Evaluating Temporally Extended ER Models

Heidi Gregersen Christian S. Jensen Leo Mark

Department of Computer Science, Aalborg University,
Fredrik Bajers Vej 7E, DK–9220 Aalborg Øst, Denmark,
Phone: +45 9635 8080, Fax: +45 9815 9889
email: {gregori,csj,leomark}@cs.auc.dk.

Abstract

The Entity-Relationship (ER) Model, is enjoying a remarkable popularity in industry. It has been widely recognized that while temporal aspects of data play a prominent role in database applications, these aspects are difficult to capture using the ER model. Some industrial users have responded to this deficiency by ignoring all temporal aspects in their ER diagrams and simply supplement the diagrams with phrases like "full temporal support." The research community has responded by developing about a dozen proposals for temporally extended ER models. These temporally extended ER models were accompanied by only few or no specific criteria for designing them, making it difficult to appreciate their properties and to conduct an insightful comparison of the models. This paper defines a set of design criteria that may be used for evaluating and comparing the existing temporally extended ER models.

1 Introduction

The Entity-Relationship (ER) Model [1], in its different versions, with varying syntax and semantics, is used as a database design model. The model is easy to use, and ER diagrams provide a good overview of a database design and are easy to comprehend. The focus of the model is on the structural aspects of the mini-world, as opposed to the behavioral aspects. This focus usually matches the levels of ambition for documentation adopted by many users.

In the research community as well as in industry, it has been recognized that the temporal aspects of the mini-world (we use the term "mini-world" for the part of reality that the database stores information about) are prominent, but are also difficult to capture using the ER model. Put simply, when modeling fully the temporal aspects, these tend to obscure and clutter otherwise intuitive and easy-to-comprehend diagrams. As a result, some industrial users simply choose to ignore all temporal aspects in their ER diagrams and supplement the diagrams with textual phrases to indicate that a temporal dimension to data exists, e.g., "full temporal support." The result is that the mapping of ER diagrams to relational tables must be performed by hand; and the ER diagrams do not document well the temporally extended relational database schemas used by the application programmers.

The research community's response has been to develop temporally enhanced ER models. Indeed, almost a dozen such models have been reported in the research literature, including, e.g., the Temporal EER model [8], the Temporal Entity Relationship model [25], and the Entity-Relation-Time model [27, 19], to name jus a few.

Temporal ER models are developed in an attempt to provide constructs that more naturally and elegantly support the modeling of temporal aspects, such as valid and transaction time of information. The temporal ER models are developed by changing the semantics of the ER model or by adding new constructs to the model. The approaches taken to add built-in temporal support to the ER model are quite different. One approach is to devise new notational shorthands that replace some of the patterns that occur frequently in ER diagrams when temporal aspects are being modeled. Another approach is to change the semantics of the existing ER model constructs, making them temporal. In its extreme form, this approach does not result in any new syntactical constructs—all the original constructs have simply become temporal. For a detailed description of the existing models, see Gregersen and Jensen [11].

A common characteristic of the existing temporally extended ER models is that few or no specific requirements to the models were given by their designers. In this paper, we therefore define a comprehensive set of evaluation criteria for temporally extended ER models. The usages of the criteria are several, including the following. The criteria can be used for evaluating and comparing the existing temporally extended ER models. This allows potential users of the models to determine which of the models to use. The criteria can also be used as design criteria when designing a temporally extended ER model. Although beyond the scope of this paper, the criteria are with various different adaptations applicable to extensions of the ER model other than temporal and to extensions of other modeling notations.

Three studies are somewhat related to the study reported here. In the first, Theodoulidis and Loucopoulos [26] describe and compare nine approaches to specifying and using time in semantic data modeling and requirement specification, in the context of information systems development. Their comparison of the approaches falls in three parts, classifying the approaches in terms of their time semantics, model semantics, and temporal functionalities. The primary emphasis of their study is the examination of the ontology and properties of time in the context of information systems, whereas our study adopts a much more narrow and in-depth focus on how the extensions of the ER model into temporal ER models are shaped.

In the second study, McKenzie and Snodgrass [18] survey and evaluate twelve different temporal extensions of the relational algebra. They evaluate the algebras against 26 design criteria. These criteria are mainly concerned with the properties of the data objects—temporal relations and their components—that the algebras manipulate and with the properties of the algebraic operators themselves. Our criteria are mainly concerned with the properties of the structural aspects of temporal ER models.

In the third study, Roddick and Patrick [21] survey the progress of incorporating time in various data models at the conceptual and, primarily, logical level of database modeling and in artificial intelligence. The study describes nine different properties of temporal modeling systems. These propeties overlap only slightly with our evaluation criteria.

