguous, they have to rely on a sound formal definition. As pointed out in [10], several proposals lack such a sound background.

Associated data manipulation language. While in the past users have been trained to a schizophrenic approach

(ER diagrams for schema definition and relational SQL for data manipulation), there is no reason for this to continue. Users should be entitled to use a single paradigm for both the DDL and the DML. An example of associated DML is provided in [9]. Comparisons of temporal languages may be found in [6][19].

Temporality. A temporal CM should fulfill the above criteria, while providing temporal support. The comprehensiveness of the model (i.e., how many features it supports) is a matter of how rich the model is, but not necessarily how good it is. Required features vary from an application domain to the next. Basic temporal features include timestamping of attributes, objects, and relationships, based on a discrete time scale. At least valid time should be supported, as well as timestamping with periods and with instants. Other important features identified in [18] are: multiple time granularities, future time, imprecise time, relative time, branching time, coexistence of temporal and non-temporal data, transaction time, support for temporal reasoners. Recent developments in video databases call for support of continuous time.

Having stated the goals, how to achieve them, in particular the apparently contradictory goals of being powerful and simple? Experience in conceptual modeling has shown that there is one golden rule to get power at a minimal cost: **orthogonality**. Orthogonality among different modeling dimensions (data structures, time, space) means that: (1) each dimension is defined independently of the others (achieving simplicity), and (2) concepts from different dimensions can be combined in any meaningful way (giving expressive power). Most temporal models are rather orthogonal, as time can be associated with objects, relationships, and attributes, and their behavior is relatively clear (but temporality is seldom applied to methods or integrity constraints). The setting is different for spatial databases, where proposed solutions clearly ignore orthogonality [17]. Orthogonality is also the key for being able to customize the comprehensiveness of the model to different application domains without having to redesign the entire model.

In summary, none of the models we have examined satisfies all of the above goals. This actually prompted the development of MADS, an object+relationship conceptual model. The major originality of MADS lies in that it is a purely conceptual model, although translators of MADS to operational database models are also defined. In the structural dimension it includes multivalued and complex attributes, derived attributes, methods, integrity constraints, n-ary relationships, is-a links, and aggregation links. Space and time dimensions have been orthogonally added to the existing structural dimension. In this paper we describe the temporal features of MADS, the spatial features are described in a companion paper [17].

MADS currently supports valid time and user-defined time at different granularities, and is being extended for transaction time. Temporality can be attached to object and relationship types, to attributes, methods, and integrity constraints. An original contribution of MADS is to highlight that some commonly accepted constraints in existing models do not always fit application requirements, e.g., allowing non-temporal object types related by temporal relationships and vice versa. Moreover, MADS includes dynamic relationships for describing **inter-object dynamics** in which time plays an important role: e.g., this land parcel was created by merging those land parcels, this town represents a further development stage of that village, this storm preceded that landslide. To the best of our knowledge previous temporal models do not support these dynamic relationships.

MADS is supported by a visual schema editor. The spatial and temporal characteristics of an application can be immediately apprehended thanks to appropriate icons visualized on schema diagrams.

4. ISSUES IN THE DESIGN OF A TEMPORAL CONCEPTUAL MODEL

Timestamping is the traditional way of modeling temporal information. Applied to values, timestamping allows expressing when a value was, is, or will be holding in the real world (valid time) or when it was known in the database (transaction time). Timestamping also applies to objects and relationships to express information on their life cycle: when an object or relationship was created, suspended, reactivated, or deleted. Object and relationship timestamps are also based on either valid time or transaction time. As these mechanisms are well known, this section focuses on open issues deserving discussion.

4.1 Timestamping Attributes

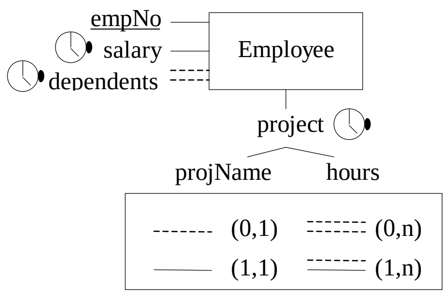

Figure 1: Timestamped attributes.

