Requirements Specification in TEMPORA

C. Theodoulidis[1], B. Wangler[2] and P. Loucopoulos[1]

[1] Department of Computation,
UMIST,
P.O. Box 88,
Manchester M60 1QD,
UK

[2] SISU,
Swedish Institute for Systems Development,
Box 1250,
S-164 28 Kista,
Sweden

Abstract

The use of formal language is a way to introduce rigour in the specification of the requirements for information systems. This stage is traditionally considered as the most informal one of the life-cycle stages. Thus, the choice of a model best suited for this purpose is still an open issue.

In this paper we propose a data model called Enity-Relationship-Time (ERT), which is able to capture the structural components of such a specification. It is part of the TEMPORA conceptual model and it is an extension of the binary relationship model including a number of additional features such as the possibility to explicitly refer to past or future states of the system, to model complex objects, etc.

1. Introduction

Recent years have witnessed a growing realisation that the development of large information systems is becoming increasingly more difficult as user requirements become broader and more sophisticated. Consequently, requirements analysis is becoming even more sensitive because the late discovery of misunderstandings of the users' needs is the source of the most expensive modifications to such systems [Greenspan, 1984; Balzer et al, 1983; van Assche et al, 1988]. By enforcing greater rigour during requirements analysis i.e., by introducing formal recording, would definitely help to avoid such misunderstandings, by removing ambiguities, redundancies and untimely choices [Mylopoulos, 1986].

Many recent approaches have suggested the use of a formal language for requirements analysis (see [CRIS-1], [CRIS-2], [CRIS-3], [Roman, 1985]). However, the features needed for a language suited for the formal expression of requirements are largely debated. Several existing languages inherit their basic concepts from other fields like data base modelling, knowledge representation or programming languages without paying enough attention to their appropriateness for expressing the customers' needs. This is the result of focusing mainly on languages and their foundations instead on understanding the process.

Much debate is currently under way as to the most appropriate set of requirements for conceptual modelling languages [Roman, 1985; Balzer et al, 1983]. An emerging consesus is the need for modelling

- Temporal Aspects. Such an extension would provide the often needed ability to reason about elements of the Universe of Discourse (UoD) which involve time. For example, historical data is required in hospital information systems in monitoring patient progress and relate current situation to previous ones. In other systems, planning is of equal importance. For example, in weather forecast systems one needs to be able to reason for the future based on current and previous data.

- Complex Objects. Such an extension would provide the ability needed for many applications for the abstraction of information that is going to be used as a single unit. For example, in CAD/CAM or CASE applications one needs to be able to deal with objects that consist of a number of components and to reason for them and at the same time to be able to deal with their components.

The process of requirements analysis is mainly an activity of modelling an application domain. This implies that certain things are needed for the natural description of various phenomena perceived in a Universe of Discourse (UoD)[Dubois, 1986; Dubois, 1987]. For example, we need:

- to classify phenomena perceived individually and associations among them,
- to classify phenomena perceived as complex structures,
- to express both static and dynamic constraints about the phenomena,
- to explicitly refer to a global time.

In section 2 of this paper we describe briefly the TEMPORA paradigm and its architecture together with the basic components of the specification environment. In section 3 we describe in detail the structural formalism of the TEMPORA conceptual model which possess the features described above. In particular, we discuss its basic concepts and externals together with the semantics of time and complex objects used.

2. The TEMPORA paradigm

2.1 Introduction

The aim of the TEMPORA project is to improve the software development process through the exploitation of an approach which explicitly recognises the role of business policy within an information system and visibly maintains this policy throughout the software development process, from requirements specifications through to an executable implementation [van Assche et al, 1988; Loucopoulos, 1989]. This implies that the TEMPORA paradigm views the development of an information system as the task of developing or augmenting a knowledge base of business rules [TEMPORA, 1988]. In particular the need to explicitly represent business rules, to be kept distinct from the procedures and elementary data operations which implement them, has been recognised in a previous ESPRIT project [RUBRIC, 1989a; RUBRIC, 1989b] and this philosophy is continued in TEMPORA.

The TEMPORA project builds on the rule-oriented system development paradigm and extends this work in two directions. The first direction is concerned with the utilization of a commercial DBMS as the underlying data management mechanism. The second direction is concerned with enhancing the paradigm with the explicit modelling of temporal aspects at both specification and application levels.

2.2 Overview of the TEMPORA Conceptual Component

The TEMPORA paradigm is that development of an information system should be viewed as the task of developing or augmenting the policy knowledge base of an organisation, which is used throughout the software development process, from requirements specification through to the run-time environment of application programs. Within TEMPORA, this knowledge base is concerned with the definition of the principal facts and operations within the organisation together with the rules which guide these operations and ensure the integrity of these facts.

It has been seen from the outset of TEMPORA that realistic modelling of the application domain demands temporal modelling. This is because nowadays "time-less" models are considered to be with respect to information systems conceptual modelling requirements, like "programming in machine code instead of using high-level programming languages" [Falkenberg, 1988]. As a consequence, the TEMPORA model must be capable of dealing with historical information issues as well as being capable of modelling temporal business rules. An abstract view of the TEMPORA conceptual components and their interrelationships is shown in the diagram of figure 1. In this figure there are two levels namely the specification level and application level. At the specification level three models are defined, the ERT model, the Process model and the Rule model while at the application level we have a database schema and a language which describes and manipulates the information contained in it.

