TIME MODELING IN OFFICE INFORMATION SYSTEMS

Barbic, F.[*] and Pernici, B[**]

[*] Dipartimento di Elettronica
Politecnico di Milano

[**] CSISEI - CNR
Politecnico di Milano

Abstract

Time management is a feature essential to office information systems. In the OIS environment, in addition to a static definition of time attributes attached to data, a more complete definition is needed, to handle both static and transition times and representation of temporal relationships The TSOS model, an extension of the SOS model for office description, contains a time model suitable for both of these requirements. In addition, the TSOS time model allows the representation of calendar times at different levels of abstraction and contains a precise definition of temporal and logical functions on time. Another main goal of the TSOS time model is to allow a flexible representation of temporal conditions for triggering and control rules in the office environment, so that an intelligent rule invocation is possible.

1. INTRODUCTION

Office Information Systems (OIS) are a class of Information Systems where some features are novel or more critical than in traditional information systems. Among these aspects, in this paper we address the problem of time modeling and management. Several models for time handling have been proposed in the literature, in different computer science research areas [Bol 82] As it will be illustrated in Section 2., each of these models takes into account somewhat different aspects in time management.

The purpose of this paper is to define the TSOS (Temporal Semantic Office System) time model to specify rule invocation conditions of control rules, and to present a formal definition of temporal functions in the presence of time abstractions.

TSOS is an extension of the Semantic Office System (SOS) for office systems description [Bra 84] In TSOS it is possible to handle temporal attributes of data and to verify static and dynamic constraints on them.

Functions are provided for handling relationships between times, as it is necessary, for instance, in planning systems, where causality relationships between times are more important that the corresponding precise times. For instance, the time of response to a letter may not be relevant as a precise time, while it is relevant in relation to the arrival time of the letter. Different levels of temporal specifications and imprecisely specified times are supported. For instance, it can be meaningful to specify monthly forecasts in planning activities, while a date and hour specification is more appropriate to organize a meeting. Finally, in TSOS a global control on several integrated office activities can be performed. Temporal information is used, besides for retrieval purposes, for activating system control rules based on tiem. Simple cases of this kind of specification are that of performing periodical checks in the system or that of triggering activities at a given time.

In Section 2., the existing time models and the problems related to time managing in different research areas are briefly reviewed. In Section 3. the TSOS architecture will be illustrated, with particular reference to the use of time. In Section 4. the TSOS time model is presented, several temporal categories and abstractions are presented, and their operators formally defined, the syntax and the semantics of rules conditions are presented.

Examples to illustrate an intelligent rule invocation mechanism and other relevant features of TSOS temporal conditions will be presented in Section 5..

2. RELATED WORK AND PROBLEMS

A particular attention has been given to time related problems in Information Systems in recent times [Pan 83].

A basic classification of models of time consists in the subdivision of models into static and dynamic models.

A similar classification has already been proposed in the literature, even if from slightly different points of view [Kun 84, Bol 82].

We classify as static time models those models in which states are the most important concept in the system [And 82, Lun 82]. This view is typical of database systems, where the current view of data is often the only one which is available. To consider time related aspects, either a time attribute (timestamp) is attached to the attributes for which it is important to keep a temporal record, or several snapshots of the data are taken and temporal considerations are done on the appropriate subset of snapshots.
In this kind of models, the temporal operations on data are querying about temporal sequences of data, with simple deductive capabilities, and verification of static (integrity) constraints on data.

In dynamic models, the time considerations focus on transitions between states. In these models the transition time is important and the preconditions and postconditions for transitions are specified.
This view is typical of IS models for modeling relationships between activities. In this case dynamic constraints on state changes are specified, for instance, as preconditions and postconditions on update transactions [Cas 78, Cas 82].
The dynamic time modeling kind of approach is also used in artificial intelligence, in systems for planning activities to achieve a given goal or scheduling resource allocation [All 83a, All 83b, Coo 83, McD 82, Ver 83]. In these cases duration of activities and precedence constraints among activities are specified as constraints in the plan to achieve.

Both static and dynamic models can express temporal data using different modalities. The simple association of time to a given data element or event can have different meanings for instance, a time attribute can specify that the data or event happened once during that time, or lasted all the time, or most of the time of the time. A formalization of some of these problems has been given in temporal logics [Res 71, Gol 83], while other problems have been mainly studied in artificial intelligence, linguistics, and philosophy.

An aspect that is seldom considered is that times can be expressed at several levels of detail, like, for instance, dates versus minutes. A formalization of different levels of detail can be found in static time models, but it is hardly found in dynamic models, where different levels of detail should coexist in a uniform framework.

The possibility of specifying imprecise temporal information is common to both types of models For instance, this is done in some dynamic models, in particular for planning, where times are indirectly specified through relationships between activities rather than absolute times such as dates [All 83a, All 83b]. Another case in which times are imprecise is that of distributed systems [Lam 78] In this case the ordering relationship on time is not always known, or it is only known with a certain approximation.

A few existing office models have already taken into consideration some time related problems.

In [Cha 82] the problem of inserting alerters to signal particular conditions in an office system, in particular on updates of values is examined. In this work it is possible to express some simple time based conditions, through a time attribute in the system, the time is considered to be a time point and no operation is defined on it, except for equality.

In OBE [Zlo 82] it is possible to express trigger expressions to be evaluated at a specified time. The time can either be a date and hour or a period (hourly, daily, weekly, monthly), but combinations of conditions are not allowed.

3. TIME MANAGEMENT IN SOS

The SOS model [Bra 84] allows the specification of types of elements in the office, static elements, such as documents, are specified in the static submodel, dynamic elements, such as activities, are specified in the dynamic submodel, a third submodel in SOS, the evolution submodel, allows the specification of control aspects in the office.