The paper is structured as follows. Section 2 introduces concepts fundamental to temporal ER modeling. In Section 3 we present a set of evaluation criteria. Finally in Section 4, a conclusion and a discussion of future work is given.

2 Fundamental Temporal ER Concepts

An entity is a thing that exists in the real world and can be separated from other things in the real world; hence, a database representation of an entity should model its existence and unique identification. The time during which the entity exists in the mini-world, we call the existence time of the entity.

An entity is characterized by its properties, modeled by attributes. At any given point in time, an entity has a value for each of its attributes. Some attribute values vary over time and some do not, that is, at different points in time, the values of an attribute for an entity may be different. This implies that different values of an attribute are valid at different points in time.

A database represents sets of entities that are similar, that is, entities that have the same attributes, or put differently, that have the same structure. Entity types define sets of entities with the same attributes. The sets that the entity types define are called entity sets.

Entities may be interrelated via relationships. Such relationships can be seen from two very different points of view. Either we can perceive relationships among entities as attributes of the participating entities, or we can perceive relationships as things that exist in their own right. We believe that it should be possible to adopt either point of view.

A relationship type among some entity types defines a set of relations among entities of these types. Each relationship relates exactly one entity from each of the entity types that the relationship type is defined over. The set of relationships defined by a relationship type is called a relationship set.

A database should be able to capture all information about the mini-world that is meaningful to capture and is relevant for the application at hand. For example, since entities exist during some periods of time it should be possible to capture this in the database. In turn, this implies that the database designers should have the ability to indicate, in the conceptual model, whether or not they want to register these periods of time for the entities.

In the database community, several types of temporal aspects of infomation have been discussed over the years. In this paper, we will focus on four distinct types of temporal aspects that are candidates for being given built-in support in an ER model, namely *lifespan*, *valid time*, *transaction time*, and *user-defined time* [13].

The *lifespan* of an entity captures in the database the existence time of the entity. If the concept of lifespan of entities is supported, this means that the model has built-in support for capturing the times when entities exist. If relationships are regarded as having existence in their own right, the concept of lifespan is also applicable to relationships, with the same meaning as for entities.

The notion of *valid time* applies to facts: the valid time of a fact is the time when that fact is true in the mini-world. Thus, all facts in the database have an associated valid time. However, the valid times may or may not be captured explicitly in the database. In ER models, unlike in the relational model, a database is not structured as a collection of facts, but rather as a set of entities and relationships with attributes. Thus, the valid times are associated only indirectly with facts. As an example, consider an Employee entity "E1" with a Department attribute. A valid time of June 1996 associated with value "Shipping" does not say that "Shipping" is valid during June 1996, but rather that the fact "E1 is in Shipping" is valid during June 1996. Thus, when valid time is captured for an attribute such as Department, the database will record the varying Department values for the Employee entities. If it is not captured, the database will (at any time) record only one department value for each Employee entity.

The *transaction time* of a database fact is the time when the fact is current in the database and may be retrieved. Unlike valid time, transaction time may be associated with any structure stored in a database, not only with facts. Thus, all structures stored in a database have a transaction-time aspect. This aspect may or may not, at the designers discretion, be captured in the database. The transaction-time aspect has a duration: from insertion to (logical) deletion.

In addition to lifespan, valid time, and transaction time, another kind of time exists, namely the so-called *user-defined time*. User-defined times are supported when time-valued attributes are available in the data model. These are then employed for giving temporal semantics—not captured in data model, but only externally, in the application code and by the database designer—to the ER diagrams. For Employee entities, such attributes could record Birth Dates, Hiring Dates, etc.

EXAMPLE: This example shows how the temporal aspects of the mini-world may clutter up an otherwise simple ER diagram, as shown in Figure 1 and described next. This example's mini-world concerns a company divided into different Departments. Each Department has a Number and a Name and is Responsible for a number of Projects. A Department keeps track of the Profits it makes on its Projects. Because the company would like to be able to make statistics on its Profits, each Department must record the history of its Profits over Periods of time.