The structural component is expressed as an extended binary-relationship model called Entity-Relationship-Time (ERT) model. It is extended because it accommodates directly the representation of time as a distinguished entity and also, it caters for the representation of complex objects.

The process component deals with the definition of operations. A process is the smallest independent unit of business activity of meaning, initiated by a specific trigger and which, when complete, leaves the business in a consistent state. By analysing processes in terms of the ERT model we end up with a set of primitive actions suffered by an entity such as salary increase.

Control of the behaviour of a system is modelled in terms of rules. Two general classes of rules are recognised: static rules which are concerned with the integrity of the database and dynamic rules which are concerned with the control of transactions. More specifically, static rules are expressions that must hold in every valid state of the database. It can be said to hold (or not hold) simply on the basis of the extension of the database with respect to a single state. In other words, the static rules represent either integrity constraints on the ERT model or derivations on it. Dynamic rules are expressions that define valid state transitions in the database. It can be said to hold (or not hold) only by examining at least two states of the database. In effect, the dynamic rules specify the interaction of state components and/or event occurrences and they represent either dynamic integrity rules or control of operations.

3. The Entity-Relationship-Time (ERT) Model

3.1 The basic concepts

Besides the temporal dimension and the provision of complex objects, ERT differs from the original Entity Relationship model [Chen, 1976] in that it regards any association between objects in the unified form of a relationship thus avoiding the unnecessary distinction between attributeships and relationships [Kent, 1979; Nijssen, 1988].

The ERT model represents explicitly *entity types* and *value types*. For each relationship the ERT model recognises *sentence predicates* which are used to make statements (e.g. "a PRODUCT is sold at a PRICE") and *referent functions* which are used as a selection mechanism for entities or values (e.g. "a PRODUCT ... sold at a PRICE .."). In essence, these are two linguistic ways of expressing the same diagrammatic structure.

Time is introduced in the ERT model as a distinguished entity class. More specifically, we timestamp each time-varying entity class and each time-varying relationship class with a time period class. That is, we assign a time period for every time-varying piece of information that exists in a schema. For example, for each entity class we associate a time period which represents the period of time during which an entity is modelled (existence period of an entity). The same argument applies also to relationships i.e., with each time-varying relationship we associate a time period which represents the period during which the relationship is valid (validity period of a relationship).

The structural components of the TEMPORA model are based upon an extended binary entity relationship modelling formalism using the following concepts.

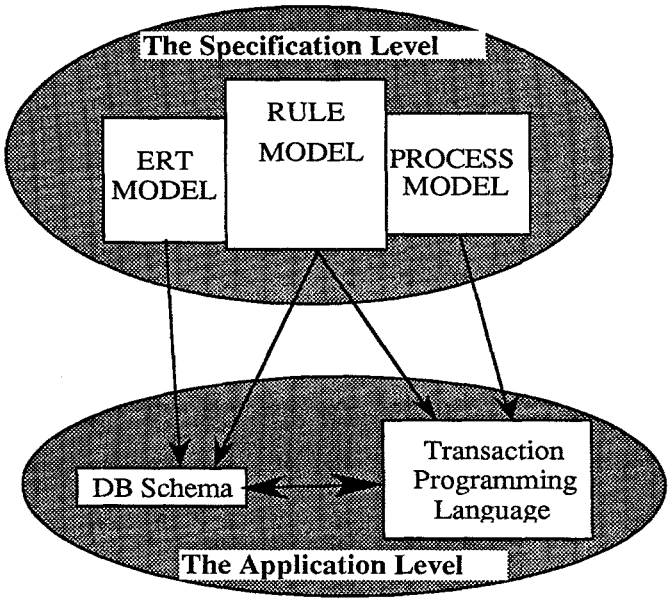

Figure 1: Overview of the TEMPORA Conceptual Component

Entity is anything, concrete or abstract, uniquely identifiable and being of interest during a certain time period.

Entity Class is the collection of all the entities to which a specific definition and common properties apply at a specific time period.

Relationship is any permanent or temporary association between two entities or between an entity and a value.

Relationship Class is the collection of all the relationships to which a specific definition applies at a specific time period.

Value is a lexical object perceived individually, which is only of interest when it is associated with an entity. That is, values cannot exist in their own.

Value Class is the proposition establishing a domain of values.

Time Period is a pair of time points expressed at the same abstraction level.

Time Period Class is a collection of time periods.

Complex Object is a complex value or a complex entity. A complex entity is an abstraction (aggregation or grouping) of entities, relationships and values (complex or simple). A complex value is an abstraction (aggregation or grouping) of values.

Complex Object Class is a collection of complex objects. That is, it can be a complex entity or a complex value class.

In addition, the following axioms apply to the concept of a relationship class.

1. An entity can only participate in a relationship if this entity is already in the population of the entity class specified in the relationship. Furthermore, the validity period of the relationship should be subperiod of the intersection of the existence periods of the two involved entities.