Control specifications in SOS are used to trigger activities, monitor abnormal situations, and support system use. A complete classification of control situations is given in [Bra 85].

Control specifications are given in SOS using control rules, composed by a conditional part and a body.

In the conditional part, the condition for rule invocation is specified, while the body specifies which is the action to be taken when the conditional part is verified.

Each SOS element has a type and a number of instances. Each instance is uniquely identified in the run-time system, e.g., several instances of the document type letter will be found in an office system.

In the same way, also for control rules several rule types can be defined and for each type several instances may exist. The rule type "when a letter arrives, send a message to the destination agent" can be instantiated by adding a particular letter id to the rule type specification, thus creating an instance.

In rules description, sometimes it is interesting to specify rules at type level, sometimes at instance level. In the above example, the type level of specification is preferable, and the related instances are automatically created by the system, one for each event of letter arrival. In other cases, it is useful for the user to directly specify an instance of a given rule type, for instance, if one wants to be reminded of a meeting, he can directly write a rule instance like the following one "at 3 p.m. tomorrow send me a message 'meeting in half an hour'".

The elements specified in control rules are those defined in the static and dynamic SOS submodels A query language to access the elements instances is defined [Bra 85].

In addition to usual conditions on element values, conditions based on time are considered in this paper.

The temporal extension of SOS, TSOS, is based on an IS type approach to time modeling, keeping in mind that the goal of temporal specifications in TSOS is, beyond querying the system about time related facts and static constraints expression, that of activating rules when their activation condition is verified. This means that we are interested in the times when transitions take place, rather than in the duration of system states. We assume to have a rule scheduling mechanism, for which each rule instance is invocated only once. If it has to be repeatedly invocated, it has to be created again. So, we are not interested in different time modalities, i.e., we consider 'only once' time modalities.

TSOS is based on SOS for data, activities, and rule specification structure, and has a formal model for time specification, that will be presented in Section 4.

Problems originated by use of abstractions and imprecise timing are dealt with, such as the meaning of operators and functions, and anomalies caused by time elements with a variable duration, such as months and leap years, are considered.

Two kinds of timing specifications will be handled in an homogeneous way absolutely and relatively defined times [Bol 83]. Absolutely defined times are time elements like, for instance, dates, relatively defined times are defined using, for instance, an event happening as a reference (e.g., "3 days after such and such event").

The temporal specifications defined in Section 4. are used not only for rules specification, but also in connection to all TSOS elements, where a temporal specification is needed.

A particular type of TSOS static element is the type event (and all its specializations). Events are office static elements types which can be instantiated like all other TSOS element types In particular, an identifier and a time specification (the happening time) are associated to each event which is registered as such in the system. The time specification can be either an absolute time or a relative time, or both. In particular, an event can have many relative time attributes that put it into relation to different other events. In this schema, it is not relevant whether those times are past or future, since we will assume a homogeneous description of times in the past and in the future. Each of these time specifications can be interpreted at a different level of abstraction in the model.

Other TSOS elements, like for instance document time-related attributes and starting and ending times of activities, can have a similar description of time attributes, in the framework of this time modeling.

In the next Section the TSOS model of time will be illustrated and in the Section 5. a few examples of how the TSOS model of time is used in rules will be presented.

4. TSOS TEMPORAL MODEL

4 1. The concept of time in TSOS

The temporal expressions we are going to consider are those useful in the office and the information systems environments.

TSOS has a few characteristics which are important in the office environment. First of all, the current time is a central concept for rule invocation, rules conditions are examined against the current time to decide if they can be invocated or not.

Second, it is possible to express the different temporal categories (time specifications, time points, intervals, periodic times) defined in the model at different levels of abstractions, using the common calendar schema for abstractions, for instance, a year is considered to be at a higher level than a month.

The general context of the model that we propose is based on a few assumptions. The time is defined on a temporal axis and is discrete, i.e., each point of the temporal axis can be mapped into an integer number. This integer number represents the (signed) number of temporal units that exist between the point of time and the origin of the temporal axis. The quantum of time is the minute, that is considered the most specific point of time. The time is infinite in the past and in the future, so for each point on the temporal axis , it is possible to find a past and a future point of time.

Considering points of time t on the temporal axis, the time is linear, i.e., it is connected ($\forall$ t1,t2 (t1<t2) V (t1=t2) V (t2<t1)) and the precedence relationship enjoys the transitivity property ($\forall$ t1,t2,t3 t1<t2 $\wedge$ t2<t3 --> t1<t3) [Res 71].

We define a Multiple Level Model (MLM) of time, where the different temporal levels in the time hierarchy are integrated in a single model (see Fig. 1).

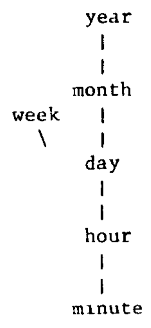

Fig. 1. Calendar model

Unfortunately, in this time model some variable transformations exist between the considered elements. For instance, months have variable durations expressed in days (28, 29, 30, 31 days for a month) This fact can hinder the precise conversion of the temporal elements from the month level to the day level, when they are not absolutely defined. In fact, once we know that the month we are considering is February 1984, i.e., we also know that its duration is 29 days and a conversion is possible This is not true if a reference to a fixed point of the temporal axis is not given For this reason, we express the relatively defined times without applying any conversion. A key issue in MLM is the possibility of expressing the same condition at different levels of detail, enabling the specification and the handling of conditions the elements of which are located at different levels of detail. The relationships among specifications at different levels of detail will be illustrated in the next Subsection. Within a single level of abstraction the equality and a precedence temporal relationships exist that enable the comparison of times.

