

EVENTS IN INTERACTIVE MULTIMEDIA APPLICATIONS: MODELING AND IMPLEMENTATION DESIGN

Michael Vazirgiannis

Computer Science Division
Dept. of Elec. And Comp. Engineering
National Tech. University of Athens
Zographou, Athens, 157 73, GREECE
e-mail: mvazirg@cs.ntua.gr¹

Susanne Boll

Database and Information Systems
Department, Computer Science Faculty,
University of Ulm
D-89069 Ulm, GERMANY
e-mail: boll@informatik.uni-ulm.de¹

December 10, 1996

Abstract

A variety of approaches and standards have been proposed to model multimedia presentations. The 'interactivity' of these models is often lacking or unsatisfactory. Interactions, however, form an essential part of an Interactive Multimedia Application (IMAP) and need to be reflected in the modeling and presentation of multimedia scenarios. In this paper, we introduce the notion of *events* as they are a promising approach for modeling the various interaction elements in IMAPs. A complex modeling of interaction elements must cover all possible inter-actions between a user and a computer as well as those inter-actions between entities within the computer system like the objects participating in the application, resources, devices, etc.. We elaborate on an event classification scheme and according to this, an object-oriented model for the different types of events in IMAPs. We also propose a rich composition scheme to allow for the definition of complex interactions. When presenting a multimedia scenario the occurring interactions must be registered and reacted to as fast as possible. Hence, we present the design of a MMDBMS client that manages IMAPs and implements the generation, detection and evaluation, and processing of such events in the context of the processing of IMAPs.

1. INTRODUCTION

To support complex Interactive Multimedia Applications (IMAPs), a system that offers both a suitable high-level modeling of IMAPs and interactive multimedia presentation capabilities is needed. The modeling should comprise the spatial and temporal composition of the participating media, the definition of interaction between the user and the IMAP, the specification of media synchronization and the presentation quality. An interactive multimedia presentation should cover the correct reproduction of the pre-orchestrated and interactive multimedia scenarios, synchronization enforcement, the observation of presentation quality as well as a fast response to user interactions. The trend in multimedia applications not only points the way to complex multimedia scenarios and to interactivity but also to multi-user

¹ Dr. Vazirgiannis and S.Boll have been working at GMD - IPSI, Dolivostr. 15, D-64293 Darmstadt, Germany, during the period substantial work for this paper was done.

environments. Therefore, the concepts needed to support IMAPs should be applicable not only to single-user but also to multi-user environments.

There are many approaches to model pre-orchestrated multimedia scenarios [SKD96, WRW95, WWR95, ISO93, LG93, ISO92]. These models proposed are mainly concerned with temporal aspects and synchronization. The „interactivity“ of these models, however, is often lacking or unsatisfactory. The modeling of interactions is more complex than it may appear on first sight and certainly goes beyond ‘button clicks’. In the multimedia literature interaction is hardly addressed as a research issue. Moreover, the few approaches of those mentioned above that take into account interaction [SKD96, WRW95, WWR95, ISO93] are very limited since they refer mostly to simple choices, jumps, selections, and temporal control actions (start, stop, pause, resume etc.) offered to a user by means of buttons, menus, and sliders.

We claim that modeling of IMAPs should put more emphasis to the interactive parts of such an application. In principle, the modeling of interaction should cover all the procedures that somehow involve the machine and a user. Such procedures, apart from button manipulations, should include scrolling actions, drag and drop actions, and temporal access control actions for multimedia objects. Moreover, interaction is not limited to the reactions between a user and the computer but can take place also between entities within the computer. For instance, objects that participate in an IMAP interact spatially and/or temporally with each other, e.g., if a button is dragged and moved over a certain area on the screen. Also, resources or devices produce messages/signals that may trigger actions in the context of the IMAP, e.g., if no audio device is available the equivalent text representation is displayed. When presenting multimedia scenarios, all occurring interactions must be sensed and reacted to adequately as fast as possible.