2. Each entity in a subclass population has also a reference (e.g., foreign key) in the population of its superclasses. In addition, the existence period of the specialised entity should be a subperiod of the existence period of the generalised entity.

3. If an entity belongs to a population of an entity class, it cannot also belong to the population of a value class at any time and vice-versa. Furthermore, any two entity classes which are not themselves subclasses of a third entity class and all have no common subclasses, must be disjoint at any time point. Note that this definition does not prevent entities from moving between entity classes during their lifetime.

In addition, an entity or relationship can be derived. This implies is that its value is not stored by default. Also, for each such derivable component, there is a corresponding derivation rule which gives the members of this class or the values of this relationship at any time.

We accommodate explicitly generalization/specialization hierarchies. This is done through the ISA relationships which have the usual set-theoretic semantics. More specifically, we assume that two subclasses of the same entity class and under the same specialization criterion are always disjoint.

3.2 External Representation of ERT

Figure 2 presents the current notation for the ERT externals. Note also that in the graphical

model we cater for representation of some of the most common rules such as partial/total ISA relationships and cardinality constraints for relationships.

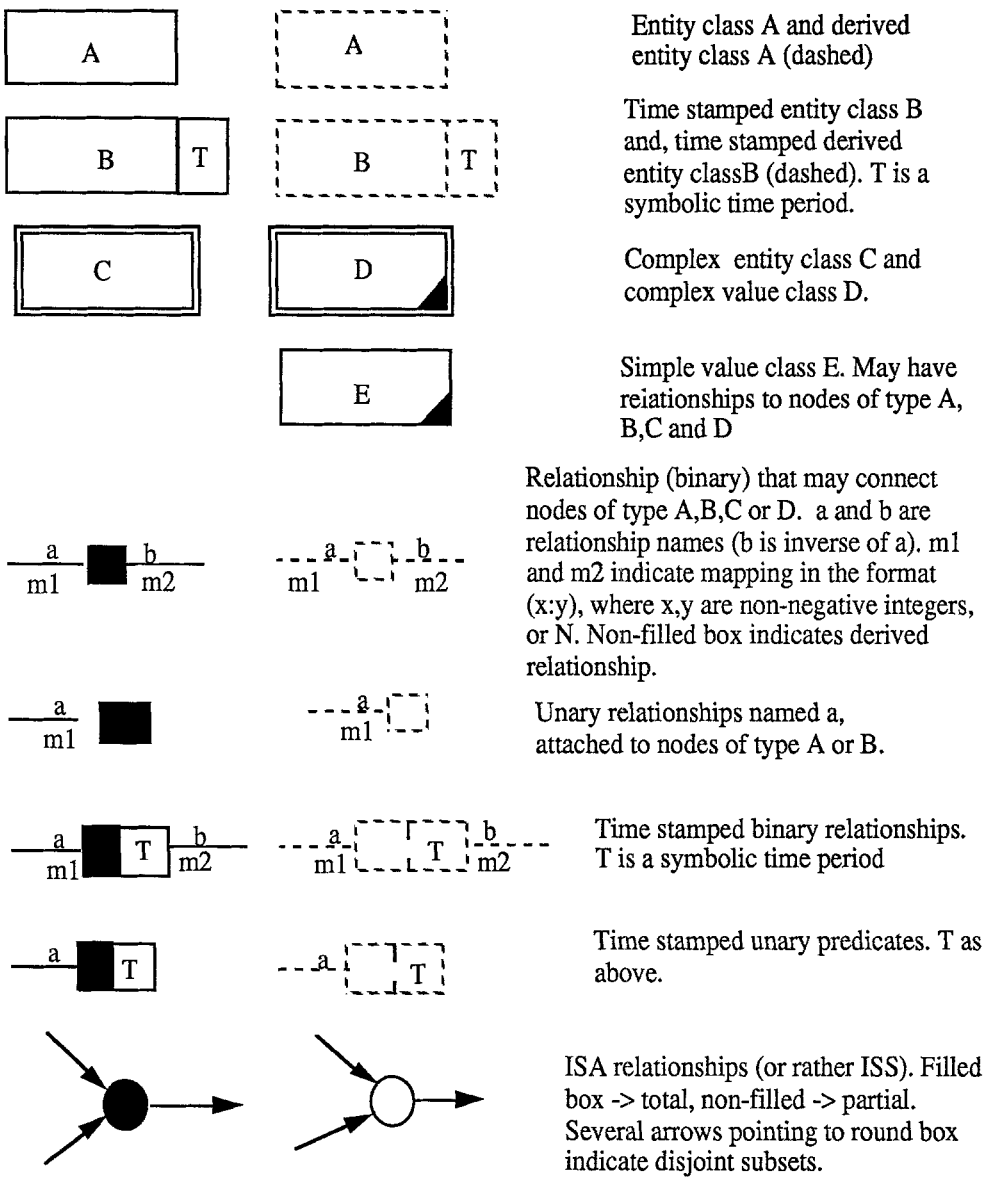

Figure 2. Graphical notation for the ERT externals

Cardinality constraints may be given to all relationships (including the IS_PART_OF

relationship) and also for their respective inverse relationships. These are expressed in the format (x-y), where x indicates the minimum cardinality and y the maximum cardinality for the set of range-objects that are related to an arbitrary domain-object via the particular relationship and correspondingly for the inverse. Hence, $0 \leq x \leq y$.