Timestamped attributes record the evolution of their values. We use the term *evolution* (instead of *history*) since past, present, and future values are recorded. Conceptually, a timestamped attribute is a set of partial functions linking a validity period of the attribute to its value domain. There are one such function and its associated validity period for each instance of the owner of the attribute (entity, relationship instance, object attribute). The validity periods are sets of disjoint intervals defined according to the granularity specified by the designer for the attribute. As shown in Figure 1, any attribute may be timestamped, whether simple or complex, mono- or multivalued. For the complex attribute project, composed of projName and hours three alternative design decisions are possible.

1) If project is timestamped and its components not (as in Figure 1), this allows keeping the evolution of projects an employee works for. Values of the attribute are (projName, hours) pairs with associated timestamp.
2) If hours is timestamped and project not, this allows keeping the evolution of hours spent by an employee on the project s/he is currently working. Notice that when an employee is moved to another project, the evolution of hours in the previous project is lost, as only the current project is recorded (project is a non-temporal attribute). This stresses that timestamping an attribute does not necessarily mean that its values are kept forever in the database.
3) If both hours and project are timestamped this allows keeping the evolution of both projects and hours in a project. Thus, it can be represented that an employee works on project MADS from January 97 until December 1998, at 35 hours per week in 97, then at 20 hours per week in 98.

Attribute cardinalities are interpreted as **static**, i.e., they define the number of attribute values at any point in time. T**emporal cardinalities**, written *h(min,max)*, allow to constrain the minimum and the maximum number of values that a timestamped attribute can take over the life cycle (or validity period) of its owner. In case of a non-timestamped owner, this life cyle is assumed to be]-∞ ,+∞[. For example, a temporal cardinality *h(1,3)* on project states that an employee is attached to at least one and at most 3 projects altogether during his/her life cycle.

Existing temporal models impose constraints among timestamps in composition structures. The latter include a timestamped attribute of a timestamped object (or relationship) type, or a timestamped component of a timestamped complex attribute. Examples of such constraints are:

- the validity period of an attribute must be within the life cycle of the object it belongs to (e.g., [26]), and
- the validity period of a complex attribute is the union of the validity periods of its components (e.g., [9]).

Although both rules rely on the intuitive idea that the lifespan of a component is included in the lifespan of the composite, this is not necessarily the case. For example, employee records (where an employee's life cycle represents the period when s/he is working for the company) may keep track of positions occupied by the employee before joining the company. Similarly, in Figure 1 an employee may work on a project before it comes into existence (as is the case for ESPRIT projects), or continue working on a project after it has officially ended.

MADS **does not enforce** any a priori constraint on timestamps. However, constraints may be explicitly defined either using temporal expressions based on a calculus that includes Allen's operators [1], or referring to predefined constraint types, such as inclusion, covering, and equality. For example, if in Figure 1 both project and hours are timestamped, an **inclusion** of hours into project enforces that the temporal elements of values of hours are included in the temporal element of the associated project. A **covering** of project by hours enforces that at any instant included in the temporal element of a project, there is an associated value of hours. Finally, an **equality** constraint on hours and project states that the unions of the temporal elements of the values for both attributes are equal.

MADS approach (i.e., no inherent constraint) achieves orthogonality, as timestamping of an attribute does not depend on where the attribute is located in the data structure.

4.2 Timestamping object types

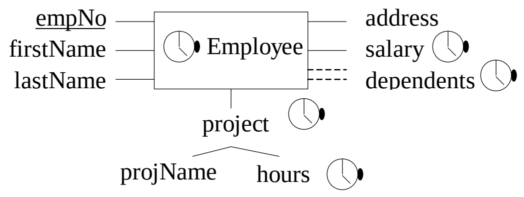

Figure 2: Timestamped object type.