The elements of the MLM can be specified to a certain extent, this means that the temporal elements do not need to be completely defined, but they can be undefined from a certain level downward. This approach allows a more flexible handling of temporal conditions and is suitable for their specification in office systems In fact, even if the quantum of time is the minute, we usually do not want to express every condition in terms of minutes. Moreover, sometimes it is actually necessary to express the condition at a higher level of detail because the knowledge about it is not complete or is not of interest at minute level.

The MLM is obtained integrating the submodels of the different levels of time definition. These submodels will be called "Single Level Submodels" (SLS).

A critical drawback of the use of a single SLS, such as, for instance, the minute submodel, is that it is extremely unflexible and only enables the specification of very rigid temporal conditions [Man 83]. The elements within a submodel are always completely defined (cd), i e., always specified until the adopted level of detail. For instance, if we consider time specifications in the minute SLS, the condition "after three weeks" is interpreted as "after the precise number of minutes corresponding to 3 weeks".

In SLSs the time is mainly represented in cardinal way, i e , a time point is identified by a (signed) integer that indicates the number of time points from the origin of the temporal axis Normally, instead, the temporal data are specified in ordinal way, i e , by referring them to the current time. For instance, "today is the 27th of November 1984" is normally (and ordinally) interpreted as the sentence "the current day is the 27th day of the 11th month of the 1984th year" In the day SLS instead, we (cardinally) specify "the current time is 1983 years, 10 months and 26 days far away from the origin of the temporal axis" The choice of representing time in cardinal way was suggested by implementation reasons (see the definition of time points in the next subsection). The resulting schema is shown in Fig. 2 in which the two representations are connected by ad hoc conversion functions.

$$\begin{tikzcd}
\text{USER} \arrow[d, shift right] \\
\text{EXTERNAL TIME MODEL (ORDINAL)} \arrow[u, shift right] \arrow[d, shift right, "tp"'] \\
\text{INTERNAL TIME MODEL (CARDINAL)} \arrow[u, shift right, "date"']
\end{tikzcd}$$

Fig. 2. The external and internal time models

MLM is defined by referring its basic concepts to the set of the SLS basic concepts. In other words, the definition of MLM is given in two steps

- Formal definition of a SLS. In next subsection, we will completely define the elements, relationships, functions, and operators of a SLS Each SLS in the MLM has the same characteristics from this point of view.
- Mapping of the MLM in terms of the involved SLSs. The methods for integrating different SLSs to get a MLM are described. To do this, we describe conversion functions, and discuss their applicability, and the meaning of relationships, functions, and operators between elements belonging to different SLSs. The basic concept is that in each MLM temporal specification, the correct SLS levels must be associated to each term.

4 2. SLS definition

In this subsection we will formally define elements, functions, relationships, and operators for a SLS.

We first define the adopted Temporal Categories (TC) that are the domains of the elements of all the SLSs. We introduce here the Time Specification TC (TSTC), the Time Point TC (TPTC), the Time Interval TC (TITC), and the Periodic Time TC (PTTC). These TC are represented in Fig. 3.

Time Specification Time Category (TSTC)

The TSTC is a temporal category formed by relative integers and is used to specify the distance of a point of time from a fixed reference, the duration of a time interval, and so on. We can define the equality relationship, the precedence relationship, the distance between ts, and the sum and