Note here that we do not include a separate notation formalism for the IS_PART_OF relationships between a complex object and its components. However, we interpret their corresponding cardinality constraints in a slightly different way. This will be explained in more detail when we discuss the semantics of complex objects in the next section. The decision not to employ a specific notation was based on the general objectives of a modelling formalism which are, besides others, that it must be simple, easy to understand and should only express the essential facts about the Universe of Discourse in question.

The notation for a complex object in TEMPORA is exemplified in figure 3. In this figure, there is an example ERT diagram with a complex entity class CAR and a complex value class ADDRESS. Furthermore, the complex entity class CAR and the complex value class ADDRESS of figure 3 may be exploited to yield their detailed structure of figure 4. This mechanism might be preferably built into an ERT editor.

3.3 Semantics of complex objects

Complex objects can be viewed from at least two different perspectives:

1. The representational perspective which focuses on how entities in the real world should be represented in the conceptual schema. This entails that objects may consist of several other objects arranged in some structure. Events in the real world are then mapped to operations on the corresponding objects. In contrast, if complex objects are not allowed, like e.g., in the relational model, then information about the object is distributed and operations on the object are transformed to a series of associated operations.

2. The methodological perspective which means that the complex object concept is regarded as a means for stepwise refinement of the schema and for hiding away details of the description. This in turn means that complex objects are merely treated as abbreviations that may expanded when needed.

The basic motivation for the inclusion of the complex entity/value class in the externals formalism, is to abstract away detail, which in a particular situation is not of interest, from the model. In addition, the semantics that we attach to complex objects are equivalent to *structural object orientation* as defined in [Dittrich, 1986].

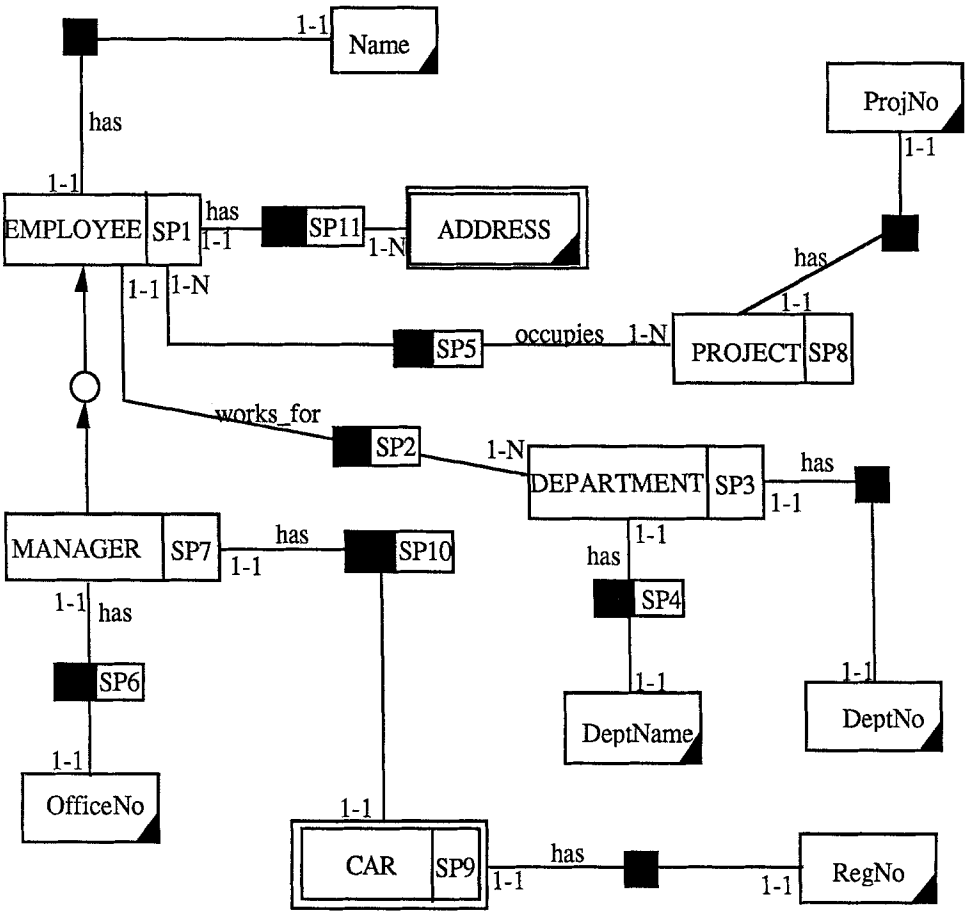

Figure 3. An example ERT schema

Many papers dealing with complex objects view a complex object as consisting of a conglomerate of objects and relationships. This means that they do not, distinguish between aggregation and grouping, but rather consider a general composition mechanism which also involves relationships/attributeships. This is the approach adopted in ERT. Graphically,

composition is shown by surrounding the components with a rectangle representing the composite object type (see figure 4).