In this paper, we first introduce notion of *events* in the specific context of IMAPs as a means to represent the *happenings* that are of interest to an IMAP in section 2. We discuss the spatial, temporal and semantic aspects of events. As our main contribution, we propose a classification scheme for events and an object-oriented modeling in section 3. Moreover, we define a rich composition scheme for events, addressing algebraic, spatial, and temporal composition of complex events in section 4. The comprehensive modeling and composition of events serves as the basis for the definition of complex IMAPs. To illustrate the potential of our approach in the context of IMAPs, we present the modeling of events in the context of two complex sample IMAP scenarios in section 5. Our implementation approach, presented in section 6, integrates the event concept into an object-oriented multimedia database system. The client/server based architecture manages events in the context of IMAPs persistently on the server and provides the design of a multimedia database client that generates, detects and evaluates, and processes the events produced in the context of the execution of an IMAP. We conclude by summarizing our contributions and discussing further work to be done.

2. SPATIO-TEMPORAL ASPECTS OF IMAPS AND THE NOTION OF EVENTS

Operations on multimedia objects are merely based on the actions that modify the spatial or temporal dimensions of the objects. *Actions* describe parts of the temporal and spatial course of an IMAP in contrast to *interactions* that trigger the course of an IMAP. Actions can be either fundamental or composite. A fundamental action is, e.g., „start a video“, „present or hide an image“, etc. Composite actions are composed of atomic actions in the spatial and the temporal dimensions, according to a set of appropriate operators. For instance: "start video clip A, and 3 seconds after that, start audio clip B". In our approach, we make use of this action notion which is defined in more detail in [VTS96, VH93], in order to introduce complex interaction into the modeling of multimedia applications. We elaborate a related concept to describe the interactions that trigger the actions in the course of an IMAP.

One of the important aspects in the modeling of IMAPs is the spatio-temporal composition to relate multiple media in the temporal and spatial dimension. There are several approaches to model the temporal aspects of IMAPs [LG93, WR94, BZ93, Ho92], while the spatial aspects are rather under-addressed in the literature. Some of the interesting efforts in this area are [VTS96, IDG94].

2.1 Temporal aspects of IMAPs

To model the temporal aspects of an IMAP we consider a suitable representation of time. A unique point in time is denoted by a *temporal instance* which is of zero length. A set of temporal instances is completely ordered. For any two temporal instances one of the temporal relationships before ($<$), after ($>$), or equals ($=$) holds. This *point-based time model* is a simple representation of time with a small number of temporal relationships. A *temporal interval* is the difference between two temporal instances. In the *interval-based time model* for any two temporal intervals one out of 13 temporal relationships holds, as outlined in [Al83]. The point-based model and the interval-based model are equivalent in so far as each temporal interval $[a,b]$ is delimited by two temporal instances, namely the start point a , the end point b of the time interval for which the temporal relationship $a < b$ holds. E.g., an interval relation I_1 *before* I_2 with $I_1 = [t_{11}, t_{12}]$ and $I_2 = [t_{21}, t_{22}]$ can be described in the point-based model by the four temporal instances $t_{11}, t_{12}, t_{21}, t_{22}$ that delimit the two intervals and the three point relationships $t_{11} < t_{12}$ and $t_{21} < t_{22}$ and $t_{12} < t_{21}$.

To make the media objects participating in an IMAP scenario perceptible, they must be presented for a certain period of time, i.e., for a temporal interval. Therefore, the temporal course of a multimedia scenario corresponds to the appropriate temporal intervals, each representing the presentation duration of a single media object. Modeling of time with intervals, however, may not be sufficient when „something happens“ at an indefinite point in time during a presentation, e.g., to interactions. An interaction element such as a button might have a start point but no definite temporal end, as the presentation of the button ends with the selection of the button by a user. Therefore, we think that in the definition phase it is sufficient to identify only those temporal instances at which „something happens“ in an IMAP. For instance, the start of the presentation of a video, illustrated in Fig. 1, is identified by the temporal instance t_1 . This temporal instance is related to the *event* e_1 that indicates the starting point of the video presentation at t_1 . An event

such as e_1 is the occurrence of an action that can be recognized by a user or a process at t_1 . In the example in Fig. 1 the semantics of e_1 is: ‘Start of Video x’. The temporal instance of the end of the video presentation, however, can be uncertain, if it is defined to stop if a user presses a certain button. That is, event e_2 , ‘Video x stopped’ has no temporal instance assigned until the actual user interaction occurs at t_2 or the video presentation simply ends.