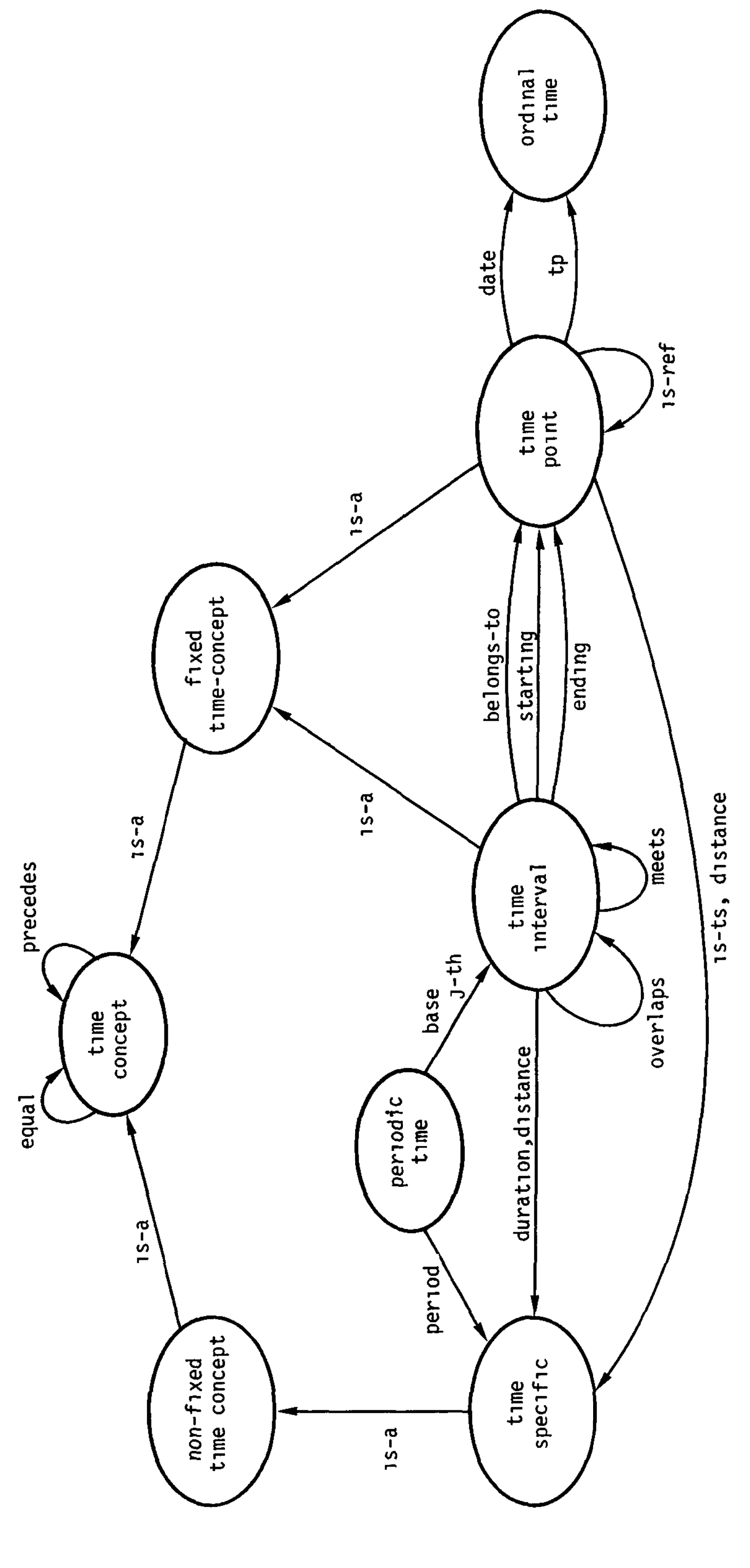

Fig. 3 Semantic network of a SLS

subtraction of ts in terms of the corresponding concepts of equality, comparison, absolute difference, sum, subtraction, that are valid in the field of the relative integers.

Time Points Time Category (TPTC)

The TPTC is the temporal category of time points (tp) This TC is based on the primitive concepts of current time (CT) and starting time (ST). CT takes into account the flowing of time, ST constitutes the origin of the temporal axis. All the other tp $\in$ TPTC are defined as pairs tp=(ts, ref), where ts $\in$ TSTC and ref is CT or ST. The functions **is_ts**(tp) and **is_ref**(tp) allow to address the components of a tp.

A tp can also refer to the occurrence time of an observable event or to another tp, tp1=(ts,tp2). A tp $\in$ TPTC is called absolutely defined (adtp) if it directly or indirectly refers to the ST Otherwise when the tp is defined only by putting it in relation with the occurrence time of a certain event (event_time) or by specifying a causality relationship (event1 time happens after event2 time), the tp is called relatively defined (rdtp).

The absolute definition is transitive, in fact a tp $\in$ TPTC defined relatively to an adtp can be interpreted as an adtp. A rdtp can possibly become adtp during the system evolution, while the opposite transition is not possible, in fact, if a rdtp refers to an observable event, when the event occurs, the rdtp becomes an adtp for transitivity, the opposite transition is obviously not possible. For instance, let us assume that at the beginning of a year it is not known when vacations will be taken One can specify conditions such as "A week before vacations". This is a rdtp. Once vacation dates are fixed, the specified time becomes an adtp.

Because of the possibility of handling both rdtp and adtp, all the relations (equality, precedence, distance etc.) can have a precise value (i.e., "TRUE" or "FALSE") or an undefined value (i.e., "UNKNOWN"). In general, when two tp $\in$ TPTC are compared, but they do not refer to the same tp, the comparison is all the same definite, but its result is "UNKNOWN". Notice that a tp can have different equivalent representations, if it can refer to different tp For that reason, we will assume that when this is possible, all the tp refer to the same tp (ST).

The following functions are defined on the elements of TPTC.

Equality

Given tp1, tp2 $\in$ TPTC, where tp1=(ts1,ref1), tp2=(ts2,ref2) the predicate (tp1=tp2) is TRUE iff (ts1=ts2) $\wedge$ (ref1=ref2), FALSE iff (ts2 $\neq$ ts1)$\wedge$ (ref1=ref2), UNKNOWN otherwise.

Proposition

Given tp1, tp2, ref1 $\in$ TPTC, where tp1=(ts1,ref1), tp2=(ts2,ref2) and ref1=(ts3, ref2), if ts1+ts3=ts2 then tp1=tp2.

Negation

Given a predicate p, the predicate **not** p is defined as follows

TRUE iff p is FALSE, FALSE iff p is TRUE, UNKNOWN otherwise, that is, unknown results remain unknown also in their negation.

An example of predicate negation is the following (negation of the result of an equality relation)

Unequality

given tp1, tp2 $\in$ TPTC, where tp1=(ts1,ref1), tp2=(ts2,ref2) the predicate (tp1 $\neq$ tp2) is TRUE iff (ts1 $\neq$ ts2) $\wedge$ (ref1=ref2), FALSE iff (ts2 = ts1) $\wedge$ (ref1=ref2), UNKNOWN otherwise.

Proposition

Given tp1 $\in$ TPTC, where tp1=(ts1,ref1), tp2=(ts2,ref2) and ref1=(ts3, ref2), then tp1 $\neq$ tp2 is TRUE if ts1+ts3 $\neq$ ts2, FALSE if ts1+ts3 = ts2, UNKNOWN otherwise.

Precedence

Given tp1, tp2 $\in$ TPTC, where tp1=(ts1,ref1), tp2=(ts2,ref2) the predicate (tp1<tp2) is TRUE iff (ts1<ts2) $\wedge$ (ref1=ref2), FALSE iff (ts2 $\geq$ ts1) $\wedge$ (ref1=ref2), UNKNOWN otherwise.

Distance d

given tp1, tp2 $\in$ TPTC where, tp1=(ts1,ref1), tp2=(ts2,ref2) let us define the function d TPTC X TPTC --> TSTC where

- d(tp1,tp2)=d(tp2,tp1) $\forall$tp1,tp2
- d(tp1,tp2) $\geq$ 0 $\forall$ tp1,tp2.