Timestamping an object type allows to keep the life cycle of its instances: when they are created, suspended, reactivated, or deleted. In many temporal models the life cycle of an object is assumed to be a single, continuous time interval, to avoid "the problem of maintaining identity across disjoint periods of existence" [26], i.e., an implementation concern. MADS allows the membership of an object in an object type to be suspended and reactivated. This allows describing situations like a professor leaving for a sabbatical. Consequently, a life cycle is a function linking the time domain]-∞,+∞[to the four possible status "up to now does not exist", "active", "suspended", or "dead".

Defining an object type as timestamped is independent of defining some of its attributes as timestamped. As already said, although MADS does not impose any restriction on the timestamp of an object and those of its attributes, temporal integrity constraints may be defined if needed by the application.

The notion of identifier must be revisited for timestamped object types. If in Figure 2 empNo is an identifier of Employee, two interpretations are possible: (1) an empNo value designates one employee at any point in time, but the same empNo may designate different employees at different points in time; (2) an empNo value will only ever be associated with one employee. MADS adopts the latter interpretation as default.

4.3 Timestamping and generalization

It is worthwhile exploring how generalization and timestamping interact. MADS supports is-a links with the usual semantics of extension inclusion, substitutability, and property inheritance (where refinement and redefinition are possible for properties). Inheritance also applies to timestamps.

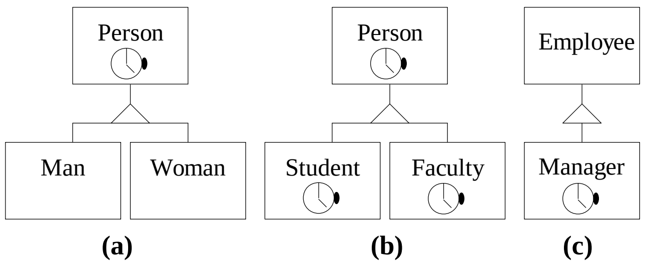

Figure 3: Timestamped object types and generalization.

Classification of objects in a generalization hierarchy may be of two kinds. A cluster of sub-types is **static** if an object cannot change its sub-type; it is **dynamic** otherwise.

In a static cluster the life cycle of an object in a sub-type is the same as its life cycle in the super-type. In Figure 3 (a) persons are classified by gender and there is no expectation that a person changes his/her gender.

The case is different in a dynamic cluster. Figure 3 (b) shows a typical example related to roles a person may play. Clearly, the life cycle of Mary as a student is different from her life cycle as a person or as a faculty. As shown in the figure by the repetition of the clock icon, a redefinition of the timestamp is needed for keeping track of the life cycle specific to a sub-type. Obviously, the timestamp in a sub-type must be included in the timestamp of the super-type. This is the only case where a timestamp is inherently constrained by another.

A sub-type may be timestamped while its super-type is not. Figure 3 (c) shows a design decision to keep track of past, present, and future managers, while being interested in only current employees. This requires inheritance of properties to be materialized in manager objects so that, when an employee object is deleted the inherited information is kept within the dead manager object.

Constraints applicable to generalization hierarchies (e.g., partitioning, covering, disjunction) naturally extend to timestamps. Migration of objects in a dynamic generalization hierarchy is discussed in Section 4.

4.4 Timestamping relationship types

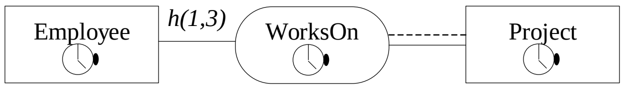

Figure 4: Timestamped relationship.

Timestamping a relationship type involves keeping track of the life cycle of its instances. Relationships can be created, suspended, reactivated, and deleted. In Figure 4, past, present, and future instances of relationship WorksOn are kept in the database.