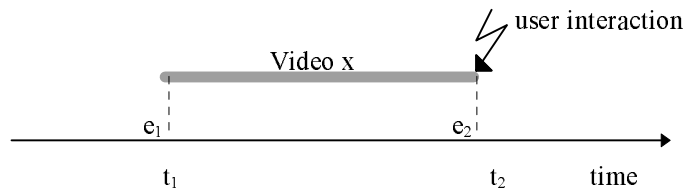


Fig. 1. The presentation interval of a video and the associated temporal instances and events

Although, in the context of an IMAP there may be a multitude of occurring events, we might only be interested in some of them. Therefore, we propose an event should be considered only if the IMAP cares about the specific event. The temporal duration of an event is zero, as it is related to a temporal instance.

2.2 Spatial aspects of an IMAP

All visual multimedia objects (image, text and video) incorporate spatial features. Thus, it is important to model the spatial semantics of an IMAP. These semantics may be classified in the following categories:

- *Spatial Composition*, that refers to the representation of the spatial positions and the spatial relationships among participating media objects.
- *Motion*, which is an important characteristic in the context of an IMAP and involves the absolute and/or relative motion of one or more media objects.
- *Spatial Events*, that are produced from actions that concern spatial relationships and/or motion. For instance, if two media objects spatially meet, an event is produced. Another example is the set of events that is generated by an object's continuous motion (still, in motion, accelerating, actual position etc.). These events may be of interest for the application scenario so as to trigger other actions.

A notion that we introduce in this context is the *spatial instance*, which refers to the spatial coordinates of a rectangle bounding an area of interest (x_1, y_1, x_2, y_2) relative to the origin of an IMAP window.

2.3 The notion of events

The concept of events is defined in several research areas. In the area of Active Databases [Ga94, GJS92a, GJS92b, CM93] an event is defined as an instantaneous *happening of interest* [GJS92a]. An event is caused by some action that happens at a specific point in time and may be atomic or composite. In the multimedia literature events are not uniformly defined. In [SW95] events are defined as a temporal composition of objects, thus they have a temporal duration. In other proposals for multimedia composition

and presentation, e.g., [HFK95, VS96], events correspond to temporal instances. We follow the latter understanding of the temporal aspect of events and consider events to be instantaneous.

Multimedia information systems, however, widen the context of events, as defined in the domain of active databases. In addition to the temporal aspect of an event, which is represented by a *temporal instance*, there are events in IMAPs that convey spatial information. This may be represented by a *spatial instance*. For example, an event captures the position of the presentation of visual objects at a certain point in time. Another aspect, crucial in the IMAPs context, is that, although the number and multitude of events that are produced by both the user and the system may be huge, we may be interested only in a small subset of them. Thus, events must be treated in a different way in this context, as compared to Active Databases.

We define an event in the context of IMAPs as follows:

An event is raised by the occurrence of an action and has attached a spatial and temporal instance. The event is recognized by some interested human or process.

3. CLASSIFICATION AND MODELING OF EVENTS

As mentioned in the previous definition, all events are attached to a temporal instance relative to some reference point, usually the beginning of the IMAP. Apart from a temporal instance we assign a spatial instance to an event in case it is related to a visual media object. This is essentially the rectangle that bounds an event (e.g., the screen area where the presentation of an image takes place). In some trivial cases though (e.g., mouse click), the rectangle is reduced to a point.

Thus, it is meaningful to integrate the two notions of temporal and spatial instances in the definition of events. Therefore, we introduce the term *spatio-temporal instance* whose representation in a tuple form is: (sp_inst, temp_inst), where sp_inst is a spatial instance and temp_inst is a temporal instance as defined in sections 2.1 and 2.2, respectively. Events can, however, be purely temporal as this is the case for the start event of an audio clip.

In order to assist the authors in the specification of IMAPs, we have to provide them with a fundamental repertoire of events. In the framework of IMAPs we further classify the events into categories. The classification of the events is done on the basis of the entity that produces the event. The elaborated categories are those presented section 3.1. This classification forms the basis for the object-oriented modeling which we propose in section 3.2.

