ABSTRACT

In recent years there has been a growing interest in the explicit introduction of time modelling in a conceptual schema. This has come about as a result of the realisation that the development of large information systems is becoming increasingly more difficult as user requirements become broader and more sophisticated. Arguably the most critical activity in the development of a large data-intensive information system is that of requirements capture and specification. The effectiveness of such a specification depends largely on the ability of the chosen conceptual model to represent the problem domain in such a way so as to permit natural and rigorous descriptions within a methodological framework. The explicit representation of time in a conceptual model plays a major role in achieving this effectiveness. This paper examines the ontology and properties of time in the context of information systems and conceptual modelling. In particular, a critical set of ontological assumptions about time are examined which in turn give rise to the way different paradigms deal with time modelling at the conceptual level. The paper describes and critically evaluates nine contemporary approaches to the specification and use of time in information systems.
1. Introduction

"What, then is time? If no one asks me, I know; but if I want to explain it to a questioner, I don't know"

Augustine of Hippo
(Confessiones XI, XIV)

The above quotation reflects the fact that humans have a natural perception of the effects of this cosmic reality but are unable to answer the philosophical question of "what there is". From a computing perspective this question could be transformed to a more tractable formal question of naming and quantification [Jardine & Matzov, 1986]. That is, to the assertion "there is something such that......".

A comprehensive survey of most of the research in these areas prior to 1982 can be found in [Bolour et al, 1982]. The study of time is increasingly becoming an important part of research efforts in a variety of strands within Computer Science. For instance, researchers in Artificial Intelligence have pointed to the need for a realistic world model to include representations not only of snapshot descriptions of the real world, but also of histories or the evolution of such descriptions over time (see for example [Allen, 1983;1984], [Allen & Koomen, 1983], [Allen & Hayes, 1985;1987], [Kahn & Gorry, 1975], [McDermott, 1982], [Villain, 1982], [Villain & Kautz, 1986], [Dean & McDermott, 1987] ). Special attention has been given to the planning problems where a model should provide the ability to reason with disjunctive and incomplete temporal knowledge (see for example [Dean, 1984; 1985; 1987]). Many different temporal reasoning algorithms have been proposed but none seems to be acceptable in practice due to efficiency problems.

Many logicians have regarded classical logic as an awkward tool for capturing the relationship and meaning of statements involving temporal reference (see for example [McArthur, 1976], [Prior, 1967], [Rescher & Urquhart, 1971], [vanBenthem, 1983]). Therefore, a number of temporal logics have been proposed (see [Rescher & Urquhart, 1971], [vanBenthem,1983] for a number of them), where, unlike classical logic, the same sentence can have different truth values at different times. These approaches may be broadly classified into those that extend the syntax and semantics of classical logics with a time component and those that adopt a more general perspective similar to that of modal logics and perform reasoning based on predefined temporal operators. The first class includes extensions to propositional, first order or even to intensional logic where, in general, time appears as a distinguished value type with more or less well-defined semantics. The second class
The Time Dimension in Conceptual Modelling

includes the so-called tense logics. Although many of these approaches do not define an ontology of time as such, they seem to be suitable in many applications.

From an information systems perspective, most implementors have to content with the fact that usually aspects of data that refer to time are neglected, treated only implicitly or explicitly factored out [Tsichritzis & Lochovsky, 1982]. None of the three major data models incorporates a temporal dimension. Thus, despite the fact that databases should model reality, conventional Database Management Systems (DBMS) lack the capability to record and process time-varying aspects of the Universe of Discourse (UoD).

However, it is increasingly becoming obvious that, for many applications, designers of information systems need to deal with the fact that when an information item becomes outdated, it need not be forgotten. With increasing sophistication of DBMS applications, the lack of temporal support raises serious problems in many cases. For example, conventional DBMS cannot support historical queries about past status, let alone trend analysis which is essential for applications such as Decision Support Systems (DSS). There is no way to represent retroactive or proactive changes, while support for error correction or audit trail necessitates costly maintenance of backups, checkpoints or transaction logs to preserve past states. There is a growing interest in applying database methods for version control and design management in computer-aided design, requiring capabilities to store and process time-dependent data. Without temporal support for the system, many applications have been forced to manage temporal information in an ad-hoc manner.

The need to handle time more comprehensively surfaced in the early 70's in the area of medical information systems where a patient's history is particularly important [Wiederhold et al, 1975]. Since these early days there has been a large amount of research in the nature of time in computer-based information systems and the handling of the temporal aspects of data. This work can be roughly divided under the two headings of:

- design and implementation of historical databases (HDBs)
- conceptual modelling

The first of these two areas has received considerable attention lately, since it has been argued that any attempt to implement systems which support temporal reasoning would need to first solve the problem of handling of historical databases [Ariav & Clifford, 1984]. However, the bulk of this research work is concerned with the design characteristics of an HDB. The implementation aspects are still under consideration [Ahn & Snodgrass, 1988], [Ben-Zvi, 1982], [Lum et al, 1984], [Dadam et al, 1984],[ Katz & Lehman, 1984].
The second area refers to both the formulation of semantics of time at the conceptual level as well as the development of models which support time.

Research interest in the time modelling area has increased dramatically over the past decade. A recent extensive bibliography [McKenzie, 1986] contained 80 articles from 1982 to 1986. At least 25 research groups are studying temporal aspects in Databases [Snodgrass, 1986b]. All this research activity simply verifies the argument that time modelling is indeed an indispensable requirement for most contemporary and future information system applications.

This paper is concerned with the examination and comparison of different approaches to treating time in conceptual modelling. The term *conceptual modelling* is viewed in this paper as referring to both the traditional view of semantic data modelling and the modelling in the context of requirements specification. The models examined in this paper which fall in the first category are: Infological Data Model, Conceptual Information Model, Time-extended Entity Relationship Model, Lundberg's Logic-based Model, Infolog, and Historical DataBase Model. The usage of a formal language for requirements analysis has been advocated in various existing approaches (see [Roman, 1985] for a survey of some of them). Although the "optimal" features of such a language are still a matter of debate, most of the researchers in the information systems area seem to agree that the explicit introduction of time is becoming a necessity as requirements for these systems become increasingly more sophisticated. Two recent developments in requirements modelling are considered in this paper, namely, the Entity Relationship Attribute Event (ERAE) Model, the Conceptual Modelling Language (CML) and the TEMPORA model.

The selected approaches are among the most representatives because they represent key developments of the different types of techniques for time modelling and reasoning in conceptual modelling. In addition, the particular approaches reflect in a way the evolution of this research subject starting from as early as 1973 and proceeding until today's advanced conceptual modelling formalisms.

The contribution of this work is a detailed analysis of why the different models adopt different approaches to time information modelling and reasoning, although addressing common application problems. The result of this analysis is a framework for evaluating the requirements of a general purpose time model for information systems.
The diversity of approaches to time modelling necessitates an examination of the possible ontological assumptions that these approaches make. This is the subject matter of section 2. Section 3 concentrates on the topic of time modelling at the conceptual level and sets up the framework of comparison by deriving a set of features against which our comparison will be carried out in section 5. In section 4, the chosen models are discussed in some detail. For each model emphasis is put on the assumptions that makes and the concepts that uses. Also, a partial specification of a retail company example is developed for each one in order to exemplify each particular approach. Section 5, compares and contrasts these models and provides two classifications schemes and finally, section 6 concludes the paper by highlighting the themes discussed.

2. Ontological Assumptions about Time

The literature on the nature and representation of time is full of never-ending disputes and many contradictory theories. This contrasts sharply with the commonly held view of time which does not prevent people in their everyday life to cope with the treatment of time. What this suggests is that there is some form of common sense knowledge about time that is rich enough to enable people to deal with the world, and that is universal enough to enable cooperation and communication between people.

In this section, the key issues concerning the nature of time are discussed together with its interaction with the non-temporal information from a conceptual modelling point of view.

2.1 The Nature of Time

One of the major issues involved in designing a conceptual model for information systems that has a built-in notion of time is "what is the nature of time and what are its properties". This is a question with a long philosophical tradition but should be treated as an ontological one in information systems theory. In fact, there are three core issues reflecting the nature of time which need examination:

i) Time Elements

An essential issue when considering a model or system for handling the temporal dimension of data is the nature of the time dimension itself. Basically, this issue comes down to whether time
The Time Dimension in Conceptual Modelling

should be modelled as *discrete elements* (such as integers) or as *dense elements* (such as rationals or reals). While there are proposals for both types of time, it appears that the use of discrete time points is more widespread. Note however that there is no limit to the subdivisibility of time since, in the authors' knowledge, there is no experimental evidence that proves the existence of a quantum of time.

A related issue concerns whether data values are associated with *points* in time or with *intervals*. Clearly this issue disappears when using discrete time, since the two representation schemes are equivalent.

ii) **Absolute and Relative Time**

Most of the research to date, has focused on modelling and managing *absolute time* in some form; that is, exact time points or intervals are associated with each data value. In Artificial Intelligence, however, a great deal of attention has been placed on *relative time*. For example, dealing with phrases such as "last week" and "a year ago" which appear in natural language understanding and story understanding systems. Clearly, the issue of relative time modelling impinges on the modelling of information systems and has to be considered alongside the absolute time modelling.

iii) **Linear and Nonlinear Time**

An issue of importance with respect to the nature of time in a scheduling context such as might take place in an office automation or job scheduling environment, is that of *periodic time*. The majority of time modelling approaches consider the time axis as being linear.