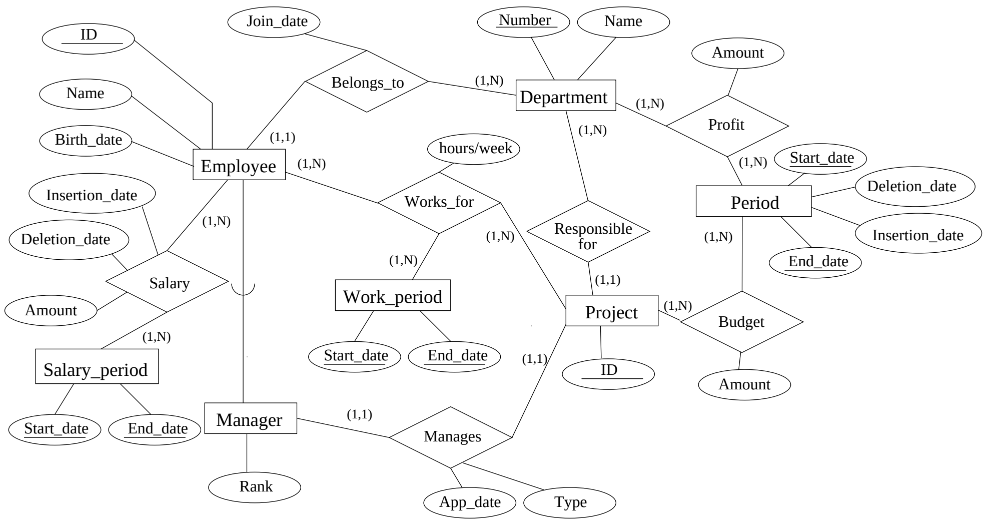

Figure 1: ER Diagram Modeling Temporal Aspects

Each Project has a Manager who Manages the Project and some Employees who Work for the Project. Each Project has an ID and a Budget. The company registers the history of the Budget of a Project. Each Project is associated with a Department which is Responsible for the Project. Employees belong to a single Department, and works for this Department for as long as the Employee is with the company. For each Employee, the company registers the ID, the Name, the Date of Birth, and the Salary. The Departments would like to keep records of the different Employees' Salary histories. For reasons of accountability, it is important to be able to trace previous records of both Profits and Salaries.

Employees work on one Project at a time, but Employees may be reassigned to other Projects, e.g., due to the fact that a Project may require Employees with special skills. Therefore, it is important to keep track of who Works for what Project at a given time and what time they are suppose to be finished working on their current project. Some of the employees are project managers. Once a Manager is assigned to a Project, the manager will manage the Project until it is completed or otherwise terminated.

If we ignored the temporal aspects in Figure 1 we could remove all the entity types modeling time periods and model the relationship types Profit, Budget, and Salary with attributes. □

3 Evaluation Criteria

Next, we present a total of 18 evaluation criteria (numbered C1 through C18), covering in combination time semantics, model semantics, temporal functionality, and user-friendliness. These criteria were derived during the studies of the existing temporally extended ER models [11]. As mentioned earlier a common characteristic for the existing temporally extended ER models is that few or no specific requirements to the models were given by their designers. We have derived the evaluation criteria with the dual purpose of evaluating existing temporally extended ER models and providing designers of temporally extended ER models with a set of design criteria.

3.1 Fundamental Criteria

The first two criteria represent fundamental decisions that have to be made when designing a temporal ER model: Should the temporal support be achieved by giving new temporal semantics to existing constructs, or should completely new temporal constructs be introduced? Should the model have a query language or should algorithms that map ER diagrams to some implementation platforms be developed? The answers to these questions fundamentally affect the nature and properties of a temporal ER model.

C1—Implicit vs. Explicit Temporal Support Two general approaches to providing temporal support in the ER model exist, namely implicit temporal support and explicit temporal support.

With *implicit* temporal support, timestamp attributes are "hidden" in the temporal semantics of the modeling constructs. For example, no timestamp attribute is necessary on a temporal relationship type to indicate that its instances record their variation over time. With implicit support, it is possible to obtain a temporal ER model without adding any new syntactical constructs to the ER model. This is so because it is possible to simply redefine all existing ER constructs to become temporal. For example, ordinary relationship types are given temporal semantics, making their instances record variation over time, rather than just single states.

This approach, where all existing ER model constructs are given a temporal semantics, has been used in several models [8, 17, 6] and has its strong points. The database designers are likely to be familiar with the existing ER constructs. So, after understanding the principle of making these constructs temporal, the designers are ready to work with, and benefit from using, the temporal ER model. However, this approach is not totally without problems. In its exstreme, this approach rules out the possibility of designing non-temporal databases, e.g., for applications where it is prohibited to record the temporal variation of data. It is also not possible to design databases where some parts are non-temporal while others are temporal. Another problem is that old diagrams are no longer correct, i.e., while their syntax is legal, their semantics have changed, and they therefore no longer describe the underlying relational database schemas.