3.1 Classification of events

3.1.1 Events caused by interaction of a user with the IMAP

These are the events that are generated explicitly by user interactions within the IMAP context. They are mainly input events as the user interacts with the system via input devices such as mouse, keyboard, touch screen, etc.. Such events are:

- *Mouse events* are related to mouse manipulations such as: mouse button clicks, mouse over an object, textline of a list box selected, drag and drop events or scrolling events.
- *Keyboard events* are generated each time a user presses and/or releases a key.

Other input events, that may not be applicable for the moment, though we refer to them for the completeness of the approach, are for instance voice input events and events related to touch-screens.

Temporal access control events are the well known actions *start, pause, resume, stop, fast forward, rewind, random positioning in time* and concern the execution of one or a group of media objects. These events could be classified into the group of mouse or keyboard events. They bear, however, specific semantics for the manipulation of a media object's presentation initiated by the user, and this is the reason why we consider them separately in the classification procedure.

3.1.2 Intra-object events

This category includes events that are recognized due to the internal functionality of an object presented in an IMAP. This functionality is realized in object-oriented approaches as method invocation. For instance, the invocation of a method corresponding to temporal access control such as `myObject.start()` produces an intra-object event. Another source of intra-object events are state changes. The viable states of an object as regards its presentation may be temporal (*active, idle, suspended*) and/or spatial (*shown, hidden, and layer classification* information, etc.). State events occur when there is a state change of a media object, e.g., image I is hidden, audio A started, etc.. Intra-object events may indicate a discontinuity in the continuous presentation of a presentation object. For instance, a video presentation produces an „this is the frame where I am in the presentation“ event on every tenth frame presented. Additionally, timer events may be of interest to denote certain points on the timeline. These events are classified in the intra-object category since a timer can be considered as an object itself. System signals that indicate actual system state (such as low network capacity) result in intra-object events that (e.g., indicate that the video presentation is too slow) are not classified separately.

3.1.3 Inter-object events

Such events occur when two or more objects are involved in the *occurrence of an action of interest*. These events are raised if spatial and/or temporal relationships between two or more objects hold. In the spatial case, an inter-object event can occur if one object, moving spatially, meets another presentation object. A temporal inter-object event can occur when the deviation between the synchronized presentation of two continuous media objects exceeds a threshold. Moreover, we may consider spatio-temporal inter-media synchronization events. Such events occur, for instance, when two objects are in relative motion during a specified temporal interval (e.g., object A approaches object B before 2am). As mentioned before system signals are not classified separately. However, they can result in inter-object events if, e.g., a low network capacity leads to a video presentation that is too slow (intra-object event) and this raises an inter-object event as a video presentation and an associated audio presentation are not synchronized anymore.

3.1.4 Application events

In this category we are interested in events related to the IMAP *state*. An IMAP state bears information that has been contributed essentially by all the objects currently presented in the IMAP. Indeed, an IMAP as a whole can be *idle*, *suspended*, or *active*. Moreover, during the execution of an IMAP, overall Quality of Service parameters are to be taken into consideration [TKWPC96]. Thus, it would be useful if events that indicate a certain state of the IMAP, were generated. An event indicating that the IMAP is idle may lead to an acoustic signal that invites an end user to interact, and thereby proceed with the IMAP. An application event can also indicate that the overall presentation quality falls below a given (user defined) threshold set. For instance, an IMAP presents synchronized video and audio data and the network capacity is low. Then single audio and video presentation generate intra-object events that indicate that their presentation quality is going down to initiate corresponding actions on the video and the audio presentation. But at the same time it is necessary to have an application event indicating that the overall presentation quality is going below a certain threshold set. Then, the occurrence of such an event can be assigned with a specific action, e.g., only still image is presented instead of the video and present only audio data.

3.1.5 User-defined events

In this category we address the events that are defined by the IMAP designer. They are related to the content of the IMAP execution. A user-defined event can refer to the content of a media object, i.e. to the occurrence of a specific pattern in a media object. For instance, an event is to be raised if the head of a news speaker occurs in a video frame to indicate that the boring advertisements are over and the interesting news are now on.