Any system that supports time should be able to model requirements which refer to the periodicity of the familiar time units of weeks, months, etc., and also to user-defined periods such as work-weeks, weekends, payroll periods, etc. Also, nonlinear time appears when there is a need to make assertions in alternative futures. This is useful in forecasting models and requires the use of *branching time*.

2.2 **The Interaction of Time and Information**

While it might be thought that the interaction of time and information is straightforward and only needs to be added to any system to provide the necessary facilities, the considerable body of research into this problem has shown that time interacts with information in a variety of different
The Time Dimension in Conceptual Modelling

ways, not all of which are as yet well understood. In this section, various aspects of this interaction are discussed.

Property types

There appear to be three primary relationships that can exist between time and the values of a property [Navathe & Ahmed, 1987]. Some properties are constant over time, for example surname and sex. The key attributes in databases are assumed to be of this type. Other properties are time-varying, taking on different values at different moments in time, such as salary or address. Finally, some properties take on values which are themselves times, for instance date-of-birth or promotion-date.

Object identification across time

A primary component of any information management system is the need to be able to provide unique and unambiguous identification of the objects been represented (called key in databases). Object identification takes on an added dimension since it is necessary to recognise that an object currently under examination is in fact the same as some other object already known to the database. Thus, there is need for a time-invariant identifier which can be provided either by the user or perhaps by the system. Its purpose is to ensure that a single real-world object is represented only once in the system, together with all of its time-varying or not properties.

Object "birth" and "death"

This issue has been addressed in the context of historical databases where there is a need to maintain a more-or-less permanent record of objects that are of current or historical interest to an enterprise (employees may be hired, leave, and subsequently be rehired). These "birth" and "death" events in the real world need to be adequately represented, for several reasons: to keep track of when objects are actively of interest, to be able to identify objects across time, to record multiple "incarnations" of the same object, and to save space when objects are inactive.

Temporal patterns of processes

A non-temporal predicate associated with time is assumed to name a process. Its semantics can be understood as activities that take place or even as facts that are true in the modelled UoD. In [Rescher & Urquhart, 1971] the following temporal patterns of processes have been identified:
• *homogeneous* is the process that goes on at all times throughout a period e.g., riding a horse.

• *majoritative* is the process that goes on at most times throughout a period e.g., your mother talking to the phone.

• *occasional* is the process that goes on at some times throughout a period e.g., writing a paper.

• *holistic* is the process that is inseparable from the period; it cannot occur in some subperiod e.g., baking a cake.

While almost all of the approaches have been concerned with homogeneous processes e.g., [Allen, 1983], the other types are also pragmatically meaningful and thus, are necessary to be modelled. There are however problems in order to understand the semantics of these temporal patterns and their relative ordering. Moreover, the boundary line between a majorative and an occasional process is clearly subjective and dependent on the process. These are some of the main reasons why these issues are still in their infancy and further research is required as to whether and how these issues are to be modelled.

**Event Time and Transaction Time**

A major aspect of the relationship between time and information is that there are multiple such relationships. Two such relationships have received the most attention. These have been called *event time*, i.e., the "real world" time at which a fact becomes valid, and *transaction time*, i.e., the time when this fact is recorded in the database. This distinction has also been referred to as extrinsic/intrinsic time [Bubenko, 1980], model/system time [LOKI, 1986] and data/transaction time [Clifford, 1987].

It seems that an ideal conceptual model should be capable of relaxing the assumption of temporal correspondence between the states of the database and the modelled environment. However, this necessity complicates things since there is a need to handle more than one dimension of time within the same system.

**Events**

In contrast to time-varying properties such as salary or address, another way in which time interacts with information is to record the occurrence of some *event* that takes place in real time. Rather than a property which can take on different values over time, an event is an object that
The Time Dimension in Conceptual Modelling

prevails for only one time unit. The notion of event is central to information systems conceptual modelling. Its importance comes from the fact that we wish to model not only an enterprise but also its environment and the interaction between these two. It is this last requirement which makes events necessary to be modelled since this interaction can easily be interpreted through events which have some preconditions, some triggering conditions and some resulting actions.

The most widely used definition of an event is *an instantaneous happening of interest to the enterprise*. However, there are also a number of other approaches to its definition. For example, McLeod in [McLeod & Hammer, 1981] distinguishes between point events and duration events. Also some authors e.g. [Sernadas, 1980], have defined time implicitly in the event structure of the system.

**Evolving Schemas**

A final aspect of the way in which time interacts with information concerns the relationship between time and the conceptual schema itself. When one considers the two components of the information system, the data itself and the metadata (the information about the data) it is clear that both of these components may evolve over time. The subject of schema evolution has received little attention up to now ([Clifford, 1987], [Martin et al 87], [McKenzie, 1990]).

These are some of the ways that time seems to interact with other aspects of information. This list surely, is not exhaustive; and the various ways presented here, are not mutually independent. Clearly, there is a need for a lot more research effort in order to obtain clear semantics for the issues discussed in this section.

3. **Time Modelling at the Conceptual Level**

3.1 **Overview**

An information system can be viewed as a model of some slice of reality about an organisation, the facts which exist in the organisation and the activities which take place [Borgida, 1986]. Therefore, the problem of developing an information system may be regarded as a problem of model description. These models fall mainly into two realms: the conceptual realm and the implementation realm.
Within the conceptual realm the models are independent of any physical considerations and are closer to the human perception of the application domain. A generic term for a model is in the conceptual realm is that of a conceptual model (or schema) [van Griethuysen et al, 1982]. A conceptual schema describes the static and dynamic components of an information system, their interrelationships and any constraints which may apply on these components. The term 'static components' refers to an information structure perspective i.e. 'what supports the organisational operations' and they are typically objects, and associations between these objects. On the other hand, the term 'dynamic components' refers to the usage perspective i.e. 'how the organisation operates' and they are typically viewed as events, actions, transactions and control structures.

According to Bubenko a conceptual schema represents abstractions, assumptions and constraints about an application domain [Bubenko, 1980]. More accurately, it represents a user's conceptualisation of the object system.

A conceptual schema is based upon a particular modelling formalism which encourages a developer to concentrate on the semantics of the application domain rather than the characteristics of the delivered system. Naturally, models oriented towards the description of a problem domain should be independent of any implementation considerations and should concentrate on the behaviour of the modelled component [Balzer & Goldman, 1979]. The importance of conceptual modelling has been emphasised by ISO [van Griethuysen et al, 1982] through the definition of a set of concepts and principles that should be applied to any proposed formalism.

The introduction of time into a conceptual database model can take place according to two different approaches. One is to extend the semantics of a preexisting snapshot model to incorporate time directly (built-in) and the other is to base the new model on a snapshot model with time appearing as additional attribute(s).

Naturally, the snapshot model used in most of the existing approaches is the well-known relational model [Codd, 1970]. This body of research has been addressed mainly in the context of Historical Databases (HDBs). The first of these approaches has been applied successfully in [Clifford & Warren, 1983], with the entity-relationship model to be used to formulate the intensional logic ILs. This logic serves as a formalism for the temporal semantics of an HDB much as the first order logic serves as a formalism for the snapshot relational model. Sernadas also has taken the same approach in defining the temporal process specification language DMTLT, which incorporates a special modal tense logic [Sernadas, 1980].
In a different approach, the snapshot relational model serves as the underlying model for the HDB. Each historical relation is embedded in a snapshot relation containing additional time attribute(s). In this approach the logic of the model does not incorporate time at all. Instead, the query language must translate queries and updates involving time into retrievals and modifications on the underlying snapshot relations. In particular, the query language must provide the appropriate values for these attributes in the relations been derived. For example in the Time Relational Model [Ben-Zvi, 1982], five additional attributes are appended to each relation. Several query languages incorporating time have been designed over the last decade. In [Snodgrass, 1987], there is an extension of Quel, called TQuel and also there is a comparison of ten query languages incorporating time.

With the advent of Semantic Data Models which try to capture more real-world meaning than the relational model, the research in this area has been directed to a more wide range of problems. The introduction of time in these proposals however has been addressed at the very early stages. The ERAE data model ([Dubois et al, 1986], [Dubois & Hagelstein, 1987] is an attempt to extend the semantics of the entity-relationship model [Chen, 1976] with a distinguished type Event in its basic constructs. Also the CML language which is based on the TAXIS model [Greenspan et al, 1983] and the TEMPORA model include time as a primitive notion. All these models have the characteristic that they are intended to be used in the requirements definition of information systems and have been developed within projects which aim at a formalisation of the life cycle of software development.

Other Semantic Data Models which have addressed this issue include the Semantic Data Model (SDM) [McLeod & Hammer, 1981] in which relationships may or may not have a time component, the RM/T [Codd, 1979] which is an extension of the traditional relational model to handle among others sequencing of events, the Functional Data Model [Shipman, 1981], the Event Model [King & McLeod, 1984], and many others.