The distance between tp1 and tp2 is definite whenever tp1 and tp2 can refer to the same tp. The operational definition of distance is the following
given tp1=(ts1', ref3) and tp2=(ts2', ref3) then
d(tp1, tp2) $=_{def}$ abs (ts1' - ts2',)

Time Interval Time Category (TITC)

The TITC is a temporal category formed by pairs (starting tp, ending tp), where starting tp and ending tp are tp. The functions **starting_tp**(ti) and **ending_tp**(ti) allow to address the components of a ti. If the starting (ending) tp is the special tp $-\infty$ $(+\infty)$ [Gre 84], the time interval, ti is open to left (open to right).
The following functions are defined on the elements of TITC.

Duration

Given ti1 $\in$ TITC, where ti1=(tp1',tp1"), the **duration**(ti1) is by definition the distance between tp1' and tp1".

Belonging relationship

Given tp $\in$ TPTC and ti1 $\in$ TITC, where ti1=(tp1',tp1"), the predicate (tp $\in$ ti1) is TRUE iff (tp $\leq$ tp1") $\wedge$ (tp1' $\leq$ tp), FALSE iff (tp1"<tp) V (tp<tp1'), UNKNOWN otherwise.

Overlapping relationship

Given ti1, ti2 $\in$ TITC, where ti1=(tp1',tp1"), ti2=(tp2',tp2") the predicate (ti1 **overlaps** ti2) is TRUE iff $\exists$ tp'| (tp' $\in$ ti1) $\wedge$ (tp' $\in$ ti2), FALSE iff not $\exists$ tp'|(tp' $\in$ ti1) $\wedge$ (tp' $\in$ ti2),

operator	use	semantics
-	-ts1 -> ts2	$ts2 \mid ts1+ts2=0$
before	ts1 before tp1 -> tp2	$tp2 \mid d(tp1,tp2)=ts1 \quad tp2<tp1$
after	ts1 after tp1 -> tp2	$tp2 \mid d(tp1,tp2)=ts1 \quad tp1<tp2$
from	from tp1 -> ti1	$\forall tp2 \mid tp2 \in ti1 \quad tp1$ \leq $tp2$
until	until tp1 -> ti1	$\forall tp2 \mid tp2 \in ti1 \quad tp2$ \leq $tp1$
every	every (ts,ti) -> pt	$pt \mid pt = (ts,ti)$
i-th	j-th lpti -> ti_j	ti_j in $lpti=U_{j=1,n}\ ti_j$
last-but-i	last-but-j lpti -> ti_j	ti_{N-j} in $lpti=U_{j=1,N}\ ti_j$
duration	duration ti -> ts	$ts \mid d(starting(ti), ending(ti))=ts$
int	ti1 int ti2 -> ti3	$\forall tp \mid tp \in ti3,\ tp \in ti1 \wedge tp \in ti2$

a) operators

function	use	also applicable to
=	ts1=ts2	tp, ti, pt
<	ts1<ts2	tp, ti
∈	tp1 ∈ ti2	
overlaps	ti1 overlaps ti2	
meets	ti1 meets ti2	

b) functions

Fig. 4 Operators and functions

UNKNOWN otherwise.

Meeting

Given ti1, ti2 ∈ TITC, where ti1=(tp1′,tp1″), ti2=(tp2′,tp2″) the predicate (ti1 **meets** ti2) is TRUE iff tp1″=tp2′, FALSE iff tp1″ ≠ tp2′, UNKNOWN otherwise.

Equality

Given ti1, ti2 ∈ TITC, where ti1=(tp1′,tp1″), ti2=(tp2′,tp2″) the predicate (ti1 = ti2) is TRUE iff (tp1′=tp2′) ∧ (tp1″=tp2″), FALSE iff (tp1′ ≠ tp2′) ∨ (tp1″ ≠ tp2″), UNKNOWN otherwise.

Precedence

Given ti1, ti2 ∈ TITC, where ti1=(tp1′,tp1″), ti2=(tp2′,tp2″) the predicate (ti1 < ti2) is TRUE iff tp1″<tp2′, FALSE iff tp2′ ≤ tp1″, UNKNOWN otherwise.

Distance D

Given ti1, ti2 ∈ TITC, where ti1=(tp1′,tp1″), ti2=(tp2′,tp2″), let us define the function
D TITC X TITC --> TSTC
The function D(ti1,ti2) is by definition equal to d(tp1″,tp2′) if ti1 ≤ ti2 and d(tp2″, tp1′) if ti2 < ti1.

Periodic Times Time Category (PTTC)

The PTTC is the temporal category of periodic times (pt) formed by pairs (period, base), where period ∈ TSTC and base ∈ TPTC or base ∈ TITC.
The functions **period**(pt) and **base**(pt) allow to address the components of a pt ∈ PTTC.
Formally, pt=(ts, ti0) ∈ PTTC, with ti0=(tp′,tp″) is the series of ti
$U_{i=-\infty,+\infty}\ ti_i$ where
$ti_i=(tp_i', tp_i'')$ and $tp_i'=tp'+i*ts$ and $tp_i''=tp''+i*ts$.
The equality relationship is defined on the elements of PTTC.

Equality

Given pt1, pt2 ∈ PTTC, where pt1=(ts1,ti1), pt2=(ts2,tp2) the predicate (pt1=pt2) is TRUE iff (ts1=ts2) ∧ ($\exists\ ti1_j \in$ pt1, $ti2_k \in$ pt2| $ti1_j=ti2_k$), FALSE iff (ts1 ≠ ts2) ∨ (not $\exists\ ti1_j \in$ pt1, $ti2_k \in$ pt2| $ti1_j=ti2_k$), UNKNOWN otherwise.