In a temporal diagram of the mini-world from the example (Figure 1) using this kind of implicit temporal support, the entity type *Work_period* would be absent, and *Works_for* becomes a binary temporal relationship type that implicitly records valid and transaction time. Relationship type *Salary* may be replaced by a single attribute *Salary*, and relationship types *Profit* and *Budget* may be replaced by *Profit* and *Budget* attributes, respectively, eliminating also the *Period* attribute. Note also that relationship types *Belongs_to* and *Manages* become temporal and thus start recording valid and transaction time.

It is also possible to retain the existing ER constructs with their usual semantics while achieving implicit temporal support. This is accomplished by adding new temporal constructs to the model that provide the implicit support [15, 20, 27, 19, 25, 16]. The extent of the changes made to the ER model may range from minor changes to a total redefinition of the model.

The models that retain the existing constructs with their old semantics and introduce new temporal constructs also have problems. If their extensions are comprehensive, they are likely to be more difficult for the database designers to learn and understand. The larger initial investment in training that this induces may prevent a model from being accepted in industry. On the other hand, it is possible with the models following this approach to avoid the problem of old legacy diagrams, with their new semantics, not describing the underlying relational database, since the semantics of the existing ER constructs are retained. This is important for industrial users with many legacy diagrams. It is also possible to design non-temporal databases as well as databases where some parts are non-temporal and others are temporal.

In the temporal diagram of the mini-world depicted in Figure 1 using new temporal constructs with implicit temporal support, the entity type *Works_for* would be replaced by a new binary relationship type that records valid time. Relationship types *Salary*, *Profit*, and *Budget* may be replaced by new attribute types that record valid and transaction time. Note that the relationship types *Belongs_to* and *Manages* remain non-temporal. As before, a much simplified ER diagram results, this time modeling the mini-world more accurately.

With *explicit* temporal support, the semantics of the existing ER constructs are retained [9] and timestamp attributes are explicit. Any new modeling constructs are notational shorthands introduced to make the modeling of temporal aspects more convenient.

It is an advantage of this approach that old diagrams are still legal, with the same meaning as in the non-temporal model. The existing translations from the ER model to the relational model are still available. It is a disadvantage that the time-valued attributes that are still present in the shorthands have no built-in semantics, that is, the time-valued attributes in the shorthands are no different from other time-valued (and non-time valued). Finally, all the explicit timestamp attributes in shorthands still clutter up diagrams.

When using shorthands, a temporal diagram of our sample mini-world may introduce shorthands for recording the *Salary* of *Employees*, as well as their time-varying assignment to *Projects*.

The ideal temporal ER model is easy to understand in terms of the ER model; does not invalidate legacy diagrams and database applications; and does not restrict databases to be temporal, but rather permits the designer to mix temporal and non-temporal parts.

C2—Mapping Algorithm vs. Query Language A mapping algorithm translates a temporal ER diagram into a corresponding database schema in another data model. The most obvious possible target data models are the relational model and the conventional ER model.

The algorithm may map temporal ER diagrams directly to relational database schemas [9, 15, 20, 27, 19, 17], or a two-phase approach may be adopted where temporal ER diagrams are first mapped to conventional ER diagrams and then mapped to relational database schemas, reusing mappings from the conventional ER model to the relational model [9, 25]. For minor extensions of the ER model, the reuse in the two-phase approach may be attractive. However, the two-phase translation yields less control over what relational schemas result from the mapping.

With this approach, which is the one assumed in most temporal ER models, the ER model is a design model that is used solely for database design and has no directly associated database instance. In industry, the ER model is generally used as a design model, with variations of the relational model (as supported by the various specific relational products) being the most popular target model.

As an alternative to mapping ER diagrams to the schema of a separate implementation platform, another approach is to assume a system that implements the ER model directly [8, 6, 27, 19, 17]. With this approach, a mapping to an implementation platform is not required. Instead, a query language should be available for querying ER databases. If this approach is taken, one faces the challenges of devising a query language for a temporal ER model.

We focus on the use of the ER model as a design model and do not consider temporal ER query languages.

3.2 Model Properties

The criteria presented in this section represent design criteria that characterize the technical aspects of a temporal ER model. They are thus important when designing a temporal ER model and when evaluating existing models.

C3—Upward Compatibility A temporal ER model is upward compatible with respect to the conventional ER model if any legal conventional ER diagram is also a legal ER diagram in the temporal model and if the semantics of the diagrams in the two models is the same. Upward compatibility is very important because it enables legacy ER diagrams to be used immediately in the new temporal model. Investments in legacy systems are protected, and a basis for a smooth transition to a temporally enhanced ER model is provided [23].