### 3.2 Framework for comparison

According to [Kung, 1983], any conceptual model should support the following five features:

- Understandability,
- Expressiveness,
- Processing independence,
- Checkability and,
Because a conceptual model is a common reference framework for users, designers and implementors, its understandability is very important. That is, the conceptual model should include user-oriented concepts and constructs and should be easily readable whether it is formally or informally described. However, by employing a high informal description language like a natural language, the ambiguity of the expressions increases and at the same time, this leads to increase in understandability problems. Thus, unambiguity is required because it will be difficult or impossible to understand properly what is really meant by an application description. Finally, the understandability feature also involves the aspects of clarity and intuitivity. Clarity means that redundancy of expressions are allowed only for enhancing the understanding of the application but, not imposed by the concepts and constructs used. Intuitivity assumes some representation means e.g. graphs or tables, for improving the understanding of the system description.

The second feature has to do with expressive power of a modelling formalism. This refers to whether the concepts and constructs used are powerful enough to describe everything that needs to be specified, without much effort from the specifier. Furthermore, the model concepts should include the time perspective since this is required by most applications. However, specification of the time perspective is not required only by the application domain but also, improves the model understandability [Bubenko, 1980].

The third feature implies that the conceptual model should be free as much as possible from data processing considerations. A conceptual model is a description of some part of the reality. Thus, the real world evolution is reflected by a sequence of operations on the conceptual database. This however, procedural specification is clearly very restrictive and should be avoided.

The checkability feature concerns the verification of the model i.e., that it does not contain any inconsistencies, the validation of the model i.e., that it does reflect the modeller's intentions and finally, the testability of the model i.e., that it is possible to determine whether the proposed implementation satisfies the information requirements.

The last feature refers to the schema evolution issue discussed in section 2.2. The necessity for changeability can be summarised in the quotation by Heraclitus "Nothing is permanent except change". To achieve high degree of changeability, the model should be localised and loosely structured. That is, a few components of the schema should be affected when a change occurs and also, the components should be easily added and removed and the schema readjusted.
Ideally, a complete examination of the relative merits of any modelling formalisms should deal with all these features. However, these features are rather general and of a subjective kind. In addition, our objective is to examine the approaches in terms of their time perspective. As a consequence, we concentrate on the following generic characteristics of a time model:

- formal semantics,
- expressive power,
- proof theory.

To this end, we provide two classification schemes. The first one deals with the nature of time that each approach assumes i.e., the answer to the question "What is time?" and the second one deals with temporal functionality issues like primitive temporal notions, temporal reasoning formalisms, temporal metrics, incomplete temporal specifications etc.

Another possible characteristic that could be discussed is that of the utilisation of each model to an application area according to its time perspective. However, the classification of the application areas according to their temporal needs is a difficult task due to the fact that similar applications may have different temporal information requirements.

4. Conceptual Models

This section discusses the ontology and properties of time in a number of conceptual modelling approaches. The discussion is centred on the kind of temporal semantics that each model adopts and how time is introduced in each model. The discussion starts with the earliest of these approaches and proceeds to the most recent ones.

Before people realised why the explicit introduction of time in any conceptual model is necessary, most of the approaches were considering dates as ordinary entities or values. Thus, there was no need to provide built-in mechanisms to record and process time varying information. When the application programmer was faced with the need to record and use time varying information, then he had to deal with all the burden of defining the appropriate constructs for time representation and manipulation. Moreover, the adopted time semantics were depending on the particular application and thus they lacked generality. Space limitations prevent us from presenting every non-time related aspect of the approaches. However, the major modelling concepts and constructs are
The Time Dimension in Conceptual Modelling

4 described in an informal way through a simple retail company example. Only a partial solution of the example application is shown for each of the approaches.

For the purposes of this discussion, we will assume that in a retail company there are products, customers, orders, deliveries, shipments, etc. Products are identified by product numbers (PNO), have prices and quantities-on-hand (QOH). Customers are identified by customer numbers (CNO), have names (CNAME), addresses (ADDRESS) and accounting balances (BALANCE). Orders are placed by customers. Each order has a unique order number (ORDNO), and specifies the product and the quantity ordered. There are shipments of products from the suppliers as well as deliveries of products to customers. Shipments and deliveries affect the quantity-on-hand of the products. If there is not sufficient quantity of a product then the corresponding order is suspended temporally, e.g., it is treated as a back order. Finally, there is a rule stating that no order will be delivered if it makes the customer in question out of balance by more than £5,000.

4.1 The Infological Data Model

The earliest of the conceptual data models which recognised time to be indeed a distinguished entity is the infological data model [Langefors, 1973], [Langefors & Sundgren, 1975]. It is based on the natural human perception of elementary facts (e-facts). Specifically, the concepts of an object, a property (or relation), and time are associated to form an atomic e-fact which is assumed to be the "building block" of human knowledge. Storing such an e-fact in the information system results in an elementary message (e-message), which includes the references to an object, a property (attribute or relation), and a time point or period.

Time in the infological model is, so to speak an attribute that provides a built-in temporal context for every fact recorded in the information system. For example, if O(p) is the set of all objects that can have property p regardless of time, then the O_t(p) is the set of all objects having property p at time t. The set O_t(p) is a subset of O(p).

Formally, an e-message is defined as "the message of which the property part states an elementary property or an elementary behaviour" [Langefors & Sundgren, 1975]. That is, an e-message is the smallest piece of information in the sense that if any part of what it makes known is ignored, then the rest does not carry information.
To make known what object an e-message is about we use an identifier or a name. That is, the identification of an object has the format \(<\text{object class name, identifier}>\). For our retail company example this might correspond to:

\(<\text{PRODUCT, P12}>, <\text{CUSTOMER, C22}>, <\text{ORDER, ORD32}>, \ldots, \text{etc.}\)

To make known what property an e-message states (of the object it informs about), a device similar to that used for objects is employed in that it is assumed that properties can be assigned to classes. Thus, an attribute or a relation is a kind of property and it is naming a property class. Typically a property is determined by a (attribute,value) pair or a (relation,object) pair. In our example this might correspond to:

\(<\text{CNAME, Babis}>, \ldots\>
\(<\text{PRICE, £100}>, \ldots\>
\(<\text{HAS\_NAME, CNAME}>, \ldots\>
\(<\text{QUANTITY, 10}>, \ldots, \text{etc.}\>

The conceptual schema for our example thus, might look like in figure 1. In this figure the bubbles represent objects of our application e.g., Product, Customer and Order. Each object must have an identifier and optionally, a number of properties i.e., attributes, relations or events. In figure 1, there are only attributes of objects that are represented as filled dots and their value domain is indicated as \(<\text{value domain}>\). Objects are linked to properties with pointed arrows that name the properties of these objects.

Concluding, a typical e-message consists of:

i) name of object class, identifier within class. This is repeated a number of times if subclasses of object classes are used.

ii) name of property class (attribute, relation or event), value of attribute.

iii) name of time attribute, value of or identification of time.
Examples of e-messages for our retail company example might be:

\[(\text{CUSTOMER,C22)}, (\text{ADDRESS,Palatine Road}), (\text{since date, 15/3/1989})\],

\[(\text{ORDER_TRANSACTION,ORD32)}, (\text{QUANTITY,100}), (\text{on date, 12/3/1989})\],

\[(\text{CUSTOMER,C22)}, (\text{NAME, Babis}), (\text{since date, 21/6/1963})\],......etc.

In conclusion, we can say that although time is identified as one of the three primary components of human knowledge, its role in the initial infological approach was limited to the mere indexing of recorded facts.

4.2 The Conceptual Information Model

The notion of "infological time" was refined in a later approach [Bubenko, 1977] in that the time need not be a component of all relationships. An important distinction is made in this approach between extrinsic and intrinsic times which are defined as follows:

- **Extrinsic time** reflects the fact that every statement is embedded in a temporal context whether or not this context is formally preserved. For example, if the statement "Babis is a postgraduate student" was made in December 1988, then we associate the extrinsic time-index "Dec1988" with it.
The Time Dimension in Conceptual Modelling

Intrinsic time is part of the content of a statement. For instance, if the statement "Babis is a postgraduate student on December 1988" was made in January 1989, then there is an intrinsic time in this statement, namely "Dec1988".

A much broader perspective of information modelling and time as a modelling construct was developed later by Bubenko [Bubenko, 1980]. This is called Conceptual Information Model (CIM) (see also [Gustafsson et al, 1982]). In this perspective, a time model is specified before any other concept or data modelling construct is defined, reflecting the primary nature of the temporal context.

In this approach, there is a denumerably infinite set of time points \( T \) and a set of time interval types \( TI \). While \( T \) is considered to be the set of real numbers, the set \( TI \) can be defined according to the particular needs of the application. Thus if we want to model the molecular world (if it is possible), we should choose for the set \( TI \) something like \{....picoseconds, nanoseconds, microseconds, milliseconds,......\}. However, for most applications the set \{seconds, minutes, hours, days, months, years\} will be sufficient.

A time interval type such as month is the set of all one month periods, e.g. June 1963. A partial ordering relation is defined on the interval type set so that if the relation holds between two time interval types, then the elements of the first type are parts of the elements of the second type.