As for object types, timestamped relationships can have attributes (timestamped or not). As for attributes, static cardinalities, linking object types to relationship types, may be complemented with temporal cardinalities [24].

To prevent dangling references (i.e., a relationship linking nonexistent objects), most temporal models impose that a timestamped relationship type can only link timestamped object types. Moreover, they also constrain the life cycle of relationship instances so that a relationship can only exist if the linked objects also exist at the same time. In MADS there are no such implicit constraints, although they can be explicitly stated if needed through predefined temporal integrity constraints.

MADS allows a timestamped relationship type to link non-timestamped object types. If WorksOn is timestamped and Employee and Project not, the effect is to keep all past, present, and future instances of WorksOn concerning currently valid employees and projects. As usual, deleting an employee or a project induces the deletion of the WorksOn instances in which it participates. Also, a non-timestamped relationship type may link timestamped object types. This would allow keeping track of all past, present, and future employees and projects, while only keeping currently valid WorksOn assignments.

Finally, MADS does not constrain the life cycle of relationships. They may link objects that do not exist simultaneously, e.g., in a relationship linking the author of a biography to the personality the biography is about.

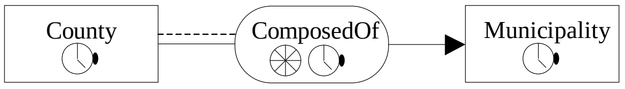

Figure 5: Timestamped aggregation.

MADS also supports aggregation relationships. Figure 5 shows an example of a timestamped aggregation, modeling the administrative partitioning of a country (e.g., Switzerland) as a set of counties, each one decomposed into municipalities. The object types and the aggregation link are timestamped to keep track of evolution: e.g., new counties or municipalities are created, while other ones disappear; also the membership of municipalities in counties varies with time, e.g., as a result of a referendum.

5. MODELING DYNAMIC ASPECTS

MADS provides four dynamic relationship types allowing the description of inter-object dynamics where time plays an essential role. They are described in the following sections. As every relationship type, they may be named, timestamped, have attributes and methods (timestamped or not), and participate in derivation formulas and integrity constraints. Some of these links have been discussed in [11][12][15].

5.1 Transition relationship

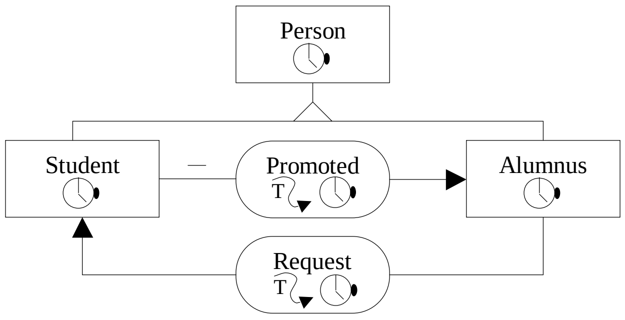

Figure 6: Transition relationship.

A transition relationship models the migration of objects from a source to a target object type. It conveys a *becomes-a* semantics. Since it is a dynamic link between objects sharing the same identity, the source and the target object types must be members of the same dynamic generalization hierarchy. Transitions are most frequently timestamped and they can link non-timestamped objects.

There are two types of transition: **evolution**, when the transition object ceases to be an instance of the source object type, and **extension** otherwise. In both cases, an instance of the transition relationship is created either explicitly by the user, or whenever it can be deduced automatically. An example of the latter is the case of a specialization defined by predicates on attribute values.

Figure 6 shows two examples: an evolution from Student to Alumnus (symbolized by the minus sign), and an extension from Alumnus to Student, meaning that an alumnus may become a student while still belonging to the Alumnus class. Since both transitions are timestamped, the time at which a transition takes place is also kept.

5.2 Generation relationship

Generation relationships represent processes that lead to the emergence of new objects: an instance (or a set of instances) of a source object type(s) generate(s) an instance (or a set of instances) of a target object type(s). Generations are n-ary relationships conveying a *yields* semantics. They are useful for modeling causal and temporal relationships involved in the appearance and disappearance of real-world objects.