IMAP authors and developers are provided with an initial set of atomic events, namely the ones in the aforementioned categories. In many cases, though, they need to define complex events that are somehow composed of the atomic ones. For instance, assume two atomic events e_1 and e_2 corresponding to the start of video clips A and B, respectively. The video clips are started due to events previously occurred in the IMAP. The author might define a composite event e_3 to be raised when e_1 occurred within 3 seconds after e_2 occurred. Another example of a composite event is the occurrence of temporally overlapping presentations of two images img_1 and img_2 between 10:00am and 11:00am. Thus, it is desirable that composite events may be defined by means of a provided set of composition operators. The necessary operators are defined and classified in section 5.

Hereafter we propose an object-oriented modeling approach for events in IMAPs, based on the event concept and classification that has been presented above.

3.2 Object-oriented modeling of events

An event occurs due to an *action* as defined in section 2. This action is caused by a *subject* that may be one or a group of presentation objects participating in the IMAP. If the presentation object is an interaction element such as a button that has been pressed, the action that caused an event is the user inter-action with

the button. Each event occurs at a specific point in time and/or position to which we further refer to as the event's *spatio-temporal signature*. In principle, the event may affect or be related to one or a group of media presentations, further called *object* of the event. Therefore, the modeling of an event must reflect the action that generated the event, its subject and object, as well as its spatio-temporal signature.

In addition to the classification scheme presented in section 3.1, we classify the events into two layers according to their applicability range. For this second classification, we define the notion of *generic* and *application-specific* events. The generic events are those that convey the semantics of the event, i.e., the action that generates the event. Such a generic event would be the event *Start*, that occurs when an continuous media object presentation is started. The event features spatio-temporal signature, subject, and object, however, are not assigned with values in a generic event. The generic events are the „template“ for the IMAP definition. The application-specific events are specializations of these generic events. They are defined on the basis of objects belonging to a specific IMAP (e.g., the event *My_background_music.Start* is defined for the media object *My_background_music*). Such events are defined during authoring time by the application designer. They assign the values relevant to the application such as the media object to the event attributes. An initial set of generic events is provided to the authors in order to define the application-specific events, which then serve as the basis for definition of the entire IMAP scenario. When the IMAP is executed, the application specific events are instantiated, that is, the events actually occur. These *instances* are to be recognized and evaluated by the IMAP's execution. At instantiation time each instance is assigned with its spatio-temporal signature (e.g., *My_background_music.Start(3.5 sec)*).

The result of the two classification schemes presented above is a class inheritance tree which is illustrated in Figure 2. A related issue is the persistency of generic and application-specific events. It is desirable for many purposes (like reusability, cooperative authoring, consistency, etc.), to have these events classes stored in a database system. Thus, we also gain all the well known advantages of DBMSs for the storage and management of events. The instances of events, however, are created only at execution time and are not persistent in the database.

Herefter, we elaborate on the object-oriented specification of the Event class that serves as the root of the hierarchy. First, we define the appropriate data types that are required for the definition of the class. We define a data type *objectList* to represent a list of objects such as media objects, input devices etc. The data type *actionList* is needed in order to represent atomic or complex actions that generated the event. Finally, we need a type *spatio_temporal_instance* for the representation of spatio-temporal signatures of events as defined before.

According to the aforementioned event definition, we need the following attributes to represent a generic event: The subject and object attributes, that are of type *objectList* essentially representing the objects that caused or are affected by the event, respectively. The attribute *spatio_temporal_signature* takes to the spatial and temporal instances attached to the event when it actually occurs.

Then the structure of the Event class in an object-oriented pseudo language is as follows:

```

class Event inherits from object
attributes // attribute name           attribute's data type,
    subject                           objectList;
    action                             actionList;
    object                             objectList;
    spatio_temporal_signature         spatiotemp_instance;
end