In summary, CIM focuses on hierarchical aspects of the structure of time and provides a formal calendar system with strict, unambiguous hierarchy of time periods. Also in this approach, the existence of an object is defined to be true between two specially designated events. For example, someone is an employee from the time of his hiring until his firing. In general, events play a fundamental role in determining the behaviour of objects and should remain indefinitely recorded in the database.

In CIM, the major design concepts and constructs are entity types, event types, relationship functions and global constraints. For our retail company example, we might model the UoD as:

entity types : PRODUCT, CUSTOMER, ORDER,......

event types : ORDER-ARRIVAL, PRICE-CHANGE, SHIPMENT, DELIVERY,......

constraints : "no order will be delivered if it makes the customer in question out of

The specification of the event types ORDER-ARRIVAL and PRICE-CHANGE may appear as in figure 2 and the specification of the entity type PRODUCT as in figure 3. Notice that the attributes of events
The Time Dimension in Conceptual Modelling

are time independent but an occurrence time for each event must be specified. As shown in figure 2, for each event, a set of attributes are defined that are used as parameter passing from the external world to the information system. In addition, a subset of these are defined as the identifier of an event in that their values are used to uniquely distinguish each event occurrence. In figure 2, the two events defined are EXTERNAL meaning that they occur outside the system in contrast to internal events that occur inside the system.

![Event specifications in CIM](image)

As seen in figure 3, for each entity type, a set of stored or derived attributes are defined and in addition, some of the stored attributes are designated as the entity identifier. Each attribute function of an entity type may have a properly chosen time argument e.g., price(D). In addition, if the function is assumed to be derivable then a derivation rule must be defined. For example, the price of a product in figure 3 is defined as the price stated in the last price change event referring to the particular product instance.

The example order constraint can be formally specified as:

\[ \exists x \exists y \exists z \exists t ( \text{DELIVERY}(x) \land \text{CUSTOMER}(y,t) \land \text{PRODUCT}(z,t) \land \text{cust}(x)=y \land \text{product}(x)=z \land (\text{qty}(x) \times \text{price}(z) + \text{balance}(y) \leq \£5,000)) \].

Finally, the perception of time as isomorphic to the real numbers illustrates the point that handling time requires special care and cannot be added in an ad hoc manner to a traditional data model. Bubenko in his approach argues that time should be introduced through recording the information at times of change. This method guarantees that the database includes a finite number of time point references.

![Entity PRODUCT](image)
The Time Dimension in Conceptual Modelling

price(D) : MONEY;
derivation rule : \( \forall x \forall t \forall d \ (\text{PRODUCT}(x,t) \land D(d) \land \text{day}(t) = d \land \exists y \ (\text{PRICE-CHANGE}(y) \land \text{product}(y) = x \land \text{day}(y) \leq d \land \exists z \ (\text{PRICE-CHANGE}(z) \land \text{product}(z) = \text{product}(y) \land \text{day}(z) \geq \text{day}(y))) \rightarrow \text{price}(x,d) = \text{price}(y) \);

qty(t) : NONNEGATIVE-INTEGER;
derivation rule : ..........

identifier : pno;
end;

Figure 3. The specification of the entity type PRODUCT in CIM.

4.3 The Time-extended Entity Relationship Model

Bubenko's approach fails to reconcile infinite conceptual models with finite machines and finite memories. A later attempt, the Time-extended Entity Relationship Model (TERM) [Klopprogge, 1983] addressed this issue by focusing on the definition of a history-structure which is a data construct designed to capture special properties of time.

The semantics of time adopted in this approach is not discussed in [Klopprogge, 1983]. However, a careful examination leads to the conclusion that it is similar to that found in Bubenko's approach. That is, it is assumed that time is isomorphic to the real numbers.

TERM concentrates on a set of modelling primitives based on the constructs of the Entity-Relationship (E-R) model [Chen, 1976]. The notion of history structure introduced in the model, augments the basic E-R constructs to create new, extended constructs (e.g. attribute-history and role-history). These are called atomic histories and can be formulated as a function from a time domain into some range of values. An entity (or relationship) history is called a composite history and is represented by the histories of the attributes (and roles) that make up this type of entity (or relationship). Time in TERM is captured within the regular E-R modelling constructs i.e., under each basic fact (attribute or relationship) one can find its history. This is a totally different perspective from the models mentioned in the two previous sections, where each point in time "owned" the corresponding description of the state of the world at that time.

The definition of a conceptual schema in TERM consists of the definition of auxiliary declarations (for constants and functions that are global) and, of definitions of structures, time structures, value structures, history structures, structure transformations, entity types and relationship types.
The Time Dimension in Conceptual Modelling

20

A structure is a set with a number of relations defined on its elements and can be restricted by a where clause. A time structure is any structure with a time interpretation and a value structure is any structure that represents values. A partial definition of the date time structure is shown in figure 4.

```
time structure date;
  record m, d, y : integer end
/*this time structure describes calendar dates*/
structure relations
  function before(t1,t2 : date) : boolean; /*simple temporal order*/
  function same(t1,t2 : date) : boolean; /*simple comparison for equality*/
structure operations
  function next_day(t : date) : date; /*returns the next day*/
  function previous_day(t : date) : date; /*returns the previous day*/
  function add(t : date, i : integer) : date; /*adds i days to a date*/
  function distance(t1, t2 : date) : integer; /*determines the number of days between t1 and t2*/

where not (element.m=2 ∧ element.d=29 ∧ element.y mod 4=0 ∧ element.y <> 1900
  ∨ element.m=2 ∧ element.d=30
  ∨ element.m in {2,4,6,9,11} ∧ element.d=31
  ∨ element.m > 12
  ∨ element.d > 31); /*disallows illegal dates*/
```

Figure 4. The definition of the date time structure of TERM

A history structure is a set of (time, value) pairs. For each history structure we define history relations which are boolean functions on histories and also history operations. There are three kinds of history operations, namely, the base operation, a derivation operation and the inductions. The base operation is the insertion of a new fact and it is defined implicitly. A derivation operation is a function that takes the stored states of a history and a set of times and determines the state for each in the latter set. Finally, a history induction is a function which approximates not explicitly recorded states i.e., splines, step-functions, least-square,....etc (see [Lundberg, 1983] for more details). Another concept contained in the TERM schema is time, value and history structure transformations; these are functions which accept input from one structure and produce output of another.

Finally, the schema is completed with the definition of entity and relationship types. The definition of the entity type PRODUCT and the relationship type DELIVERY is shown in figure 5 together with the definition of the history structure ex.

The basic operation in the TERM model is the addition of a state to a history structure. An observer can issue a database registration transaction either to initiate or to complete the history
of an entity or relationship. The addition operations of observers are verified by testing the so called consistency assertions. Observation errors beyond the scope of these assertions must be detected in some other way. In TERM there is a privileged authority called referee who has the ability to correct such observation errors. This authority is assumed to be a priori correct. The correction transactions issued by the referee can erase or alter facts already stored and moreover, the new facts are assumed to overwrite the old ones.

**entity type** Customer;

- **existence** ex;

- **attributes**
  - cno constant integer;
  - cname constant string;
  - address variable location;
  - balance variable pounds;

**relationship type** Delivery;

- **attributes**
  - qty constant integer;
  - debit constant pounds;

- **roles**
  - ofOrder constant ↑ Order
  - forProduct constant ↑ Product

**history structure** ex: date x Boolean

- where (exists b,e in history, all x in history)
  - (x.v=false and (before(x,b) or becon(e,x)))
  - or x.v=true /*b marks the beginning and e the end of existence
  - or x.v=false and before(x,b) or x.v=true /*the existence history is completely defined
  - or x.v=true and before(x,e) or x.v=false /*the end is still undefined
  - or history = {} /*the existence history is undefined
  - or x.v <> nil or least-recent(history,x)= nil
  - or least-future(history,x)=nil);

**history derivation**

- **function** derivex(h : ex, t : set of date) : ex;
  - begin ........end;

returns the state for each date

Figure 5. Definitions of an entity type, a relationship type and a history structure in TERM.

The above process introduces the notion of recording history which represents the history of the recording operations and is effectively a temporal sequence of initiate, complete, alter or erase operations. Thus the information system has to manage two time dimensions which correspond to the event and transaction time dimensions. One is the recording or internal history and the other is the external mini world history.

Any state of the recording history is an approach to the representation of the UoD history. However is only an approach because we may have observational errors. Probably the best available approach to reality is represented by the latest state of the recording history. This view recognises that the database itself is an object whose history is of interest. The model should not
only concerned with the history of the changes to the objects in its domain but should also deal with the development of the content of the database over time.

4.4 Lundberg's Logic-based Model

A logic-based database model which includes the notion of time from the onset, is presented in [Lundberg, 1982]. In this approach the time model adopted consists of an infinite set of discrete time points i.e., isomorphic to the set of integers. Lundberg argues that even if the nature of physical time is usually considered to be continuous, in information modelling, time can be considered to be discrete for a number of reasons. Firstly, the references concerning time are usually made in discrete terms. For example, we say that the hiring of an employee occurs during a day. Secondly, when references to locations in time are made their representation must be finite, e.g., in a computer system or on a piece of paper. That is, we do not draw a curve for the inflation during 1988 but we simply represent its value at different time points. Finally, from a modelling point of view time intervals may be considered to be point like. That is, instead of the interval from 1/7/1989 to 31/7/1989, we can consider the time point "July 1989" (see also [Bubenko, 1980]).