A periodic time can be specified within a limited interval, a lpt is a limited periodic time, defined as a set of intervals $U_{i=1,N}\ ti_i$, this definition allows the definition of time functions of finite series of intervals, such as **i-th** (see Fig. 4).

A set of temporal operators and functions are also applied to temporal objects of different TC during conditions specification. By applying an operator, we get a temporal object, while by applying a temporal function we get a predicate that can assume a boolean value.
The basic and some derived operators and functions are listed in Fig. 4 where their use and semantics is shown It should be noted that the same operators and functions can be applied to objects of other temporal categories. Not all cases are shown in Fig 4 For instance, the every operator can also be used to define a series of periodic intervals ti of period ts, by specifying **every** (ts,ti).

4.3 The integrated Multiple Level Model

In MLM the SLSs at minute, hour, day, week, month, year levels are integrated in a unique model. It is possible to address in the same framework times defined at different levels.

A completely defined time in the MLM model has the form
YY MM WW DD HH MM

An important aspect of MLM is the necessity of intelligent conversion functions for mapping a SLS into another SLS in order to reduce the loss of information in passing to a level of abstraction to another.

The notation LEVEL (time) is used to indicate the more detailed time level used in that particular time specification, for instance , M (3) means "3 months". When a time is specified without an explicit level, the levels are counted starting from the higher year level as a default, for instance, the more detailed level in (1984 11) is the month level. When the **tp** function is used for conversions, the week level is skipped.

An important aspect of MLM is the necessity of intelligent conversion functions for mapping a SLS into another SLS in order to reduce the loss of information in passing to a level of abstraction to another.

When converting adtp, or time elements expressed in terms of adtp, such as, for instance, intervals, information about anomalies is known, so it is possible to handle conversions without any special problem. For instance, if "exactly one month after March 20, 1984" is specified, the duration of the month can be precisely expressed in days, since it is known that March has 31 days.
The formal specification is the following
D (D (M (1)) after tp (1984 03 20))

Instead, rdtp present some problems in passing from one level of detail to another. For instance, the condition "exactly one month after event time" cannot be precisely converted into days, since the generic month has a variable duration. The adopted choice is to express the points of time that cannot be fixed on the temporal axis without applying any conversion Once the condition becomes explicit, i.e., for instance, the event.time is specified, then the points of time are fixed on the temporal axis.

Imprecise times cannot be converted to lower level SLS times. In fact, it is not meaningful to express one month (approximate) in a number of days When imprecise times are involved, the conditional expression based on this times have to be converted to the corresponding SLS Obviously the temporal objects must be at least defined to the level required to the whole condition. This process is not always possible (for instance, when the single parts are specified in months and the whole condition is expressed in hours), this case is considered to be a wrong specification, and a diagnostic message is given. The conversion of SLS elements that are imprecisely defined introduces some errors.

The further problem is what kind of conversions is possible to apply when the one-level relationships and functions are applied to elements of SLSs of different levels.

The basic strategy is to convert the more detailed operator to a specification in the level of the other operator.

1984 11 = 1984 11 20
can be interpreted as
M (M (1984 11) = M (1984 11 20)) that is M (M (1984 11) = M (1984 11)) so it is TRUE at the month level, and UNKNOWN at the day (and all other lower) levels.

In case of temporal functions, each case must be considered separately, starting from the definitions. For instance
(1984 11 20 $\leq$ from 1984 11)
is TRUE in the interpretation (1984 11 20 $\leq$ from D (M (1984 11))), i.e., all the month, while it is UNKNOWN if M (1984 11) is an imprecise time.

We will assume as valid the second interpretation, since it is more general, to have the first case, it is necessary to explicitly specify the correspondent form given above.

4.4 Temporal conditions

The time model defined in TSOS allows to express temporal conditions in the temporal part of TSOS control rules.

The syntax of the language used for specifying temporal conditions is illustrated in Appendix I. This language must be considered an internal specification language easily interpreted by the system, rather than a language used for interaction with the user. The advantage of having such a language is that it is possible to express unambiguously complex temporal conditions.

The conditional part is a logical expression. Operands in the logical expressions are connected with AND, OR, and NOT connectives.

4 4.1 Operands

Operands in logical expressions are based on queries on data of the static and dynamic TSOS instances.

In addition to queries with boolean results on data on other domains, the data of TSOS instances can be in the temporal domain and in this case all the operations and functions defined in the TSOS time model can be applied For instance, a logical expression involving temporal data is the following

A1 = activity where activity.id=A12
A2 = activity where activity.id=A14

conditional expression

{A2.ending_time < A1 starting_time}

meaning "if the activity A1 started after that the activity A2 ended".

Other elements in logical expressions of the conditions are times defined within the TSOS time model, with no relation to data of TSOS instances (time points or intervals or periods, or any combinations of them through temporal operators and functions, as defined in Appendix I). In this case time elements in the condition are interpreted in the following way, with reference to the current time CT
CT $\leq$ time element

4.4.2 Logical connectives

The AND, OR, and NOT connectives can be used between operands. The truth table for rule invocation based on the conditional part is given in Table 1.

Tab. 1. Truth table for rule invocation

Lop Oper	Oper1	Oper2	Condition result	Invocation result
NOT	T		F	F
	F		T	T
	U		U	F
AND	T	T	T	T
		F	F	F
		U	U	F
	F	T	F	F
		F	F	F
		U	F	F
	U	T	U	F
		F	F	F
		U	U	F
OR	T	T	T	T
		F	T	T
		U	T	T
	F	T	T	T
		F	F	F
		U	U	F
	U	T	T	T
		F	U	F
		U	U	F

The AND connective requires that all the operands it connects are true, to yield a true result.