The source objects of a generation can be preserved or consumed. A **transformation** occurs when all the instances of the source object types are consumed in the generation process. A **production** takes place when all the source instances survive the generation process. Notice that it is also possible that some source objects are preserved while others are consumed.

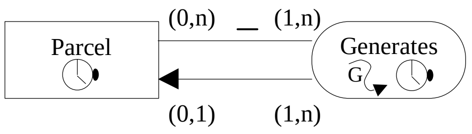

Figure 7: Generation relationship.

In a cadastral application, a parcel may be split generating several smaller parcels, or on the contrary, several parcels may be merged and generate a bigger one. In Figure 7, the transformation relationship (symbolized by the minus sign) keeps track of which parcels give rise to other parcels as well as the period at which the generation takes place (since the generation is timestamped).

Like transitions, generations are most likely timestamped, and also apply to non-timestamped objects. As shown in Figure 7, they bear cardinalities on both sides of the roles, expressing: (1) how many generation instances can be linked to an object (i.e., the standard definition of relationships cardinalities) and (2) how many objects can be linked to a generation instance.

5.3 Coalescence

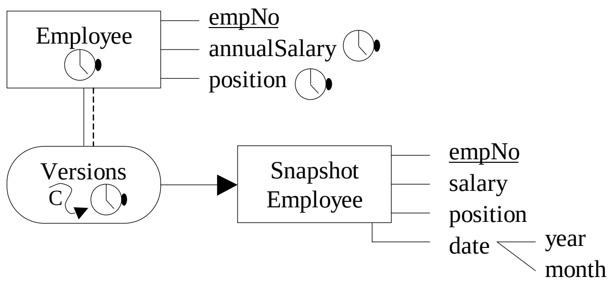

Figure 8: Coalescence.

Coalescence is a special case of aggregation with the meaning that the component objects are snapshots of the composite object. Derivation formulas may specify how the life cycle and/or attributes of the composite object are derived from the component. In Figure 8 the life cycle and attributes annualSalary and position of an instance e of Employee are derived from the instances s of SnapshotEmployee related to e by the coalescence Versions.

5.4 Timing relationship

Timing relationships allow specifying constraints on the life cycles of the participating objects. They convey useful information even if the related objects are not timestamped. They allow in particular to express constraints on schedules of processes. Figure 9 shows a timing relationship Produces between Project and Deliverable. A temporal operator *during* qualifies the relationship to state that the life cycle of any instance of Deliverable has to be included in the life cycle of the related instance of Project.

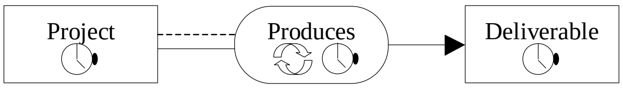

Figure 9: Timing relationship.

Any formula built on Allen's temporal operators may qualify a timing relationship. Allen's operators are extended to the comparison of complex life cycles (i.e., temporal elements for the four status) in two main ways: comparing the entire life cycle (i.e., timestamps of birth and death) or comparing only active periods.

6. IMPLEMENTATION

MADS has been implemented by mapping its specifications into those of operational database models. This section describes two typical mappings: the first one is onto a non-temporal model, ERC+ [21], to show how time-related aspects are turned into classical structures; the second one is on the temporal language TSQL2 [20]. Other mappings that have been realized are to the IE Composer tool [13], and to the Swiss Interlis standard [14] for GIS data interchange.

6.1 Mapping into ERC+

ERC+ and MADS data structures are equivalent; in particular both support complex and multivalued attributes, n-ary relationships, is-a and aggregation links. Choosing ERC+ as target classical data model allows focusing on representation issues for temporal specifications. Moreover, algorithms mapping ERC+ or a similar model into an object-oriented model (e.g., ODMG's ODL) or a relational model are well known.