Lundberg presents an axiomatization for the time points within the framework of first-order predicate logic, based on an ordering predicate (earlier_than) and the identity relation (=). Also he defines the successor relation and the axioms that hold for it. That is, if we assume that $tpt(x)$ denotes that $x$ is a time point, then for the ordering predicate we have the axiom:

$$\forall x \forall y (et(x,y) \rightarrow tpt(x) \land tpt(y)).$$

Also, for any two time points it holds that one is earlier than the other or that they are identical, i.e.,

$$\forall x \forall y (et(x,y) \lor (x=y) \lor et(y,x)),$$

and that the relationship between time points is transitive, i.e.,

$$\forall x \forall y \forall z (et(x,y) \land et(y,z) \rightarrow et(x,z)).$$

The successor relation for time points is represented by the predicate $ss(x,y)$ which denote that $y$ is the immediate successor of $x$. For this predicate it holds:
The Time Dimension in Conceptual Modelling

\( \forall x \ \forall y \ (ss(x,y) \rightarrow tpt(x) \land tpt(y)). \)

Also, the reverse holds, i.e.,

\( \forall x \ \forall y \ \forall z \ (ss(x,y) \land ss(z,y) \rightarrow (x=z)). \)

Furthermore, all time points have a successor (infinite time), i.e.,

\( \forall x \ \exists y \ (ss(x,y)) \)

and the relationship between the \( et \) and the \( ss \) predicate is defined as

\( \forall x \ \forall y \ (ss(x,y) \rightarrow et(x,y)). \)

Lundberg argues that the concept of event is central to the temporal dimension of an information system. An event is defined as a phenomenon (i.e., a decision or an action) that takes place at a certain time point. For example, in our retail company example the \( order\_arrival \) and the \( price\_change \) are events. To complete the modelling of events we have to represent not only the event types but also the rules that hold for sequences of interdependent events and their effects.

Our retail company example can be represented in a first-order predicate logic language with the following predicates:

- \( PRODUCT(x,y) \) /* pno(x) and day(y)
- \( PRICE(x,y,z) \) /* pno(x), pounds(y) and day(z)
- \( QOH(x,y,z) \) /* pno(x), integer(y) and day(z)
- \( CUSTOMER(x,y) \) /* pno(x) and day(y)
- \( NAME(x,y) \) /* pno(x) and string(y)
- \( ADDRESS(x,y) \) /* pno(x) and string(y)
- \( BALANCE(x,y,z) \) /* pno(x), pounds(y) and day(z)
- \( ORDER(w,x,y,z,t) \) /* ordno(w), pno(x), cno(y), qty(z) and day(t)
- \( BACKORDER(w,x,y,z,t) \) /* ordno(w), pno(x), cno(y), qty(z) and day(t)
- \( ORDER\_ARRIVAL(x,y,z,t) \) /* pno(x), cno(y), qty(z) and day(t)
- \( PRICE\_CHANGE(x,y,z) \) /* pno(x), pounds(y) and day(z)

In order to complete our example we have to state the rules concerning the sequences of events and their effects. For example, we can state that if an order cannot be satisfied then it is treated as a backorder. This can be expressed as:

\[
ORDER\_ARRIVAL(x,y,z,t) \land QOH(y,z',t) \land (z>z') \rightarrow BACKORDER(w,x,y,z,t).
\]

Also, the order integrity constraint can be formulated as:
ORDER_ARRIVAL(x, y, z, t) ∧ BALANCE(x, s, t) ∧ PRICE(y, p, t) ∧ ((s + p * z) ≥ 5,000) → BACKORDER(w, x, y, z, t).

4.5 Infolog and DMTLT

Another logic based approach to temporal semantics, but using tense logic is presented in [Sernadas, 1980]. In this approach, a number of modal tense quantifiers is added to the usual propositional logic and a rich set of temporal operators is provided in order to handle time. The future is ignored so the message-oriented relational model defined is concerned only with the past and the present. As a consequence, a number of problems associated with the introduction of branching time are avoided.

According to the INFOLOG approach, an information system is an independent agent that keeps its own database, may communicate with other systems through messages and performs some activities. The concept of event is introduced as an abstraction for the modelling of the atomic operations (either database updates or message-exchange operations), whereas the state of a system is seen as the sequence of all events that have already occurred, starting with the system's birth event.

The time structure is assumed to be linear and consisting of a set of worlds (or states) with a strict total order $R$ (i.e., $<$) over them. A number of modalities are defined including $G$ (always in the future), $Go$ (in the present and always in the future), $H$ (always in the past), $Ho$ (in the present and always in the past) and the until operators $U$ ($\alpha$ is true until $\beta$ is true) and $Uo$ ($\alpha$ is true in the present and until $\beta$ is true).

The basic modelling concepts and constructs found in DMTLT are concurrent processes and multityped relations. A partial solution of our retail company example can be defined as shown in figure 6.

The process $price\_modify$ specifies that

for every product number pno and price p,

whenever there is a price change whose product number is valid

then there occurs a transition that turns the previous product assertion

into false and creates a new assertion with the new price.
The Time Dimension in Conceptual Modelling

RELATIONS
CUSTOMER(CNO,CNAME,ADDRESS,BALANCE);
PRODUCT(PNO,QOH,PRICE);
BACKORDER(ORDNO,CNO,PNO,QTY);
SHIPMENT(PNO,QTY);
DELIVERY(ORDNO,PNO,QTY,DEBIT);
ORDER_ARRIVAL(CNO,PNO,QTY);
PRICE_CHANGE(PNO,NEWPRICE);

PROCESSES
PROCESS price_modify
PARALLEL FOR pno, p;
PRICE_CHANGE(pno,p)
(∃ q ∃ x PRODUCT(pno,q,x)
⇒ (pno,q,p) → PRODUCT(pno).

PROCESS backorder_processing
PARALLEL FOR c, pno, q;
SEQUENTIAL FOR o;
BACKORDER(o,c,pno,q)
G(BACKORDER(o,c,pno,q)
(∃r SHIPMENT(pno,r))
∃ca ∃ b ∃ cn CUSTOMER(c,cn,ca,b)
∃p ∃ x PRODUCT(pno,x,p)
x ≥ q
p*q-b ≤ 5,000
⇒ (o,c,pno,q) ← BACKORDER
(pno,x-q,p) → PRODUCT(pno)
(o,pno,q,p*q) → DELIVERY

Figure 6. Schema Definition in DMTLT.

The process backorder_processing specifies that

if there is a backorder
and after this has been created there happened some shipment of the product
and the current stock is sufficient
and the balance of the customer does not exceed £5,000
then the backorder is invalidated
and the quantity-on-hand is adjusted
and a delivery is generated.
An information system is seen by the DMTLT approach as a network of concurrent processes that exchange messages among themselves and with the environment through the temporal database of the information system. The temporal database is a family of database states, whose indices are made implicit. The processes in the system may only change the current state of the temporal database and may refer to all the previous states.

This approach is further refined in [Carmo & Sernadas, 1987]. According to this, a layered approach is proposed for the specification and verification of information systems. The dynamic requirements of this approach are described by a unifying temporal logic framework which uses linear formalisms in some layers and branching formalisms in other layers.

### 4.6 HDBM and ILs

The intentional logic ILs is used in Clifford's approach [Clifford, 1983] in order to describe the semantics of his extended relational model. Time is introduced into the Historical Database Model (HDBM) by the inclusion of two special fields STATE and EXISTS? in each relation. The value of STATE is some time period and the value of EXISTS? is 0 or 1, depending on whether the relationship represented by a tuple exists during the period specified in STATE.

The set of times is treated as dense so we again have the problem of mapping the continuous model to a finite representation. To overcome this problem, Clifford identified three major principles that must be addressed in any practical implementation of an historical database system.

The first of these was the notion of a completed relation which will have a tuple in each state for every entity. That is, even if a fact does not hold (i.e., EXISTS?=0) in a particular state, there is a tuple in that state with the same key attribute and a special null value for the non-key attributes.

The second of these was the Comprehension Principle, a three-dimensional analog of the closed-world assumption which stated that:

> Any and all information about the objects of interest to the enterprise can be assumed to be contained in or implied by the historical database for the entire time period that the database models reality.

Finally, the need for explicit Continuity Assumptions was identified. These are functions which are associated with time-varying attributes in the model, and define a mapping from a partial specification of their value (e.g., from sampling at a small number of points) to a complete specification of their value over some time interval.
Our retail company example can be formulated in HDBM with the following relations where the underlined attributes are the key fields of each table.