When times are connected by an AND, this is interpreted to be an intersection of the times, in case of the intervals and time points. A special case is that of periodic times, when two periodic times are connected by an AND their intersection is made, and the rule is invoked once in each of the resulting intervals.

The OR connective between temporal elements can only be an exclusive or, since it is not admissible to have ambiguity in rule invocation. For instance, if an activity has to be started at 3 or at 4 of a certain day, it must either be started at 3 or at 4, not at both times.

4 4 3. Rule invocation

Each rule instance is invocated once when the logical expression in the condition is verified. Rules with periodic times are considered as many instances of the rule. For instance, if an activity has to be done every hour, the rule expressing this condition can be considered as the set of rules for invocation of the activity, one every hour. Notice that only a finite number of instances can be handled in a physical machine, hence periodic times are always limited (lpt) in practice.

If the conditional expression is undefined because of the UNKNOWN result of some of its operations, then the rule instance is considered to be not invocable.

5. EXAMPLES

In this Section a few examples of use of the concepts defined in the TSOS model of time will be illustrated.

Example I

Let us give the following two rule instances

1. conditional part
   {tp (1984 11 30_15)}
   body·
   print document where id=D110 on printer where id=P02
2. conditional part
   {tp (1984 11 30_16 00)}
   body
   print document where id=D120 on printer where id=P02

The two rule instances require an activation of an activity on the same resource, so they cannot be invocated at the same time If a first found approach in rule invocation is found, then the first rule instance would be invocated first, but then it would be impossible to invocate the second rule. If an intelligent approach in rule invocation is taken, then the second rule is invocated first.

This example shows the advantage of specifying times at different levels of abstraction, another approach to solve the above mentioned problem would be that of specifying an interval from 16 00 to 16 59 for the first condition, however, this specification would not reflect the nature of the request, which has a vague character.

Example II

An example of a complex temporal condition is the following "Every Monday from 11 15 to 12 15 starting on November 1984 until June 1985".

Our time model allows the specification of such a condition, provided we define a periodical time for Mondays, to do this we can take any Monday, for instance Monday, December 17, 1984, from 11 15 to 12 15 as a base for the periodic time, and 7 days as the period, in addition, the conditions expressing that the period is limited from November 1984 to June 1985 have also to be expressed.

The conditional part of the rule can be expressed as follows

```
{every
   (1week'
    (from tp (1984 12 17_11 15) until
     tp (1984 12 17_12 15))))
 and
 (from tp (1984 11 01 until tp (1985 06 30))}
```

Example III

We present now an example where deduction capabilities can be applied to the time specifications.

Let us assume to have two activities, A1 and A2, and an activity starting time defined in an indirected way for A1.

A1 starting_time = form A2.ending_time
A2.ending_time = tp (1984 11 02)

Let us assume to have another activity A3, where

A3 starting_time = tp (1984 11 01)

The following condition

{A3 starting_time < A1.starting_time}

can be demonstrated to be TRUE, in fact, the interval in which A1.starting_time is located is starting at tp (1984 11 02), which is the ending_time of A2, and so it is after A3.starting_time Thus, it is possible to express coordination conditions between activities (and more generally, activity types).

Example IV

As a final example, let us see how a temporal specification in a natural language form can be ambiguous, if the semantics of operators is not precisely defined.

Let us consider the following conditions
"On October 12, 1984 and on November 1, 1984"
the following (erroneous) meaning could be associated to the expression "on both occasion activate the rule", while instead the rule is incorrect, because the condition Oct. 12 and Nov. 1 cannot be TRUE simultaneously. To correctly express the above condition in TSOS, it is necessary to specify two rules, one for the first invocation, and one for the second.

6. CONCLUDING REMARKS

In this paper the time model in the TSOS model for the description of office systems has been presented.

The main features of the TSOS time model are the precise definition of temporal objects, operators, and functions, the possibility of handling them at different levels of detail, and the possibility of expressing periodic times.

The time model in TSOS is used in temporal expressions in control rules for the office system environment.

An implementation of a scheduler for activities, based on temporal triggering conditions, has been made on a Vax/780, under Unix* in C language [Man 83].

Acknowledgements

The authors thank Ceri, S., Schreiber, F A, and Bracchi, G. for their comments and suggestions on this work and Manzi, G, Mulazzani, I, Oliari, G, Maiocchi, R., and Poggioli, M for implementing different ideas about time modeling in TSOS.

7. REFERENCES

[All 83a] Allen, J F and Koomen, J A, "Planning using a temporal world model", Int. Joint Conf. on Artificial Intelligence, pp. 741-747, Karlsruhe (1983)

[All 83b] Allen, J F, "Maintaining knowledge about temporal intervals", Comm of the ACM, vol. 26, n. 11, pp. 832-843 (Nov. 1983)

[And 82] Anderson, T L, "Modeling time at the conceptual level", in Improving Usability and Responsiveness, ed. Scheuermann (June 1982)

[Bol 82] Bolour, A, Anderson, T L., Dekeyser, L J, and Wong, H K T., "The role of time in information processing A survey", SIGART Newsletters, pp 28-48 (April 1982)

[Bol 83] Bolour, A and Dekeyser, L J, "Abstractions in temporal information", Information Systems, vol 8, n 1, pp. 41-49 (1983)

[Bra 84] Bracchi, G and Pernici, B, "SOS A conceptual model for office information systems", Data Base, vol 15, n 2, pp. 11-18 (Winter 1984)

[Bra 85] Bracchi, G and Pernici, B, "Specification of control aspects in office information systems", TFAIS '85, Barcelona, Spain (April 1985)

[Cas 78] Casanova, M A. and Furtado, A L, "On the description of database transition constraints using temporal languages", in Advances in Database Theory, ed Gallaire, Minker, and Nikolas, pp. 211-236, Plenum Press, New York and London (1978)