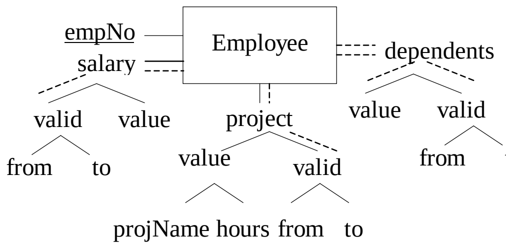

Figure 10: Mapping timesamped attributes in ERC+.

Figure 10 shows the mapping of the timestamped attributes of Figure 1. A timestamped attribute a at any level, with cardinalities *(i,j)* and *h(c,d)*, is replaced by a complex attribute a with cardinality *(x,d)*, composed of attributes value*(1,j)* and valid*(1,n)*, where *x*=0 if *c*=0, else *x*=1. The valid attribute represents the temporal element associated to the value. It is a complex multivalued attribute composed of from and to, which are defined according to the granularity of the original a attribute.

Constraints on values of from and to cannot be represented in the corresponding ERC+ schema and should be enforced separately. Such constraints include:

- the values of valid.from and valid.to define disjoint intervals;
- the intervals specified for mandatory attributes, such as address and projects, should be contiguous; and
- temporal constraints (e.g., inclusion or covering) over complex timestamped attributes composed of other timestamped attributes.

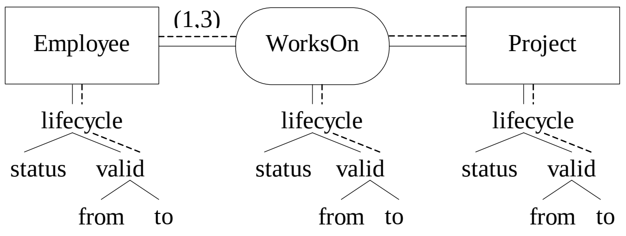

Figure 11: Mapping timestamped object and relationship types in ERC+.

Figure 11 shows the mapping of the timestamped object and relationship types of Figure 4. In both cases, a complex multivalued attribute lifecycle is added with components status (of domain {does not exist, active, suspended, dead}) and valid. The valid attribute represents the temporal element associated to the status.

Notice that cardinalities (0,1) and (1,1) of timestamped relationships become (0,n) and (1,n), respectively, in the corresponding ERC+ schema. For example, in Figure 11, Employee participates in WorksOn with cardinality (1,3). Indeed, since WorksOn is a timestamped relationship with temporal cardinality h(1,3), up to three past, present, and future instances of the relationship are kept. Thus, an additional constraint stating that "for any employee the active periods of all his WorksOn relationships must be disjoint" is needed in Figure 11 to express the static (1,1) cardinality in the original MADS schema of Figure 4.

Dynamic relationships are translated into regular relationships, with a system attribute TYPE defining the type of the relationship: transition, generation, coalescence, or one of the predefined timing relationships (or the formula given by the designer to define his own timing relationship). Constraints implement the semantics of the relationships. For example, a *during* relationship generates a temporal integrity constraint specifying that the temporal element associated to the active status of the first object is included in that of the second object.

6.2 Mapping onto TSLQ2

Since TSQL2 [20] is based on the relational model, well-known problems of representing richer conceptual schemas into relational schemas also apply for TSQL2. In particular, TSQL2 has no support for complex or multivalued attributes. Also, both object and relationship types should be mapped into a single concept, the table. Further, since TSQL2 is a tuple-timestamped temporal model, there is no direct way to represent in the same table several timestamped attributes. The general approach for mapping a timestamped object or relationship type into TSQL2 is as follows:

- A main table stores the life cycle of the object or relationship. This table also stores non-timestamped monovalued simple attributes (or simple monovalued components of complex monovalued attributes).
- One additional table for each timestamped monovalued attribute. For timestamped complex attributes having timestamped components, one table is needed for each timestamped level.
- One additional table for each multivalued attribute, whether timestamped or not. Notice that timestamped multivalued attribute values are represented by nesting on the atomic values.