- `CUSTOMER_REL(STATE, EXISTS?, CNO, CNAME, ADDRESS, BALANCE)`
- `PRODUCT_REL(STATE, EXISTS?, PNO, PRICE, QOH)`
- `ORDER_REL(STATE, EXISTS?, ORDNO, CNO, PNO, QTY)`
- `BACKORDER_REL(STATE, EXISTS?, ORDNO, CNO, PNO, QTY)`
- `ORDER_ARRIVAL(STATE, EXISTS?, CNO, PNO, QTY)`
- `PRICE_CHANGE(STATE, EXISTS?, PNO, NEW_PRICE)`

Also, the integrity constraint that no order will be delivered if it makes the customer in question out of balance by more than £5,000, can be formulated in ILs as in figure 7. In this figure, the `ORDER_ARRIVAL`\* , `PRODUCT`\* , `CUSTOMER`\* , `BACKORDER`\* are Entity Existence Constants (EECs) and the qty', price', balance', cno', pno' are Role Constants (RCs). An EEC denotes for each relation and for each state the set of entities specified through the key attributes which exist in that state. An RC denotes for each non key field in a relation and for each state a function from a set of states to individuals [Clifford, 1983].

\[\forall st, cn, pn, qt \ \text{ORDER\_ARRIVAL\_}\* (st,cn,pn) \land \text{qty'(st,qt)} \land (\exists \text{PRODUCT\_}\* (st,pn) \land \text{price'(st,pr)}) \land (\exists \text{CUSTOMER\_}\* (st,cn) \land \text{balance'(st,bal)}) \land ((\text{bal} + (qt*pr)) \geq \£5,000) \land (\exists \text{BACKORDER\_}\* (st,ordno) \land \text{cno'(st,cn)} \land \text{pno'(st,pn)} \land \text{qty'(st,qt)})\]

Figure 7. Our example constraint in ILs.

Summarising, this work has been addressed in the context of natural language querying systems. Although it is formally developed, the way that it describes the time semantics is very simplified. For example, the step-function continuity assumption that is adopted, is clearly restrictive. Also the tuple time stamping, even if it is a fairly simple idea, it is not adopted by many researchers. One reason is because information is lost when combining relations and another is that there is a high redundancy of "duplicated" and thus, unused information when the implementation issue is concerned. This work has been extended later by Clifford and others (see for example [Clifford, 1985], [Clifford & Croker, 1987], [Clifford, 1987])

4.7 The ERAE model

The Entity, Relationship, Attribute, Event (ERAE) model ([Dubois, 1986;1987], [Hagelstein, 1988]) has been developed as part of the ESPRIT project METEOR. It is an extension of the E-R
model, including a number of additional features such as the possibility to describe individual objects, as well as groups of objects, to allow simultaneous events, to explicitly refer to past or future states of the system, etc.

In particular, the basic components of the model are objects and associations. An object may be an entity, a value or an event whereas an association may be a relationship or an attributeship. Time is introduced in the model as a distinguished value type. All these concepts are assumed to be handled in the framework of a typed first-order predicate logic with equality.

An event is defined as an instantaneous happening of interest. Each event has associated with it a time of occurrence but its properties are time independent. An entity can change over time. In fact, with each entity they associate one time period during which the entity is of interest to the enterprise. Thus, time periods are assumed to exist in the model. Moreover, because each entity is associated with just one time interval, the model does not support "multiple incarnations" of entities. For example, if an employee is fired, there is no way to represent the fact that he is rehired unless we create another distinct entity with the same properties but different existence period or we assume that the existence period is not just one time interval but a set of time intervals. The first solution is followed under the assumptions of ERAE while the second is explicitly dropped out.

A relationship or attributeship can be permanent or time-varying. The relationships between entities and attributes of entities may depend on time, while relations and attributes involving events may not. This is because events are assumed to be instantaneous and thus it is likely that their properties do not change. Also, for every relation among entities, an axiom is added, stating that the corresponding fact may only be true iff the corresponding entities exist. This is not required for events.

ERAE is directed towards instances. However groups of entities can be declared. These groups are not necessarily disjoint and in the course of time an entity may move from one group to another. This can provide us with a solution to the "multiple incarnations" problem discussed earlier. That is, we can create a group of fired employees so when the employee is fired we move him in the fired employee group and when he is rehired we move him again in the employees group.

Another interesting point addressed in ERAE is the expression of dynamic constraints i.e., restrictions on the evolution of the world in the course of time. These are concerned either with state transitions or with events or with connections between events and states. In order to express
constraints on the possible sequences of states (i.e., state transitions), they employ a set of temporal operators [Dubois, 1988] which are referring to the past or future. Also, they extend these operators by a duration component in order to permit the expression of real-time properties. Constraints on events i.e., ordering, simultaneity, causality etc. are directly expressible in the framework while for the interaction of state components and events they define the predicate \textit{occurs} which is true in a state for an event if that event took place between the previous state and the current one.

A specification written in ERAE consists of two parts. The first is the \textbf{declaration part} where concepts are identified and classified into the various categories described earlier. The second one is the \textbf{statement part} where the concepts defined in the declaration part are further constrained using a typed first order language with equality.

For our retail company example, the declaration part of the specification is shown in figure 8. In this figure, the entity groups appear as squared boxes, relations as arrows, attributes as lines and events as circles. The small squared boxes inside the entity groups represent entity instances and all the dashed lines represent time varying associations.

![Diagram of the declaration part of the specification in ERAE](image-url)
The Time Dimension in Conceptual Modelling

The statement part of the specification consists of all kinds of constraints that we are able to express. However, some kinds of constraints that are frequent can also be expressed in graphical notation e.g., cardinality constraints for relations and attributes (not shown in figure 8). Our example constraint can be specified in the ERAE language as follows.

\[
\begin{align*}
\text{ord : Order; prod : Product; cust : Customer; t,t': Time; bal : Balance; qt : Qty; pr : Price;}
\end{align*}
\]

\[
\begin{align*}
\text{is_in(ord, Order, t) \land refers_to(ord, prod, t) \land made_by(ord, cust, t) \land (t \geq t')} \\
\land \text{has(cust, bal, t) \land has(ord, qt) \land has(prod, pr, t) \land ((bal + qt \times pr) \geq \£5,000)} \\
\Rightarrow \text{is_in(backorder, ord)}
\end{align*}
\]

Concluding, in ERAE time is considered to consist of a linear sequence of states with a set of events labelling the transitions between states. Each state is a configuration of existing entities, relations between them, and attributes characterising them and each state is associated with a time value (i.e., time point), which increases along the sequence.

The temporal reasoning is based on a set of functions and predicates. Among them, there is a predicate \textit{exists} which takes an entity and a time as arguments and returns a boolean value indicating whether the entity is in existence at that time. \textit{Exists} is true for each entity in some set of consecutive states. It reflects the intuitive fact that entities exist for some time. Also, there is a function \textit{time} which when applied to an event gives the time value associated to the state where \textit{occurs} is true for that event. Whereas \textit{time(event)} is state-independent, the function \textit{time} without arguments is state-dependent and its value in a state is the time value labelling that state. In addition, there are functions 'day' and 'hour' which convert values of type Time into values of type Day or Hour and also the ordering predicate (<) and the equality predicate (=) for time points.

4.8 The Conceptual Modelling Language

The Conceptual Modelling Language (CML) [LOKI, 1986; CML, 1987] has been developed as part of the ESPRIT project LOKI for modelling the domain of discourse and requirements of information systems. CML has an object oriented framework with many improvements such as the exception handling mechanism and the integration of time. The object oriented framework adopted by CML is based on semantic network ideas and has evolved out of research done for the TAXIS [Greenspan, 1983] and RML [Greenspan, 1984] languages.

Time in CML is considered to be discrete. The representation of time follows the formal framework presented in [Allen, 1983]. According to this, time is represented using time intervals. The temporal relations among them are axiomatised using the primitive predicate \textit{meets} [Allen,
The Time Dimension in Conceptual Modelling

1987]. The full list of temporal relations offered in CML is: equals, during/over, before/after, startsbefore,startsafter, endbefore/endsafter, meets, overlaps, costarts and coends.

Every CML object (i.e., instance-object, isa-object, property-object) has a temporal component attached to it by default. This is a time interval which is called SymbolInterval and is denoted by a symbol sign (e.g., TI1). Any CML declaration contains two optional time parameters that restrict the corresponding symbol intervals. The first of these parameters is used to record the system's knowledge about the inserted CML objects and is called model time, while the second is used to record the system's knowledge about its own knowledge or the system's belief about its knowledge and is called system time. Also, these parameters have the format $R\;ti$ where $R$ is a temporal relation and $ti$ is a ConventionalInterval i.e., a time interval expressed in absolute time values (e.g., date).

In CML there are also two special time constants called Alltime and Now. The first is assumed to be the largest finite time interval which by default is over all the others and the second is assumed to represent always the current system time and is used to axiomatise the open ended intervals (ElasticIntervals). All the time intervals and their relationships are represented by a graph where the intervals are nodes and the relations edges.

The temporal reasoning in CML is done by the temporal reasoner. This provides the following services:

- extracts and stores temporal knowledge from the user's assertions, as well as computes and stores the combined knowledge of the existing and the incoming temporal information, and

- answers temporal queries of the form "ti1 $R$ ti2", where ti1,ti2 are time interval identifiers and $R$ is any temporal relation.