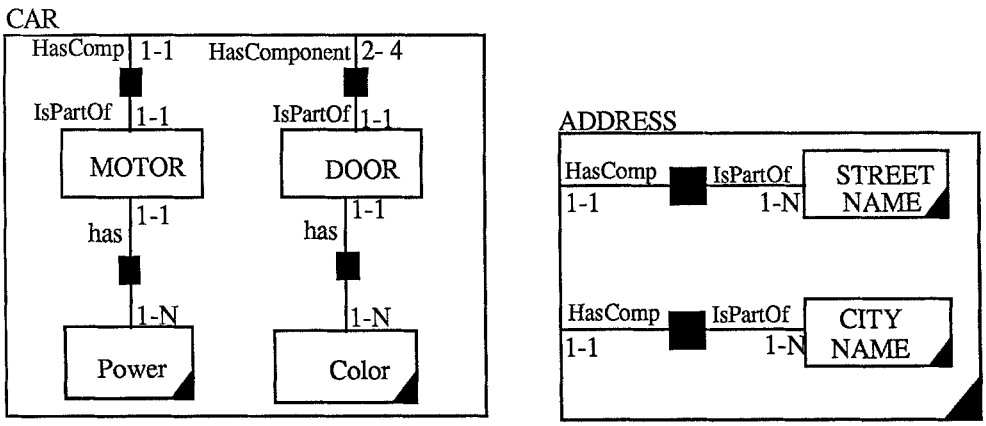

Figure 4. The complex objects of the example in more detail

The components of a complex object comprise one or more hierarchically arranged substructures. Each directly subordinate component entity must be is_part_of-related to the complex object border so that the relationship between the composite object and its components will be completely defined. Whether the HasComponent relationship is one of aggregation or grouping, can be shown by means of the normal cardinality constraints. That is, if its cardinality is 0-1 or 1-1 the component is aggregate whereas if its cardinality is 0-N or 1-N the component is a set.

Most conceptual modelling formalisms which include complex objects [Kim et al, 1987; Lorie, 1983; Rabitti et al, 1988], model only *physical part hierarchies* i.e, hierarchies in which an object cannot be part of more than one object at the same time. In ERT, we extend this notion in order to be able to model also *logical part hierarchies* where the same component can be part of more than one complex objects.

To achieve this we define four different kinds of IS_PART_OF relationships according to two constraints, namely the dependency and exclusiveness constraints. The dependency constraint states that when a complex object ceases to exist, all its components also cease to exist (dependent composite reference) and the exclusiveness constraint states that a component object can be part of at most one complex object (exclusive composite reference). That is, we accommodate the following kinds of IS_PART_OF variations [Kim, 1989] :

i) dependent exclusive composite reference

ii) independent exclusive composite reference

iii) dependent shared composite reference

iv) independent shared composite reference

Note that we do not accommodate specific notation for these constraints. Their interpretation comes from the cardinality constraints of the IS_PART_OF relationship [Wangler, 1989a]. That is, assume that the cardinality of the IS_PART_OF relationship is (α,β). Then, $\alpha=0$ implies non dependency, $\alpha\neq0$ implies dependency, $\beta=1$ implies exclusivity while $\beta\neq1$ implies shareness.

Finally, the following rules should be obeyed concerning complex objects:

- Complex values may only have other values as their components. In addition, the corresponding IS_PART_OF relationship will always have dependency semantics unless it takes part in another relationship.

- Complex entities may have both entities and values as their components. Every component entity must be IS_PART_OF-related to the complex entity.

- Components, whether entities or values, may in turn be complex, thereby yielding a composition/decomposition hierarchy.

In the next section, we discuss the complex objects under the time dimension. In particular, we elaborate on how complex object hierarchies evolve over time and the constraints that should always be valid during this process.

3.4 Semantics of time stamping

The time period representation approach has been chosen because it satisfies the following requirements [Loki-86; Villain, 1982; Villain, 1986]:

1. Period representation allows for imprecision and uncertainty of information. For example, modelling that the activity of eating precedes the activity of drinking coffee can be easily represented with the temporal relation *before* between the two validity periods [Allen, 1983]. If we try, however, to model this requirement by using the line of dates then we will have problems since we do not know the exact start and ending times of the two activities.
2. Period representation allows to vary the grain of reasoning. For example, we can at the same time reason about turtle movements and main memory access times.
3. Humans comprehend periods of time much more easily than time points.

The modelling of information using time periods takes place as follows. First, we assign to each time varying object in our model (entity or relationship), an instance of the built-in class *SymbolPeriod.* Instances of this class are system-generated identifiers of abstract time periods e.g., SP1, SP2, etc. Members of this class can relate to each other by one of the thirteen temporal relations between periods [Allen, 1983]. In addition, these members can be restricted by instances of the class *CalendarPeriod.* Instances of this class are all the conventional calendric periods e.g., 10/3/1989, 21/6/1963, etc. with absolute specified start and end points.