Temporally extended models where no syntactical constructs from the basic models have been given new semantics are upward compatible with respect to their basic model [9, 20, 27, 19, 16]. On the other hand, if the extended model has changed the semantics of the modeling constructs of the original model, then the new model cannot be upward compatible with the original model.

When evaluating a model against this criterion, the evaluation must consider whether the model is upward compatible with respect to the ER model that it extends, if specified; otherwise, we recommend using Chen's ER model [1] for models without superclass/subclass relationships and the EER model [7] for models with superclass/subclass relationships.

C4—User-specifiable Temporal Support A temporal ER model offers *user-specifiable* temporal support if it is up to the database designer to decide which temporal aspects of data to capture. For example, a temporal ER model may provide built-in support for both valid and transaction time, but a specific application may only require support for transaction time. It should then be possible to omit support for valid time.

The choice of a general approach to temporal support, as discussed in Criterion C1, impacts a designer's ability use the model for a flexible specification of what temporal aspects to capture. Thus, some existing models provide no flexibility, while other models do provide the desired flexibility.

C5—Mandatory vs. Optional Use of Temporal Constructs Use of the new temporal constructs introduced in a temporal ER model may be *mandatory* or *optional.* If the use of a temporal construct is mandatory, use of the original non-temporal constructs in diagrams is not possible. Optional use of the temporal constructs provides the designer with the possibility of mixing temporal and non-temporal constructs in the same diagram.

If a temporal ER model has become temporal by changing the semantics of the constructs of the original model, the model requires mandatory use of the temporal constructs [8, 6, 25, 17].

If a model has become temporal by adding new syntactical constructs with implicit temporal support or uses explicit temporal support, the model offers optional use of the temporal constructs [9, 15, 20, 27, 19, 16].

C6—Time Dimensions with built-in Support Valid and transaction time are general—rather than application specific—aspects of all database facts [22]. As such, they are prime candidates for being built into a temporal ER model. Indeed, for a model to be considered temporal, at least one of these time dimensions is generally required to be supported [13]. Being orthogonal and independent aspects of database facts, it is possible to support the two time dimensions independently. Support for these time dimensions may take different forms and may be more or less elaborate.

Since only facts have valid time and attributes (as well as relationships, see next) are used to capture facts in ER modeling, it should be possible to indicate that the application under consideration must register valid time for attributes. If a relationship is viewed as an attribute of the participating entities, support for the registration of valid time of relationships should also be offered. Support for transaction time should be offered for all the modeling constructs, since any construct that may be recorded in a database has a transaction time.

While the temporal ER models generally support valid time (e.g., [9, 15, 20, 8, 6, 27, 19, 25, 17, 16]), support for transaction time is provided only rarely (i.e., by one model [17]).

User-defined time attributes, i.e., time-valued attributes with no special support, are supported by all the temporal models, as well as by the ER model. The temporal information in the example, described in Figure 1, is captured exclusively using user-defined time. The *Salary* information and *Project* assignments of *Employees* and the *Profit* and *Budget* information of *Departments* and *Budgets*, respectively, may be modeled more appropriately using built-in support for valid and transaction time.

C7—Data Types Supported for Valid Time Different data types for implicit or explicit timestamps may be used for capturing the valid time of an attribute value or a relationship. Possible time data types include instants, time intervals, and valid-time elements. Valid time intervals are finite unions of time intervals [10]. Each of these data types are supported by several temporal ER models.

For example, one option is to associate valid-time intervals with attribute values of entities, and another option is to timestamp attribute values with valid-time elements. An attribute value may also be defined as a function from a time domain to a value domain. This is similar to element timestamping in that an attribute may associate a value with some subset of the time domain. A data type of sets of time intervals somewhat resembles this latter data type. With a discrete and bounded time domain, the two types are equivalent, but this does not hold in general.

A temporal ER model may provide the database designer with a choice of data types, thereby increasing the utility of the model. Instants, intervals, and elements may all be used for encoding durations. When instants are used for this purpose, they have associated interpolation functions. The instant data type may also encode the occurrence of instantaneous events.

The importance of the availability of time data types is dependent on whether the model under consideration is used solely as a design model or is also used as an implementation model, complete with database instances and a query language.

In Figure 1 the entity type *Salary_period* models that the temporal attribute *Salary* is to be timestamped with valid-time elements.

C8—Data Types Supported for Transaction Time Because valid and transaction time capture temporal aspects with distinct meanings, the timestamp types used for the two may also differ. The possible data types are those used for encoding durations in valid time, namely instants, intervals, and temporal elements.