The reasoning algorithm used is based on the one found in [Villain, 1986]. In this paper, it is shown that the reasoning based only on intervals [Allen, 1983] is NP-hard problem. Instead, the time point based algorithm that they propose is tractable and more specific its performance is proved to be $O(n^3)$, where $n$ is the number of the temporal relations. Moreover, it is proved to be sound and complete. In CML, they use a slight version of this algorithm.
In CML, any model of the UoD and the IS is visualised as an interplay between entities and activities. Entities can have attributes and they can relate to each other. On the other side, classes of activities are intended to capture information about events.

For our retail company example, we might model the UoD as:

entity classes : Product, Customer, Order, Backorder,...etc.

activity classes : Delivery, Shipment, Order-Arrival, Price-Change,...etc.

The exact definition of the entity class Order and the activity class Delivery is shown in figure 9.

*IndividualClass* Delivery in ActivityClass

with

input
StartDelivery : Signal
control
output
DeliveryDocument : Document
constraints
NotEnoughStock :
OrderToBeDelivered.withQty  > OrderToBeDelivered.ForProduct.qoh
UnacceptableBalance :
( OrderToBeDelivered.withQty ) * (OrderToBeDelivered.ForProduct.price)
+ ( OrderToBeDelivered.FromCustomer.balance) > 5,000 {pounds}.

end.

*IndividualClass* Product in EntityClass

with

unique
pno : Integer
attribute
price : Real {pounds}
qoh : Integer

end.

Figure 9. Definitions for our example in CML.

To understand the definitions of figure 9, we provide the generic format by which each CML object is defined.

ω α in β [θ t1 -believed θ t2]
isa γ [θ t1 -believed θ t2]
with δ
ε : ζ [θ t1 -believed θ t2]

end α.

where
The Time Dimension in Conceptual Modelling

\( \omega \) is one of the omega classes i.e., IndividualClass, AttributeClass and Individual,
\( \alpha \) is the name of the CML object been defined,
\( \beta \) is the user-defined or predefined class name of which \( \alpha \) is an instance,
\( \gamma \) is its superclass,
\( \delta \) is the name of a property category (i.e., attribute class), e.g., unique, necessary... etc.,
\( \epsilon \) is an attribute name
\( \zeta \) is an attribute value type or an attribute value or an assertion which has the character of an integrity constraint or a deductive rule which computes values of the underlying 
\( \theta \) is any temporal relation i.e, during, overlaps,... etc.

4.9 The TEMPORA model

The TEMPORA model [Theodoulidis et al, 1990; Loucopoulos et al, 1990] has been developed as part of the ESPRIT project TEMPORA that explicitly recognises the need for information systems which provide a closer alignment between business policy and system operation c.f. [Fjeldstad et al, 1979; Balzer et al, 1983; Mathur, 1987; van Assche et al, 1988; Loucopoulos, 1989].

The central argument in support of such an approach is that the evolution of information systems is, to a large extent, due to changes in the business environment and therefore, substantial improvements can be made in the development and evolution of systems if the business knowledge is explicitly captured and represented. This implies that organisational policy must be separated from the procedure which supports that policy and so allow a greater distinction between the ‘what’ and the ‘how’ aspects of a system (thus conforming to the preliminary report by ISO [van Griethuysen et al, 1982]).

The TEMPORA model consists of two parts namely, the Entity Relationship Time (ERT) model and the Conceptual Rule Language (CRL). The ERT model is an extension of the E-R model including a number of additional features which are considered to be necessary for the modelling of contemporary information system applications. More specifically, it accommodates the explicit modelling of time, taxonomic hierarchies and complex objects. The basic components of the ERT model are entity classes (simple and complex), relationship classes and value classes (simple and complex) where entity classes and relationship classes can be permanent or time varying.

The distinction between simple and complex objects is that simple objects are irreducible in the sense that they cannot be decomposed into other objects and thus, they are capable of independent
existence whereas a complex object is composed of two or more objects and thus, its existence might depend on the existence of its component objects. The relationship between a complex object and its component objects is modelled through the use of the IS_PART_OF relationship.

Time is introduced in ERT as a distinguished class called time period class. More specifically, each time varying simple entity class or complex entity class and each time varying relationship class is timestamped with a time period class. A time period is assigned to every time varying piece of information that exists in an ERT schema. The term time varying refers to pieces of information that the modeller wants to keep track of their evolution i.e. to keep their history and consequently, to be able to reason about them. For example, for each simple entity class or complex entity class, a time period might be associated which represents the period of time during which an entity is modelled. This is referred to as the existence period of an entity. The same argument applies also to relationships i.e., each time varying relationship might be associated with a time period which represents the period during which the relationship is valid. This is referred to as the validity period of a relationship. As a consequence of the adopted timestamping semantics, only the event time is modelled in ERT; i.e. the time that a particular piece of information models reality.

Besides the objects for which history is kept, another type of object, called event is also supported. These are objects that prevail for only one time unit and thus, the notion of history does not apply to them. Alternatively, one can say that these objects become history as soon as they occur. Events are denoted by defining the duration of their timestamp to be one time unit.

The ERT diagram for our retail company example is shown in figure 10. As shown in this figure, entities are represented as squared boxes, relationships as filled squares and values as squared boxes with a black triangle on the right hand side corner. Entity classes and relationship classes can be timevarying and this is indicated as an added square with the letter T standing for the timestamp. All relationships are named in two directions and have cardinality constraints attached. In this example, there was no need to model complex objects. However, the value class address could be easily modelled as a complex value class with components the street name, street number, post code etc.
Modelling of business policy and rules is carried out in terms of the Conceptual Rule Language (CRL) which is concerned with constraints placed upon the elements of ERT, with the derivation of new information based on existing information and with the invocation of procedures that denote execution semantics of transactions. Within the CRL formalism, different rule types are distinguished in order to achieve orthogonality of concepts and more assistance in the rule elicitation process. In addition, a textual layout with natural language semantics was adopted for the CRL language in order to increase its understandability and usability. More specifically, the following different types of rules are distinguished:
The Time Dimension in Conceptual Modelling

36

- **Constraint rules** which are concerned with the integrity of the ERT components. They are further subdivided to static constraint rules which are expressions that must hold in every valid state of an ERT database and transition constraint rules which are expressions that define valid state transitions in an ERT database. An example of an static constraint rule might be 'The number of employees working in a department must be less than 100 at all times'. An example of a transition constraint rule might be 'The salary of an employee must never decrease'.

- **Derivation rules** which are expressions that define the derived components of the ERT model in terms of other ERT components including derived components. There must be exactly one derivation rule for each such component. As the constraint rules, derivations rules are also subdivided to static derivation rules and transition derivation rules depending on whether the derived ERT component is timestamped or not. An example of a static derivation rule might be 'A supplier is the cheapest supplier for a particular product if his offer for this product has the minimum price'. An example of a transition derivation rule might be 'A customer is the best customer of this month if the total amount of his orders placed this month is the maximum'.

- **Action rules** which are concerned with the invocation of procedures. In particular, action rules express the conditions under which procedures are considered fireable i.e., a set of triggering conditions and/or a set of preconditions that must be satisfied prior to their execution. An example of an action rule might be 'When the stock of a product falls below the reorder quantity level specified for this product then execute the reorder procedure immediately'.

The CRL language provides a very flexible and effective way to express the rules of your application. In addition, for any rule in the application there might be more than one ways to express it depending on the design choices made. For instance, the rule of our example can be expressed either as a constraint rule or as an action rule depending on the ERT diagram of the application. For the ERT diagram of figure 10, the rule can be expressed as follows:

```
WHEN order_arrival
  IF CUSTOMER.X makes ORDER.Y and has balance =< 5,000
  THEN deliver_order(Y)
```
This action rule states that delivery can take place only if the balance of the customer is less than £5,000. In order to cater for the case where his balance is more than this amount another action rule should be introduced stating that his order will be treated as backorder when the balance of the customer is more than £5,000. This rule can be expressed as follows:

\[
\text{WHEN order_arrival} \\
\text{IF CUSTOMER.X has balance > 5,000} \\
\text{THEN create_backorder(Y)}
\]

Concluding, the formal framework employed in TEMPORA as the temporal reasoning mechanism is that of Interval Calculus proposed in [Allen, 1983] with the addition of a formal calendar system in order to provide for the modelling and reasoning about the usual calendar periods.

Besides the possibility to express relative and absolute time, other notions of time such as duration and periodic time are also represented directly in TEMPORA. Furthermore, besides the usual time units defined in the calendar system, the TEMPORA formalism includes a set of functions and predicates that allow for the definition of application-dependent calendar units.

5. Comparison of Models

To date many approaches for the explicit treatment of time in a conceptual modelling formalism have been developed. The most important of these have been presented in the previous section. A key issue that has to be clarified in full from the onset, is the nature of time in the conceptual modelling approach. In general, time has itself an ontology. However in a computing environment, such a wide-ranging ontology is not enough. We want to be able to represent and reason about time varying information in a specific application context. That is, we must find out what should be the expressive power of the temporal formalism that we are going to adopt. For example, when we are interested in prediction systems we should consider alternative futures. In the context of such an application domain this becomes a necessary assumption.