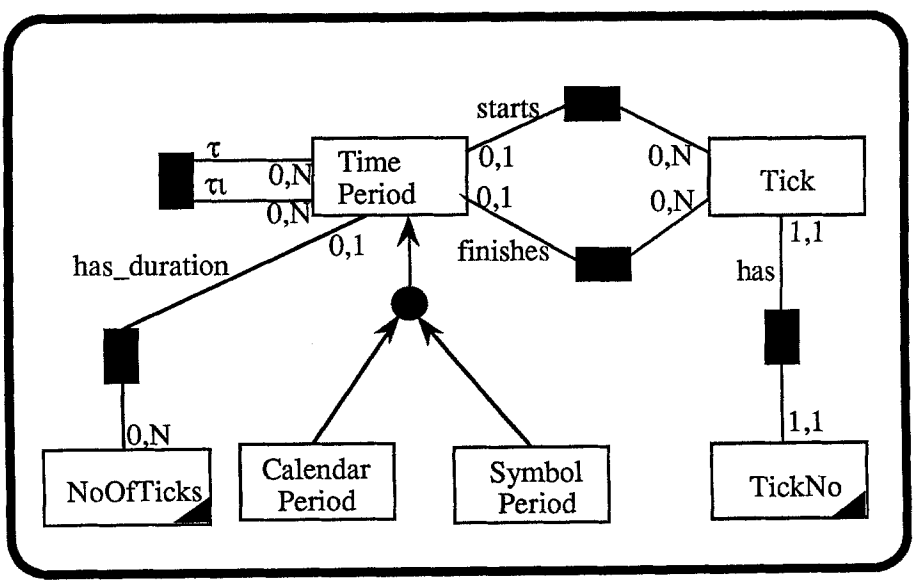

Figure 5. Time Period Metamodel

In figure 5, we show graphically the definition of these concepts. In this, the symbol τ represents

a temporal relationship and the symbol $\tau\iota$ its inverse. Also, in this figure we indicate the fact that the two classes *SymbolPeriod* and *CalendarPeriod* are disjoint. Note however, that the exact definition of the calendar period units is not included in this figure. The reason is because we want to keep it as simple as possible. For details the interested reader is referenced in [Wangler, 1989b]. The only think perhaps that we could add here is that for example, a date format like 21/6/1963 is just a shorthand notation of a calendar period.

According to the above discussion, we can time stamp the information in our conceptual model by using SymbolPeriod identifiers. However, we do not distinguish between time periods and time points. The fact that the abstraction level of a *SymbolPeriod* time stamp is say *day* can be inferred by its constraining temporal relations. For example, On the other hand, we can still represent explicitly in the conceptual schema the fact that an entity is time stamped only at the day abstraction level. This is done by distinguishing between different SymbolPeriod subclasses according to their abstraction level i.e., SP_D, SP_M,..etc. This last notation is not represented in the example of figure 3. We call this form of constraint a *resolution constraint* which when applied to a SymbolPeriod class restricts its members to calendar periods of the same resolution.

It is suggested that it would be convenient to represent directly in the conceptual schema some other notions of time such as duration and periodic time. The first consequence of this is that the expressive power of our external formalism is increased and also the readability of the schema. The definition of the duration class is shown in figure 6. Members of this class are simple durations expressed in any abstraction level. Each duration consists of an amount of calendar time units and it is uniquely identified by the combination of the real and calendar unit values. For example, the duration *"1,5 year"* is a valid duration according to our definition.

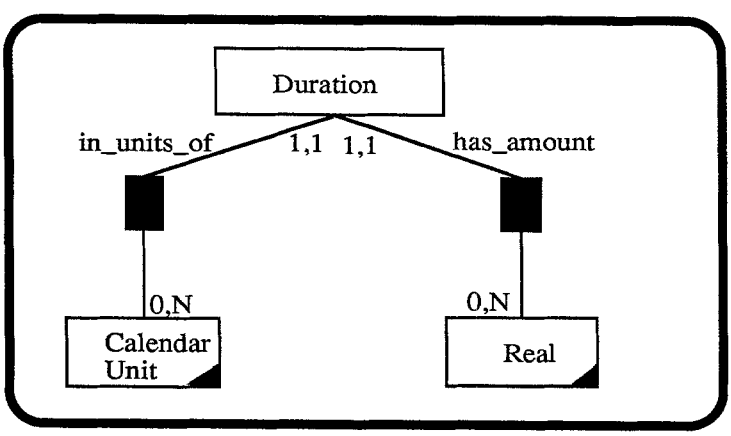

Figure 6. Metamodel of the duration class

The periodic time class is defined in figure 7. As shown in this figure, a periodic time has a base which is a calendar period, a duration and also it can be restricted by a symbol period. In other words, the interpretation of a periodic time can be expressed as "the base of every duration during symbol period". For example, the expression *"first week of each month during next year"* is a valid definition of a periodic time. In this case, the calendar period corresponds to "1-7 days", the duration corresponds to "1 month" and the restricting symbol period is the next year corresponding to [1/1/1991, 31/12/1991]. Finally, a periodic time is uniquely identified by the combination of its base and its duration.