```

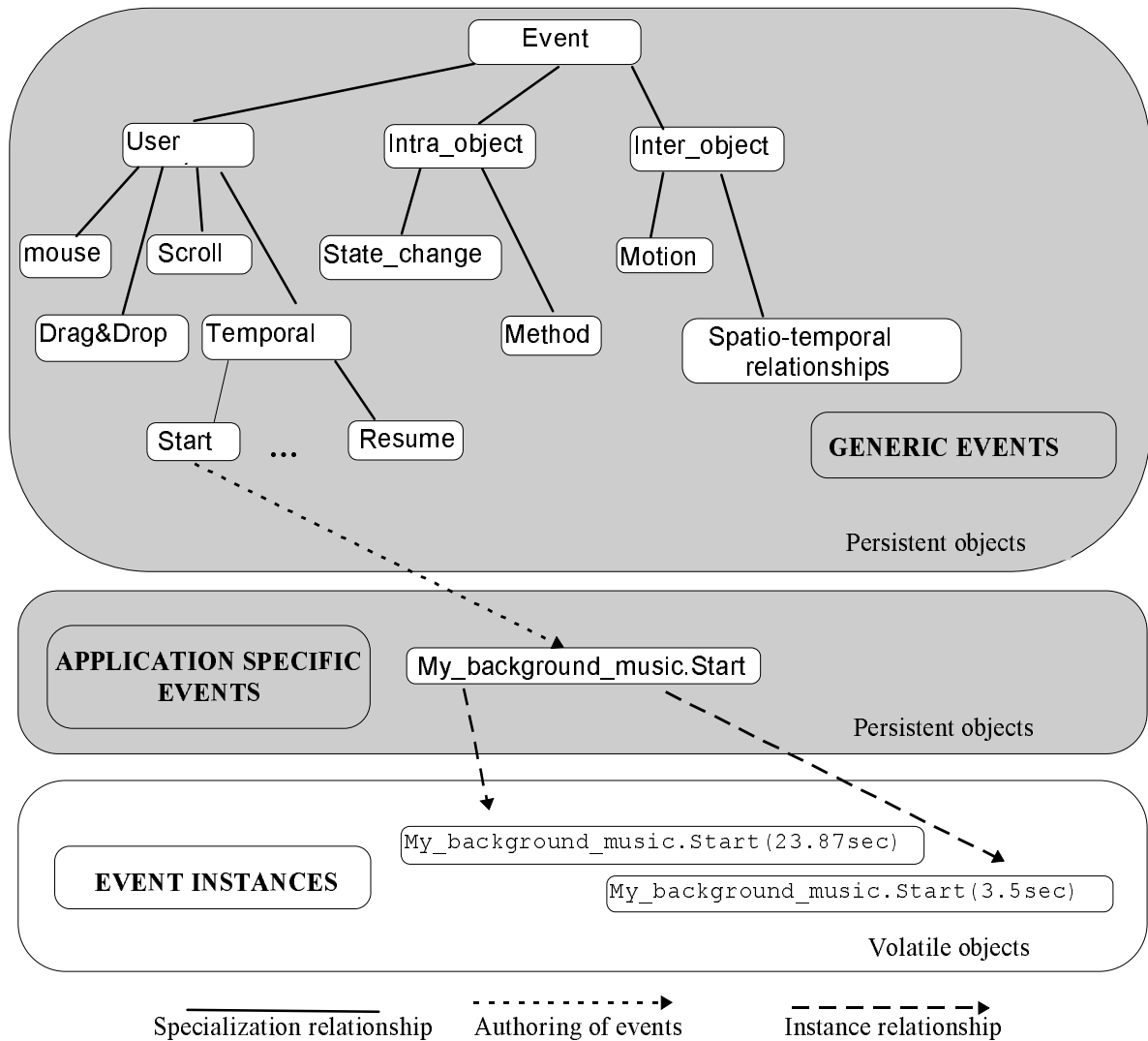


Fig. 2. Inheritance and instance relationships among events classes in the framework of IMAPs

In order to model the inheritance tree illustrated in Figure 2, the class Event is specialised by subtyping mechanisms with the respective structure. For instance, a class StateChangeEvent may have additional attributes as new_state, former_state. Subtyping also means that certain values of attributes in subclasses are already set, e.g., the generic event class StartEvent assigns the value start to the action attribute. In this paper, however, we do not elaborate all the definitions of the other generic event classes as they simply specialize the aforementioned class description. The generic events are to be persistently stored and managed by a DBMS. The object-oriented specification serves as the basis for the implementation of the event concept in a distributed multimedia database environment (see further sections).