For example, the object type of Figure 2 can be mapped in TSQL2 in the following set of tables

Employee(<u>empNo</u>,firstName,lastName,address)

EmpSalary(<u>empNo</u>,salary)

EmpDependents(<u>empNo,depend</u>)

EmpProject(<u>empNo,</u>projName)

EmpProjHours(<u>empNo,</u>projName,hours)

where all tables are valid state tables with the appropriate granularity. Table Employee stores the life cycle of employees, table EmpProject stores the validity of project while EmpProjHours keeps the validity of hours. A value for dependents such as {{John,Mary}@[9/97,12/97], {John,Mary,Peter}@[1/98,now]}} is represented by 3 tuples in EmpDependents: John and Mary with validity [9/97, now] and Peter with validity [1/98,now].

The above TSQL2 schema may be optimized if additional temporal constraints are known. For example, if there is an equality on project and hours, meaning that at every point in time that the employee was attached to the project there is a corresponding value for hours and vice versa, then tables EmpProject and EmpProjHours in the above schema could be replaced by

EmpProjHours1(<u>empNo,</u>projName,hours)

where the table keeps the validity of hours and the validity of the project can be obtained by the restructuring operator (i.e., temporal projection) in TSQL2 queries.

7. CONCLUSION

Although impressive research efforts have been done in the field of temporal databases, current results have produced little impact both on commercial DBMSs and in the users' world. We agree with [18] that the main reason for this situation is the lack of a temporal *conceptual* model free of implementation concerns. In this paper we discussed important issues for defining a temporal conceptual model. We have shown how the temporal models proposed so far fail to match all of the criteria for a conceptual model, although such criteria basically state general and well-known principles. We have also shown that existing models tend to impose unnecessary constraints on temporal features. The MADS model, whose temporal component is presented here, has been specifically designed as a conceptual model for spatio-temporal data. It meets the criteria that have been identified, and imposes almost no constraint on the usage of temporal constructs. Moreover, it includes facilities for explicit modeling of various types of relationships expressing temporal inter-object dynamics.

MADS was developed in an application framework and has been used for modeling several real-world applications: oil management in Colombia, management of the networks of clear and used waters of the Geneva city, study of the evolution of the watershed of the upper part of the Sarine river, and the management of water resources of the Vaud county. The users' feedback received from using MADS in these applications is very encouraging. In the water management case we were able to measure the benefit of using MADS instead of a traditional ER model, in terms of simplicity of the schema: MADS remodeling reduced the number of object and relationship types by a factor of 23% [16]. Moreover, using MADS lead the application designers to discover the importance of temporal information within their application.

Ongoing work to further extend the temporal features of MADS includes:

- Supporting imprecise time, future time, relative time, and branching time.
- Bridging the gap to legacy applications. This is crucial since most applications already manage temporal information in an informal and ad hoc manner.
- Integration of databases with different time granularities, as well as coexistence of temporal and non-temporal data. Solving such heterogeneity issues is important in particular for geographical databases, where reuse is a must due to the cost of acquiring spatial information.

- Definition of a visual data manipulation language for expressing temporal queries in the conceptual paradigm, as well as a language for expressing temporal integrity constraints. Although there is an abundant literature in temporal languages and temporal logics, there is still an important need for more user-oriented languages.

All of these will be validated by handing over our results to users and getting their feedback from experimentation in real applications.

8. ACKNOWLEDGMENTS

This work has been supported by the Swiss Commission for Technology and Innovation. It benefited from cooperation with the Administration Cantonale Vaudoise and Texas Instruments Software.

9. REFERENCES

[1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11): 832-843, 1983.