In Figure 1 the attributes *Insertions_date* and *Deletion_date* model that the temporal attribute *Salary* is to be timestamped with transaction-time elements.

C9—Lifespans of Entities All entities have existence and thus have an associated existence time. The lifespan of an entity captures the time during which the entity exists in the mini-world. If the concept of lifespan of entities is supported by a temporal ER model, this means that the model has built-in support for capturing the times when entities exist, i.e., it must be possible for the database designer to indicate that the existence time of any entity should be captured in the database.

Because entities may exist beyond the times when their attributes have (non-null) values, it is not possible to infer lifespans from the assignments of timestamps to the values of their attributes.

Some ER models provide built-in support for capturing the lifespans of all entities [15, 8, 27, 17]; other models support only lifespans for a subset of their entities [6, 16]; and some models do not support lifespans at all [9, 20, 25].

C10—Lifespans of Relationships In Section 2, we argued that a relationship may be seen as having independent existence. In that case, relationships also have associated existence times, and it should be possible to capture lifespans for relationships, in the same way as for entities.

When a model provides a built-in notion of relationship lifespans, it may also enforce certain temporal constraints that involve these lifespans. The obvious constraint on a relationship lifespan is that the lifespan of an relationship instance should be a subset of the union of the lifespans of the participating entities.

As for entities, some models support lifespans for all relationship types [15, 8, 27, 17], while other models only support them for some relationship types[6, 16]; yet other models provide no built-in support [9, 20, 25].

C11—Support for Multiple Granularities It may be that the temporal variability of different objects in the mini-world are captured using times of different granularities [28, 4]. It should then also be possible to capture the variability of the different objects in the database using these different granularities.

To exemplify, the granularity of a minute may be used when recording the actual *Working Hours* of *Employees*, while the granularity of a day may be used when recording the *Department* assignments of *Employees*.

One model [15] supports multiple granularities. Another model supports specifying different frequencies for the recording of the attribute values [20], yielding also some support for multiple granularities. Notice that this criterion relates to lifespans and valid time.

C12—Support for Temporal (Im-) Precision The temporal variability of different objects in the mini-world may be known with different precisions [15, 2, 5, 3], and although some imprecision may be captured using multiple granularities, granularities do not provide a general solution.

For example, the variability of an attribute may be recorded using timestamps with the granularity of a second, but the varying values may only be known to the precision of ± 5 seconds of the recorded time. This phenomenon may be prevalent and important to capture in scientific and monitoring applications that store measurements made by instruments.

C13—Temporal Constraints A temporal ER model may include built-in temporal constraints, as well as facilities for user-definable temporal constraints. If built-in temporal constraints are not present, the possibility of having illegal data is present. For example, a (binary) relationship between two entities can usually not exist if the entities do not exist. The presence of an appropriate set of (built-in) constraints on the built-in temporal constructs is thus of essence. Next, it should be possible for the database designer to specify additional temporal constraints. For example, we have seen that one of the existing models [25] supports two types of temporal constraints on relationship types, namely snapshot and lifetime cardinality constraints.

Several of the existing temporal ER models [15, 8, 6, 27, 17]provide such support to varying degrees.

C14—Temporal Interpolation Functions Temporal interpolation functions derive information about times for which no data is explicitly stored in the database (see, e.g., [15] and [14]). For example, it is possible to record times when new *Salaries* of *Employees* take effect and then define an interpolation function (using so-called step-wise constant interpolation) that gives the *Salaries* of *Employees* at any time during their employment. In the scientific domain, interpolation is particularly important, e.g., when variables are sampled at different rates.

Three temporal ER models [15, 20, 27] provide support for interpolation.

C15—Snapshot Reducibility of Attribute Types Temporal ER models that add temporal support implicitly may include temporal counterparts of the ordinary attribute types, i.e., provide temporal single valued, temporal multi-valued, temporal composite, and temporal derived attribute types. These temporal attribute types may be snapshot reducible [24] with respect to their corresponding snapshot attribute types. This occurs if snapshots of the databases described using the temporal ER diagram constructs are the same as databases described by the corresponding snapshot ER diagram where all temporal constructs are replaced by their snapshot counterparts. For example, a temporal single-valued attribute is snapshot reducible to a snapshot single-valued attribute if the temporal single-valued attribute is single valued at each point in time.

Generalizing snapshot constructs this way yields a natural temporal model that is easily understood in terms of the conventional ER model. Indeed, many existing temporal ER models satisfy this criterion (e.g., [9, 15, 20, 8, 6, 17]).