In this section, three comparison matrices are given which provide an overall view of the presented approaches. Together, they cover the time semantics, the model semantics and the temporal functionalities of each approach. The reason for adopting this view is to provide a unified comparison framework for the treatment of time together with the primitive notions and the inference method of the underlying model.
The semantics of time used in the approaches discussed in this report, vary according to the designers’ choice. The available choices which are shown in figure 11, are:

**Time elements:** This issue refers to whether time is modelled as discrete elements (such as integers) or as dense elements (such as rationals or reals) and also, whether time data values are associated with points or intervals. The last choice is not applicable when using discrete time, since the two representation schemes are equivalent.

<table>
<thead>
<tr>
<th>Time Elements</th>
<th>Absolute/Relative</th>
<th>Linear/Non linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infological Data Model</td>
<td>time points isomorphic to the set of reals</td>
<td>absolute</td>
</tr>
<tr>
<td>CIM</td>
<td>timepoints isomorphic to the set of reals and time period types</td>
<td>both</td>
</tr>
<tr>
<td>TERM</td>
<td>time points isomorphic to the set of reals</td>
<td>absolute</td>
</tr>
<tr>
<td>Lundberg's Logic-Based Model</td>
<td>time points isomorphic to the set of integers</td>
<td>both</td>
</tr>
<tr>
<td>INFOLOG</td>
<td>time points isomorphic to the set of reals</td>
<td>relative</td>
</tr>
<tr>
<td>HDBM</td>
<td>time points isomorphic to the set of reals</td>
<td>absolute</td>
</tr>
<tr>
<td>ERAE</td>
<td>time points isomorphic to the set of reals</td>
<td>both</td>
</tr>
<tr>
<td>CML</td>
<td>discrete time periods</td>
<td>both</td>
</tr>
<tr>
<td>TEMPORA</td>
<td>discrete time periods</td>
<td>both</td>
</tr>
</tbody>
</table>

Figure 11. Classification of models in terms of time semantics
The Time Dimension in Conceptual Modelling

39

**Absolute/relative time:** This issue refers to whether exact time points or intervals are associated with each time data value or whether relative time expressions are supported in order to increase the expressive power of the approach.

**Linearity of time axis:** This issue refers to whether the time axis can be always considered as linear or non-linear. The non-linearity of the time axis deals with aspects of time such as periodic time and branching time.

From figure 11, it can be seen that most of the approaches assume that time is represented by distinct "points" and that the time axis is essentially isomorphic to the real numbers line. This choice however brings together some difficulties concerning the implementation issue.

While it is correct according to Newtonian physics to assume that time is continuous, this has undesired implications when we try to map it to finite state machines like computer systems. This has been recognised in some of the approaches discussed in section 4, namely CIM, TERM and HDBM. In particular, it has been argued that since the UoD of the information system behaves discretely, we assume that time has also such an ontology. So, although ontologically time is considered to be isomorphic to the real numbers, for their purposes some approaches assume that time is discrete in order to avoid implementation problems.

The reasons can be summarised in the following (see also section 4.2):

- references concerning time are in information systems usually made in discrete terms, e.g., hirings occur daily
- when references to locations in time are made, their representation must be finite, e.g., in a computer system or on a piece of paper
- from a modelling point of view time intervals may be considered to be point like (in discrete terms).

A related issue concerning the time elements is whether we consider time to be representable using points or intervals. While these two representation schemes are equivalent in discrete terms there are some pragmatic differences. Most of the approaches examined consider time points as the basic time elements based on the fact that physics and science in general do so. However, it is very common to refer to time intervals in our everyday life and thus, duration-based representation should also be included in the basic time elements.
In addition, there are advocates of the use of time intervals instead of time points as the basic unit of time representation. The justification provided by them is that the interval based timestamping is more suitable because:

- It allows imprecision and uncertainty of time information
- It allows to vary the grain of reasoning
- It can be comprehended more easily by humans

The absolute/relative dichotomy has its roots in a long standing philosophical controversy. The view of time as an absolute phenomenon means that time is independent, having an existence unrelated to any events or entities with which it may be associated. Relative time refers to the view of time as a set of timing relationships such as before or after between events. The relativistic view offers no ontology of time as such but has the advantage that timing relationships are directly representable. The fact that some approaches have defined both views means that the need to accommodate relative time as well as absolute has been recognised.

Most of the approaches assume that the time axis is linear. While this seems to be the case for the time requirements in most information system applications, there are some cases where the nonlinearity of the time axis is needed. These can be found in areas which have the need to reason in alternative futures such as decision support systems, job scheduling systems and forecasting models. Clearly, the choice is context-depended and needs to be stated explicitly.

The second comparison dimension deals with the temporal functionalities of each approach i.e., what aspects of the interaction between time and information it subsumes and the formal framework that it adopts. A summary of the choices for each approach is shown in figures 12 and 13. Note that in these figures we did not include all the aspects discussed in section 2.2. The comparison is made according to the following characteristics:

**Primitive notions:** These are the basic elements that each model describes and they form the basis for specifying characteristics of the application. Most of the models discussed support a notion of event which is the basic element for specifying that something has happened in the system. The other notions reflect the particular choice on how we best model reality.
**Primitive temporal notions:** These are the basic units used for temporal inferencing. Note that these are different from the time elements. The latter refers to the ontology of time. Here the choices vary according to what is considered as the most appropriate temporal notion for reasoning about time. Two orientations exist. The first is based on the notion of state that allows the system to prove properties which are true in a given state and the second is based on the notion of time points or time periods which are used as a reference to describe the basic operations performed when making deductions.

**General inference method:** This refers to the formalism used to make deductions which are not dependent on time. First Order Logic (FOL) is used in general for reasoning and it is usually restricted in order to make it computable.

**Temporal reasoning formalism:** This refers to the formalism used to deduce temporal information out of a set of facts. All the models except the Infological Model, use some sort of logic extended to manage temporal data; based on the FOL or the temporal logic (TL) or the tuple relational calculus (TRC). Here computational complexity issues should be dealt with together with the incorporation of any optimisation strategies.
### Monotonic/nonmonotonic

This refers to a specific assumption that is made in a model concerning the retroactive changes. That is, a system is said to be nonmonotonic if an update to a set of facts already stored in the system may cause changes to some conclusions previously drawn. Most of the approaches assume nonmonotonicity.

### Temporal metrics

This issue refers to the need of defining temporal data at various levels of granularity and of performing reasoning involving time i.e., about dates and durations. Some of the approaches support temporal metrics that are user defined as in TERM or predefined as in CIM. At least a simple time metrics should be supported in any modelling formalism.
Persistence assumption: This assumption relates to the storage requirements of the modelling paradigm. If a formalism requires all its objects to be time-referenced and this must be reflected in the storage structure, we say that the formalism conforms to the persistence assumption.

Incomplete temporal information: This refers to the degree of imprecision that the model allows in the specification of temporal data. It is clearly a desirable feature which however, is supported by few approaches. Some provision for incomplete information should be given in any modelling formalism.
Event/transaction time: Both time dimensions are necessary in any modelling formalism [Abbod et al, 1987]. Event time is essential for keeping track of past data and for considerations about the future whereas transaction time helps to make a distinction between what happens in the external world and what is stored in the system.

Some other aspects of the interaction of time with information were omitted from this comparison either because all the approaches take them into account by adopting a uniform format or because they are not relevant to this paper. For example, all of the approaches recognise the need for different property types and also, all the approaches assume implicitly or not that they deal only with homogeneous processes. In addition, the issue of unique object identification across time and the provision of multiple incarnations of the same object should be examined in the implementation context. The aspect of evolving schemas is not considered in any approach.

6. Conclusions

The need for the explicit treatment of time in conceptual modelling in order to enrich the captured information with more real-world semantics, is well understood by the majority of researchers. This is self evident from the huge amount of publications which in one way or another deal with this subject. The purpose of this paper is to provide a detailed overview of the issues resulting from the incorporation of the time dimension in a conceptual modelling formalism and in addition, to provide a framework for comparing the different approaches irrespective of their origins.

To this end, a number of ontological assumptions about the nature of time and its interaction with other pieces of information were identified and these provided the basis over which a throughout analysis of some of the most important and diverse approaches to treat time was carried out. This discussion clearly indicated that the choice of a time model is restricted by the semantics of a given application domain and as a result, a common reference framework for describing the treatement of time is necessary.

Consequently, a comparison framework was established which uses criteria in two dimensions. The first one deals with the nature of time itself and includes the basic features that are used to describe it i.e., time elements, absolute/relative time and linearity of time axis. The second dimension deals with the temporal functionalities of each approach i.e., what aspects of the interaction between time and information it subsumes and the formal framework that it adopts. This dimension includes the primitive notions of the model together with its general inference mechanism, the primitive temporal notions and the corresponding temporal reasoning mechanism.
and also, issues concerning the interaction of time with the other informal conceptual models. However, research is still needed to provide temporal systems with a complete set of features to manage all the issues of real-time temporal data.

References


The Time Dimension in Conceptual Modelling


The Time Dimension in Conceptual Modelling


LOKI-86 A Logic Oriented Approach to Knowledge and Databases Supporting Natural Language User Interfaces Esprit Project 107 (LOKI), Institute of Computer Science, Research Center of Crete, Greece, March 1986.


The Time Dimension in Conceptual Modelling

48