[2] E. Bertino, E. Ferrari, and G. Guerrini. T_Chimera: A temporal object-oriented data model. *Theory and Practice of Object Systems*, 3(2):103-125, 1997.

[3] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language: User Guide*. Addison-Wesley, 1998. To appear.

[4] T. Catarci and M.F. Costabile. Visual Query Systems. *Journal of Visual Languages and Computing*, 7(3): 243-245, 1996.

[5] R.G.G. Cattell and D.K. Barry (Eds.). *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann Publishers, 1997.

[6] J. Chomicki. Temporal query languages: A survey. In *Proc. of the 1st Int. Conf. on Temporal Logic*, 506-534. Springer-Verlag, LNAI 827, 1994.

[7] J. Clifford and A. Tuzhilin. *Recent Advances in Temporal Databases. Proc. of the Int. Workshop on Temporal Databases*. Springer-Verlag, Zurich, Switzerland, 1995.

[8] Y. Dennebouy et al. SUPER: Visual interfaces for object + relationship data models. *Journal of Visual Languages and Computing*, 6(1):73-99, 1995.

[9] R. Elmasri, G. Wuu, and V. Kouramajian. A temporal model and query language for EER databases. In [23], 212-229.

[10] H. Gregersen, C.S. Jensen, and L. Mark. Evaluating Temporally Extended ER Models. In *Proc. of the 2nd CAiSE97 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Barcelona, Spain, 1997.

[11] R. Gupta and G. Hall. An abstraction mechanism for modeling generation. In *Proc. of ICDE'92*, 650-658, Tempe, Arizona, 1992.

[12] G. Hall and R. Gupta. Modeling transition. In *Proc. of ICDE'91*, 540-549, Kobe, Japan, 1991. IEEE Computer Society.

[13] *IEF Information Engineering Facility, A Guide to Information Engineering Using the IEF*, Texas Instruments, 1988.

[14] S.F. Keller. *A presentation model and a mapping language for INTERLIS*. Document RFC-1101e, Federal Directorate of Cadastral Surveying, Switzerland, September 1997.

[15] J.L. de Oliveira, F. Pires, and C. Bauzer Medeiros. An Environment for Modeling and Design of Geographic Applications. *Geoinformatica*, 1(1):29-58, 1997.

[16] C. Parent. Modélisation de Gesreau en MADS, EPFL-DI-LBD Project Report, Lausanne, Switzerland, 1995.

[17] C. Parent et al. Modeling Spatial Data in the MADS conceptual model. In *Proc. of SDH'98*, 138-150, Vancouver, Canada, 1998.

[18] N. Pissinou et al. Towards an Infrastructure for Temporal Databases, *SIGMOD Record*, 23(1):35-51, 1994.

[19] R. Snodgrass. Temporal object oriented databases: A critical comparison. In *Modern Database Systems: The Object Model, Interoperability, and Beyond*, 386-408. Addison-Wesley, 1995.

[20] R. Snodgrass (Ed.). *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.

[21] S. Spaccapietra and C. Parent. ERC+: An Object based Entity Relationship Approach. In *Conceptual modeling, databases, and CASE*, 69-86. John Wiley & Sons, 1992.

[22] A. Steiner and M.C. Norrie. Temporal object role modeling. In *Proc. of CAiSE'97*, 245-258, Barcelona, Spain, 1997.

[23] A. U. Tansel et al.. *Temporal databases. Theory, design, and implementation.* Benjamin/Cummings, 1993.

[24] B. Tauzovich. Towards Temporal Extensions of the Entity-Relationship Model. In *Proc. of ER'91*, pages 163-179, San Mateo, California, 1991

[25] Y. Wu, S. Jajodia, and X. S. Wang. *Temporal Database Bibliography Update*, http://www.isse.gmu.edu/~csis/tdb/bib97/bib97.html

[26] G.T.J. Wuu and U. Dayal: A Uniform Model for Temporal and Versioned Object-oriented Databases, in [23], 230-247.