C16—Snapshot Reducibility of Relationship Constraints Snapshot reducibility also applies to the various constraints that may be defined on relationship types, including specialized relationship types such as ISA (superclass/subclass) and PART-OF (composite/component) relationships. For example, the temporal cardinality constraint 1–N on a binary temporal relationship type is snapshot reducible to the snapshot cardinality constraint 1–N on a binary snapshot relationship type if the 1–N snapshot constraint applies to any state, valid at any single point in time, of any possible instances of the temporal relationship type.

Two existing temporal ER model provide snapshot reducible relationship constraints ([25, 16]).

3.3 Other Criteria

The criteria in this section differ from the criteria presented earlier in that they do not concern the technical properties of temporale ER models per se. However, they are still important for characterizing the usability of existing models.

C17—Graphical Notation Provided While it is usually assumed that a graphical notation is available for describing ER diagrams, this needs not be so. It is also possible to provide only a textual notation for describing ER diagrams. It is generally believed that ER models with a graphical notation have an advantage over ER models with a programming language-like notation. Graphical notations tend to be easier to learn, and we believe that the simplicity of the graphical ER notation is one of the main reasons for its success. Indeed, with the exception of one temporal ER model [15], which provides a Pascal-like syntax, all temporal ER models provide a graphical notation [9, 20, 8, 6, 27, 19, 25, 17, 16].

C18—Graphical Editor Available If the notation of a model is graphical, then the presence of an editor supporting the model is very important if the model is to be used widely. Potential users should have the opportunity to try and use at least some prototype of an editor supporting the model. Four of the existing models have an associated graphical editor [27, 19, 25, 8, 17]. Two of these models [8, 17]. do not add new syntactical constructs and thus can reuse existing editors for the ER models (although they cannot reuse the existing mappings to the relational model).

4 Conclusions and Research Directions

In this paper, we have presented 18 evaluation criteria for evaluating and comparing temporally extended ER models. The criteria cover, in combination, aspects of time semantics, model semantics, temporal functionality, and user-friendliness.

Elsewhere, we have used the criteria presented here to evaluate all existing temporally extended ER models [11]. While no single model satisfies all the properties, the models collectively cover the design space. The models illustrate different ways in which the temporal aspects of data can be conveniently captured at the conceptual level, and it is our contention that all the temporal ER models, to varying degrees, have succeeded in more naturally capturing the temporal aspects of data than does the ER model.

We believe that the criteria may be successfully employed to uncover intrinsic properties of temporal ER extensions, rather than more extrinsic, or extraneous, properties such as, e.g., the specific choice of graphical notation for the modeling constructs. It is thus our contention that these criteria constitute a new and solid basis for evaluating temporal ER extensions.

The criteria also serve as a checklist of design aspects that should be considered when designing a temporal extension. We recommend that designers of future temporally extended ER models consciously consider the ER extension for each criterion.

A number of topics are the subjects of ongoing research or are candidates for future research. First, the set of criteria is not neccesarily complete. It may thus be feasible to include additional criteria for evaluating and comparing temporal ER models. Second, there is a need for a taxonomy of evaluation criteria that charts the design space of temporal extensions and thus may be used for indicating areas with missing criteria. With such a taxonomy, it becomes possible to ensure that we evaluate all important properties of temporally extended ER models.

Third, most of the criteria may be applied to extensions of the ER model other than temporal ones. In this paper we have described the criteria in the specific context of temporal extensions for concreteness. It is an interesting next step to explore the application of the criteria to other types of ER extensions, such as spatial [12], spatio-temporal, or multi-media ER models. Fourth, the application of the criteria to extensions of modeling notations other than the ER model is a promising direction deserving attention.

5 Acknowledgements

The research was supported in part by grant 9502695 from the Danish Technical Research Council, grant 9400911 from the Danish Natural Science Research Council, and the CHOROCHRONOS project, funded by the European Commission DG XII Science, Research and Development, as a Networks Activity of the Training and Mobility of Researchers Programme, contract no. FMRX-CT96-0056.

References

[1] P. P-S. Chen. The Entity-Relationship Model — Toward a Unified View of Data. *ACM Transaction on Database Systems*, 1(1):9–36, March 1976.

[2] C. E. Dyreson and R. T. Snodgrass. Valid-time Indeterminacy. In *Proceedings of the 9th International Conference on Data Engineering*, pp. 335–343, Vienna, Austria, 1993.

[3] C. E. Dyreson and R. T. Snodgrass. Temporal Granularity and Indeterminacy: Two Sides of the Same Coin. TR 94-06, University of Arizona, Dept. of Computer Science, Feb. 1994.