[Cas 82] Castilho, J M V, Casanova, M.A, and Furtado, A L, "A temporal framework for database specifications", Tech. Rep DB 038201 Dep. Informatica, Univ Catolica de Rio de Janeiro (March 1982)

[Cha 82] Chang, J M and Chang, S K, "Database alerting techniques for office activities management", IEEE Trans on Communications, n 1, pp 74-81, vol. COM-30 (Jan. 1982)

[Coo 83] Coolahan, J.F. and Roussopoulos N, "Timing requirements for time-driven systems using augmented Petri-Nets", IEEE Trans on Software Engineering, vol. SE-9, n. 5, pp. 603-616 (Sept. 1983)

[Gol 83] Golshani, F, Maibaum, T.S E, and Sadler, M R, "A modal system of algebras for database specification and query/update language support", Conf on Very Large Data Bases, pp. 331-339, Florence, Italy (Nov. 1983)

[Gre 84] Greenspan, S J, "Requirements modeling a knowledge representation approach to software requirements definition", Tech. Rep CSRG-155, Univ of Toronto (March 1984)

[Kun 84] Kung, C.H., "A temporal framework for database specification and verification", Conf on Very Large Data Bases (Aug. 1984)

[Lam 78] Lamport, L, "Time, clocks, and the ordering of events in a distributed system", Comm. of the ACM, vol. 21, n. 7, pp. 558-565 (July 1978)

[Lun 82] Lundberg, B., "An axiomatization of events", SYSLAB Report n 12, Univ. of Stockholm (Nov. 1982)

[Man 83] Manzi, G., Mulazzani, I., and Oliari, G., "Definizione e gestione delle regole temporali nel sistema informativo per l'ufficio SOS", Graduation Thesis, Politecnico di Milano, Dept of Electronics (1983)

[McD 82] McDermott, D, "A temporal logic for reasoning about processes and plans", Cognitive Science, vol. 6, pp 101-155 (1982)

[Pan 83] Panel, "Panel on time and databases", SIGMOD Database Week (May 1983)

[Res 71] Rescher, N and Urquart, A., Temporal Logic, Springer Verlag (1971)

[Ver 83] Vere, S A, "Planning in time windows and durations for activities and goals", IEEE Trans on Pattern Analysis and Machine Intelligence, vol. 5, n 3, pp. 246-266 (May 1983)

[Zlo 82] Zloof, M, "Office-by-example a business language that unifies data and word processing and electronic mail", IBM Systems Journal, vol. 21, n. 3, pp. 272-304 (1982)

* Unix is a trademark of AT&T Bell Lab

APPENDIX

SYNTAX FOR CONDITIONAL PART OF RULES

```
COND-EXPR --> '{' LOG-EXPR '}'
LOG-EXPR --> LOG-EXPR LOG-OP OPERAND | OPERAND | '(' LOG-EXPR ')'
LOG-OP --> NOT | AND | OR
OPERAND --> TIME-OPERAND | DATA-OPERAND
DATA-OPERAND --> DATA-VARIABLE | DATA-EXPR
DATA-EXPR --> '(' DATA-EXPR ')' | DATA-EXPR DATA-OP DATA-VAR |
              DATA-VAR
DATA-OP --> > | < | = | .  .
DATA-VAR --> variable | string
TIME-OPERAND --> TIME-VAR | TIME-EXPR | LEVEL '(' TIME-EXPR ')'
TIME-VAR --> tp | ti | pt | LEVEL '(' tp ')' | LEVEL '(' ti ')'|
             LEVEL '(' pt ')'
TIME-EXPR --> EQUAL-TE | PRECEDES-TE | BELONGS-TE | OVERLAPS-TE|
              MEETS-TE
EQUAL-TE --> TIME-SPEC '=' TIME-SPEC | TIME-INT '=' TIME-INT |
             TIME-POINT '=' TIME-POINT | TIME-PER '=' TIME-PER
PRECEDES-TE --> TIME-SPEC '<' TIME-SPEC | TIME-POINT '<' TIME-POINT |
                TIME-INT '<' TIME-INT
BELONGS-TE --> TIME-POINT ' ' TIME-INT
OVERLAPS-TE --> TIME-INT 'overlaps' TIME-INT
MEETS-TE --> TIME-INT 'meets' TIME-INT
TIME-SPEC --> '-' ts | ts | LEVEL '(' ts ')' | 'duration' TIME-INT |
              'distance' '(' TIME-POINT ',' TIME-POINT ')' |
              '(' TIME-SPEC ')' | 'is-ts' '(' TIME_POINT ')' |
              'period' '(' TIME-PER ')'
              TIME-SPEC + ts | TIME-SPEC - ts
TIME-POINT --> TIME-SPEC 'before' TIME-POINT |
               TIME-SPEC 'after' TIME-POINT |
               tp | LEVEL '(' tp ')' | 'is-ref' '(' TIME-POINT ')' |
               'starting' '(' TIME-INT ')' | 'ending' '(' TIME-INT ')'
TIME-INT -->  'from' TIME-POINT | 'until' TIME-POINT |
              number '-th' TIME-LPER | number '-th' TIME-PER |
              TIME-INT 'int' TIME-INT |
              'from' TIME-POINT 'until' TIME-POINT | ti |
              LEVEL '(' TIME-INT ')' | 'base' '(' TIME-PER ')'
TIME-PER -->  'every' '(' TIME-SPEC ',' TIME-INT ')' |
              TIME-PER | pt | LEVEL '(' pt ')'
TIME-LPER --> TIME-PER 'AND' TIME-INT
LEVEL --> Y | M | W | D | H | m
```