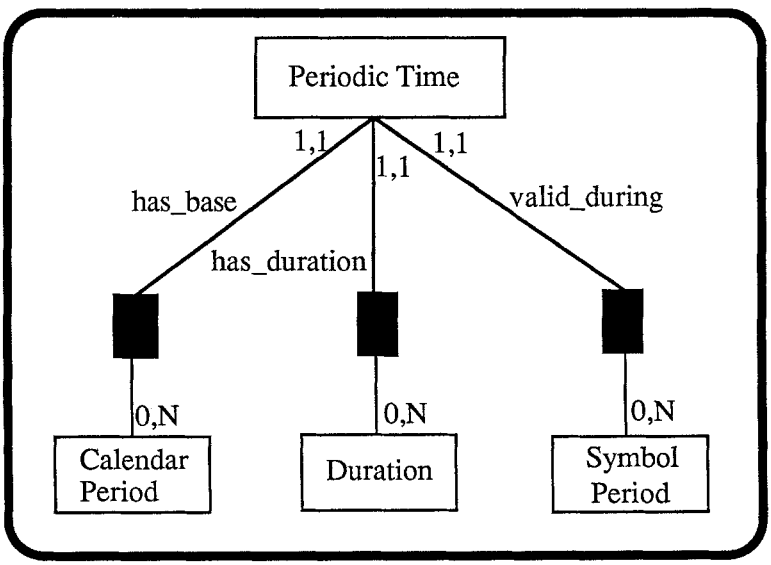

Figure 7. Metamodel of the periodic time class

In the sequel, we discuss how the previously presented time semantics interact with the other components of an ERT schema. First, we state some default assumptions for timestamping:

1) Reincarnation of entities is permitted and moreover, the entity keeps its identity through time.
2) Existence and validity periods should always be mapped onto the calendar time axis i.e, they should be specified in absolute terms. That is,
   - if the existence period of a timestamped entity is not specified explicitly as an absolute value then we take the current time as the starting point of its existence period.
   - if the validity period of a timestamped relationship is not specified

explicitly as an absolute value then we take as its starting point the most recent starting point of the existence periods of the two involved entities.

3) Non timestamped entities and relationships are assumed always existing i.e., from system startup time until now.

In ERT, we do not timestamp value classes and the IS_PART_OF relationships in a complex value class should always be time invariant. This is because an aggregation or grouping of values is defined through the participating value components. These assumptions affect the way that we map ERT to the relational model.

As discussed already, the validity period of a relationship should be a subperiod of the intersection of the existence periods of the involved entities. This does not hold for the ISA relationships where from the current semantics employed, we conclude that the existence period of the specialized entity should be a subperiod of the existence period of its generalization and that the ISA relationship is always time invariant.

Timestamping, when applied to derived ERT components has slightly different semantics than usual. Since, the derived components are not stored by default, the interpretation of timestamps refers to their corresponding derivation formulas. That is, if a derived component is not timestamped then the derivation formula returns the value of it at all times i.e, for every valid state of the database. Alternatively, for the timestamped derived components, the derivation formula returns a value which is valid for the existence or validity period of this component. i.e., the derivation formula must somehow refer to this period.

Finally, timestamping in a time varying IS_PART_OF relationship is translated to the following constraints (see [Theodoulidis, 1989] for more details). The dependency constraint in a time varying IS_PART_OF relationship boils down to:

i) The existence periods of the complex object and the component object should finish at the same time with the validity period of the IS_PART_OF relationship.

Also, the exclusiveness constraint is translated to:

i) If an object A is part of the complex objects B and C , then the period during which A is part of B should have an empty intersection with the period during which A is part of C.

Concluding, the above presented time semantics permit us to keep historical information for the UoD, include a strong vocabulary for expressing temporal requirements [McBrien, 1989] and also, model the evolution of complex objects through time in a natural way.

4. Conclusions

The aim of this paper was to present a data model which provides the expressive freedom required for the purpose of requirements analysis, perceived mainly as a real world modelling activity.

The ERT model as discussed in this paper is used to describe the structural components of the TEMPORA conceptual model. It contains some features which bring it in the state of the art among the other models. More specifically, it accommodates directly the representation and manipulation of time and complex objects in a uniform way.

Currently, we are working on the Rule and Process models of our formalism. In particular, we head towards a natural-language based External Rule Language enhanced with some form of abstraction mechanism(s) and a Process model able to express business functions in a simple and uniform way. A number of extensions have been planned including the introducion of history of rules.

Acknowledgments

The work reported in this paper has been partly funded by the Commission of the European Communities under the ESPRIT R&D program. The TEMPORA project is a collaborative project and the partners who contribute to this are: BIM, Belgium; Hitec, Greece; Imperial College, UK; LPA, UK; SINTEF, Norway; SISU, Sweden; University of Liege, Belgium and UMIST, UK. More specifically, SISU is sponsored by the National Swedish Board for Technical Development (STU), ERICSSON and Swedish Telecomm.
The authors wish to thank everybody involved in the TEMPORA project for their helpful discussions and suggestions during the first year of the project.

References

[Allen, 1983] Allen J.F. *Maintaining Knowledge about Temporal Intervals* CACM, 26(11) Nov.1983.

[Balzer et al, 1983] Balzer, R., Cheatham, T.E, Green, C. *Software Technology in the 1990's: Using a New Paradigm*, Computer, November 1983, pp. 39-45.

[Chen, 1976] Chen P.P-C. *The Entity-Relationship Model-Toward a Unified View of Data* ACM TODS vol.1 no.1, pp.9-36, March 1976.

[CRIS-1] *Information Systems Design Methodologies*, T.W. Olle, H.G. Sol and A.A. Verrijn-Stuart(eds), North-Holland, 1982.