The application-specific events are persistent instances of generic events. The generic events simply offer the structure and behavior of the event types classified in section 3.1 to an IMAP designer. During the design of an IMAP, application-specific events are created to describe the *high level structure* to the IMAP. The application-specific events hereby assign application-specific values to the attributes of a generic event. For the creation of an application-specific a generic event is instantiated, assigned to the necessary and persistently stored in the context of an IMAP in the database. Only the volatile instances of the application-specific event assign all the remaining attributes with values. For instance, the generic event Start assigns the value `start` to the action attribute. The corresponding application-specific event `My_background_music.Start` instantiates an object of class `Start`, adds the identifier of the medium `My_background_music` to the subject attribute. During authoring this application-specific event is made persistent. When it comes to the execution of the corresponding IMAP, the application-specific events are instantiated and the instance of this event are each assigned values to the remaining attributes, e.g., the `spatio_temporal_signature` with the values 23.87 sec or 3.5 sec., or the attribute `new_state` is assigned to the value `paused`. Figure 2 illustrates the described relationships between the persistent generic and application-specific events and the volatile instances of events during the execution of an IMAP.

3.3 Events and multimedia scenarios

A multimedia scenario is defined as the high level definition of a multimedia application functionality that covers the spatial and temporal synchronization of participating media objects and the handling of interaction between the user, the system and the application [VH93]. In this paper we will exploit a scenario specification approach as described in [VS96] for the purpose of illustrating the potential of our events modeling approach. Events in that approach, are not analyzed regarding their semantic and composition aspects. According to that approach, the scenario of an IMAP may be represented by a set of tuples so called *scenario tuples*. Each tuple represents a fundamental or autonomous functionality in the framework of an IMAP and includes the events that will trigger this functionality, the events that will stop it and of course the list of actions that will be executed in when this tuple is triggered. It may also represent constraints that apply to the execution of the tuple.

The scenario tuples attributes as described in [VS96] are:

`start_time`: identifies the event that triggers the execution of actions described in the action expression.

`stop_time`: identifies event that terminates the execution of this tuple.

`action_expression`: gives the list of actions that will take place and their composition in the spatial and temporal domains.

`content_condition`: is the expression that indicates a condition that refers to the content of the multimedia objects included in this scenario tuple.

`synch_events`: the beginning and the end of a scenario tuple might generate events. These events may be named and exploited for synchronization.

`Constraints`: represents a set of constraints (spatial, temporal etc.) that must be fulfilled during the execution of the scenario tuple.

`is_active`: indicates whether the tuple execution is active

As it is evident, events are an essential part of multimedia scenarios since they are the entities that cause tuple execution, suspension and facilitate tuple communication through the `synch_events`.

4. COMPOSITION OF EVENTS

As we mentioned before, it is important to provide the tools to the authors for the definition of composite events. The composition of events in the context of an IMAP has two aspects:

1. *Algebraic composition* is the composition of events according to algebraic operators, adopted to the needs and features of an IMAP.
2. *Spatio-temporal composition* reflects the spatial and temporal relationships between events.

First we define some fundamental concepts:

Spatio-temporal reference point (θ):

This is the spatio-temporal start of the multimedia application named as θ . This serves as the reference point for every spatio-temporal event and instance in the IMAP.

Temporal interval:

This is the temporal distance between two events (e_1, e_2) namely the start and end of the interval.

$t_int ::= (e_1, e_2)$, where e_1, e_2 are events that may either be attached to predefined temporal instances relative to some reference or occur asynchronously.

Relative events:

A relative event r_e occurs after a specific temporal interval relative to another event.

$r_e ::= (e_1, t_int)$, event r_e occurs t_int time units after event e_1 .

4.1 Algebraic composition of events

In many cases the author wants to define specific events that relate other existing events. We exploited some of the operators presented in other proposals on composite events in Active Databases [Ga94, GJS92b]. We distinguish between the following cases:

Disjunction:

$e = OR(e_1, \dots, e_n)$: This event occurs when at least one of the events e_1, \dots, e_n occurs. For instance we may be interested in the event e occurring when button A (e_1) or button B (e_2) was pressed.

Conjunction:

$e = ANY(k, e_1, \dots, e_n)$: This event occurs when at least any k of the events e_1, \dots, e_n occur. The sequence of occurrence is irrelevant. For example, in an interactive game a user proceeds to the next level when she/he is successful in two out of three tests that generate the corresponding events e_1, e_2 , and e_3 .