[4] C. E. Dyreson and R. T. Snodgrass. Temporal Granularity. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, Chapter 19, pp. 347–383. Kluwer, 1995.

[5] C. E. Dyreson and R. T. Snodgrass. Temporal Indeterminacy. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, Chapter 18, pp. 327–346. Kluwer, 1995.

[6] R. Elmasri, I. El-Assal, and V. Kouramajian. Semantics of Temporal Data in an Extended ER Model. In *Proceedings of the 9th ER Conference*, pp. 239–254, Lausanne, Switzerland, Oct. 1990.

[7] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, Inc., 2nd edition, 1994.

[8] R. Elmasri and G. T. J. Wuu. A Temporal Model and Query Language for ER Databases. In *Proceedings of the Sixth International Conference on Data Engineering*, pp. 76–83, 1990.

[9] S. Ferg. Modeling the Time Dimension in an Entity-Relationship Diagram. In *Proceedings of the 4th ER Conference*, pp. 280–286, Silver Spring, MD, 1985. Computer Society Press.

[10] S. K. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, 13(4):418–448, Dec. 1988.

[11] H. Gregersen and C. S. Jensen. Temporal Entity-Relationship Models—A Survey. R-96-2039, Aalborg University, Department of Mathematics and Computer Science, Sep. 1996.

[12] T. Hadzilacos and N. Tryfona. An Extended Entity-Relationship Model for Geographic Applications. Technical report, Computer Technology Institute, Greece, 1996.

[13] C. S. Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. Hayes, and S. Jajodia [eds]. A Glossary of Temporal Database Concepts. *ACM SIGMOD Record*, 23(1):52–64, March 1994.

[14] C. S. Jensen and R. T. Snodgrass. Semantics of Time-Varying Information. *Information Systems*, 21(4):311–352, March 1996.

[15] M. R. Klopprogge. TERM: An Approach to Include the Time Dimension in the Entity-Relationship Model. In *Proceedings of the 2nd ER Conference*, pp. 477–512, Washington, DC, Oct. 1981.

[16] P. Kraft. Temporale kvaliteter i ER modeller. Hvordan? Working paper 93, The Aarhus School of Business, Department of Information Science, Jan. 1996.

[17] V. S. Lai, J-P. Kuilboer, and J. L. Guynes. Temporal Databases: Model Design and Commercialization Prospects. *DATA BASE*, 25(3):6–18, 1994.

[18] E. McKenzie and R. Snodgrass. An Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. *ACM Computing Surveys*, 23(4):501–543, Dec. 1991.

[19] P. McBrien, A. H. Seltveit, and B. Wangler. An Entity-Relationship Model Extended to describe Historical information. In *International Conference on Information Systems and Management of Data*, pp. 244–260, Bangalore, India, July 1992.

[20] A. Narasimhalu. A Data Model for Object-Oriented Databases with Temporal Attributes and Relationships. Technical report, National University of Singapore, 1988.

[21] J. F. Roddick and J. D. Patrick. Temporal Semantics in Information Systems – A Survey. *Information Systems*, 17(3):249–267, Oct. 1992.

[22] R. Snodgrass and I. Ahn. A Taxonomy of Time in Databases. In S. Navathe, editor, *Proceedings of the ACM SIGMOD '85 International Conference on Management of Data*, pp. 236–246, Austin, TX, May 1985.

[23] R. T. Snodgrass, Böhlen M., C. S. Jensen, and A. Steiner. Change Proposal to SQL/Temporal: Adding Valid Time—Part A. *International Organization for Standardization*, page 40, Dec. 1995. ANSI Expert's Contribution.

[24] R. T. Snodgrass. The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[25] B. Tauzovich. Toward Temporal Extensions to the Entity-Relationship Model. In *Proceedings of the 10'th ER Conference*, pp. 163–179, San Mateo, California, Oct. 1991. E/R Institute.

[26] C. I. Theodoulidis and P. Loucopoulos. The Time Dimension in Conceptual Modelling. *Information Systems*, 16(3):273–300, 1991.

[27] C. I. Theodoulidis, P. Loucopoulos, and B. Wangler. A Conceptual Modelling Formalism for Temporal Database Applications. *Information Systems*, 16(4):401–416, 1991.

[28] G. Wiederhold, S. Jajodia, and W. Litwin. Dealing with Granularity of Time in Temporal Databases. In R. Anderson et al. [eds], *Proceedings of CAiSE '91*. LNCS, Vol. 498, Springer Verlag, 1991.