[CRIS-2] *Information Systems Design Methodologies : Comparative View* T.W. Olle, H.G. Sol and A.A. Verrijn-Stuart(eds), North-Holland, 1983.

[CRIS-3] *Information Systems Design Methodologies : Improving the Practice* T.W. Olle, H.G. Sol and A.A. Verrijn-Stuart(eds), North-Holland, 1986.

[Dittrich, 1986] Dittrich K.R. *Object-oriented Database Systems: The Notion and the Issues* (extended abstract), Proc. OODB, Pacific Grove, Ca, Sept.1986.

[Dubois, 1986] Dubois E., Hagelstein J., Lahou E. et al *The ERAE Model : A Case Study* in CRIS-3.

[Dubois, 1987] Dubois E., Hagelstein J. *Reasoning on Formal Requirements: A Lift Control System*, Proceedings on S/W Specification and Design, 1987.

[Falkenberg, 1988] Falkenberg, E. *Knowledge-Based Information Analysis Support*, Proceedings IFIP TC2/TC8 Working Conference on 'The Role of AI in Databases and Information Systems', Canton, China, July, 1988, North Holland.

[Greenspan, 1984] Greenspan, S.J. *Requirements Modeling: A Knowledge Representation Approach to Software Requirements Definition*, Technical Report No. CSRG-155, University of Toronto, 1984.

[Kent, 1979] Kent W. *Limitations of Record-Based Information Models*, TODS, 1979.

[Kim et al, 1987] Kim W., Banerjee J., Chou H.T., Garza J.F., Woelk D. *Composite Object Support in Object-Oriented Database Systems*, in Proc. 2nd Int. Conf. on Object-Oriented Programming Systems, Languages and Applications, Orlando, Florida, Oct. 1987.

[Kim, 1989] Kim W., Bertino E., Garza J.F. *Composite Objects Revisited*, SIGMOD RECORD 18(2), June 1989.

[Loki-86] ESPRIT P107- LOKI, *A Logic Oriented Approach to Knowledge and Databases Supporting Natural Language User Interfaces* Institute of Computer Science, Research Center of Crete, Greece, March 1986.

[Lorie, 1983] Lorie R., Plouffe W. *Complex Objects and Their Use in Design Transactions*, in Proc. Databases for Engineering Applications, Database Week 1983 (ACM), San Jose, Calif., May 1983.

[Loucopoulos, 1989] Loucopoulos, P. *The RUBRIC Project-Integrating E-R, Object and Rule-based Paradigms*, Workshop session on Design Paradigms, European Conference on Object Oriented Programming (ECOOP), 10-13 July 1989, Nottingham, U.K.

[McBrien, 1989] P.J. McBrien *TEMPORA: Language Definition and Library*, TEMPORA report, Imperial College, E2469/IC/T2.1/5.

[Mylopoulos, 1986] Mylopoulos, J. *The Role of Knowledge Representation in the Development of Specifications*, In Information Processing 86, Kugler, H-J. (ed), Elsevier Science Publishers B.V. (c) IFIP 1986.

[Nijssen, 1988] Nijssen G.M., Duke D.J., Twine S.M. *The Entity-Relationship Data Model Considered Harmful*, 6th Symposium on Empirical Foundations of Information and Software Sciences, Atlanta, Georgia (USA), October 1988.

[Rabitti et al, 1988] Rabitti F., Woelk D., Kin W. *A Model of Authorization for Object-Oriented and Semantic Databases*, in Proc. Int. Conf. on Extending Database Technology, Venice, Italy, March 1988.

[Roman, 1985] Roman G-C. *A Taxonomy of Current Issues in Requirements Engineering* IEEE Computer 18(1), 1985.

[RUBRIC, 1989a] RUBRIC, ESPRIT Project 928, *Concepts Manual*, Nov. 1989.

[RUBRIC, 1989b] RUBRIC, ESPRIT Project 928, *Implementation Manual*, Nov. 1989.

[TEMPORA, 1988] TEMPORA Technical Annex, Oct. 1988.

[Theodoulidis, 1989] B.Theodoulidis *The IS_PART_OF Relationship Reconsidered (Working Note)*, TEMPORA report, E2469/UMIST/T1.1/15, Sept. 1989.

[Wangler, 1989a] Wangler B. *On the Semantics of Complex Objects in TEMPORA*, TEMPORA report, E2469/SISU/T1.1/13, October 1989.

[Wangler, 1989b] Wangler B. *On the Interpretation of timemarks in ERT schemas*, TEMPORA report, E2469/SISU/T1.1/14, October 1989.

[Van Assche et al, 1988] Van Assche, F., Layzell, P.J., Loucopoulos, P., Speltincx, G., *Information Systems Development: A Rule-Based Approach*, Journal of Knowledge Based Systems, September, 1988, pp. 227-234.

[Villain, 1982] Villain M.B. A *System for Reasoning about Time* Proceedings of AAAI-82, Pittsburgh, Pa., Aug.1982.

[Villain, 1986] Villain M.B., Kautz H. *Constraint Propagation Algorithms for Temporal Reasoning* Proc. of AAAI-86, 1986.