$e = \text{SEQ}(e_1, \dots, e_n)$: This event occurs when all events e_1, \dots, e_n occur in the order appearing in the list. For example, in another interactive game the user proceeds to the next level when she/he succeeds in three tests causing the events $e_1, e_2,$ and e_3 one after the other.

$e = \text{TIMES}(n, e_1)$: This event occurs when there are n consecutive occurrences of event e_1 . This does imply that other events *can* occur in-between occurrences of e_1 .

In many cases the authors want to apply constraints related to event occurrences in specific temporal intervals. To facilitate this requirement we define a set of operators that are of interest in the context of multimedia applications:

Inclusion:

$e ::= \text{IN}(e_1, t_int)$, event e occurs when event e_1 occurs during the temporal interval t_int . For example, in an IMAP we might want to detect three mouse clicks in an interval of 1 sec., so that a help window appears. If $t_int = (e_2, e_3)$, where e_2 corresponds to the starting point of a timer while e_3 corresponds to the end of a timer whose duration is defined as 1 second. The desired event would then be defined as $e = \text{IN}(\text{TIMES}(3, \text{mouse.click}), t_int)$.

Negation:

$e ::= \text{NOT}(e_1, t_int)$: event e occurs when e_1 does not occur during the temporal interval t_int .

Strictly consecutive events:

In some cases we are interested in whether a series of events of interest is „pure” or mixed up with other events occurring. The event $e ::= \text{S_CON}(e_1, \dots, e_n)$ is raised when all of e_1, \dots, e_n have occurred in the order appearing in the list and *o other* event occurred in between them.

4.2 Spatio-temporal composition of events

The term spatio-temporal stands for the spatial and/or temporal ordering of events, e.g., e_1 to occur spatially and/or temporally after e_2 .

4.2.1 Temporal composition of events

Here we discuss the temporal aspect. Since events are of zero temporal duration, the only valid temporal relationships between two events are *fter*, *before*, and *simultaneously* .

Temporal relationship after:

$e ::= \text{after}(e_1, e_2)$, event e occurs when e_1, e_2 both occurred and the temporal signature of e_1 is smaller than the corresponding signature of e_2 .

Temporal relationship before:

$e ::= \text{before}(e_1, e_2)$, event e occurs when e_1, e_2 both occurred and the temporal signature of e_1 is smaller than the corresponding signature of e_2 .

Temporal relationship simultaneously:

$e ::= \text{simultaneously}(e_1, e_2)$, event e occurs when e_1, e_2 both occurred and the temporal signature of e_1 is equal to the corresponding signature of e_2 .

There are cases in which algebraic operators and temporal relationships may be used interchangeably. For instance, for two events e_1, e_2 the expressions $\text{after}(e_1, e_2)$ and $\text{SEQ}(e_1, e_2)$ convey the same fundamental semantics. Although the former is a temporal relationship between *two* events, the latter is a conjunction operator bearing also temporal semantics for a *list* of events.

4.2.2 Spatial composition of events

The spatial aspects are related to presentation and/or motion of spatial objects (images, buttons etc.). We can assume that each event has a spatial signature that is the rectangle (Minimum Bounding Rectangle) that bounds the area of the event. Spatial relationships between objects involve three different aspects: topology, direction and metrics (see Fig. 3.). A model that represents all these aspects is defined in [VTS96]. We will exploit this model for defining a complete spatio-temporal event composition scheme.

4.2.3 Generic spatio-temporal composition scheme

Having defined all the temporal and spatial relationships among events, we can now introduce a complete spatio-temporal composition scheme. This set of relationships includes the set of all possible spatio-temporal relationships (169 spatial relationships * 3 temporal relationships = 507 spatio-temporal relationships) between two events.

A requirement that arises during event composition specification, is the definition of metrics between events. Thus, we define the notion of *spatio-temporal distance* between two events. For this purpose we consider the definition of the spatial distance between two rectangles as the Euclidean distance between their *closest vertices* as defined in [VTS96]. For our spatio-temporal composition scheme we extend this definition with the temporal distance concept, resulting in the following definition:

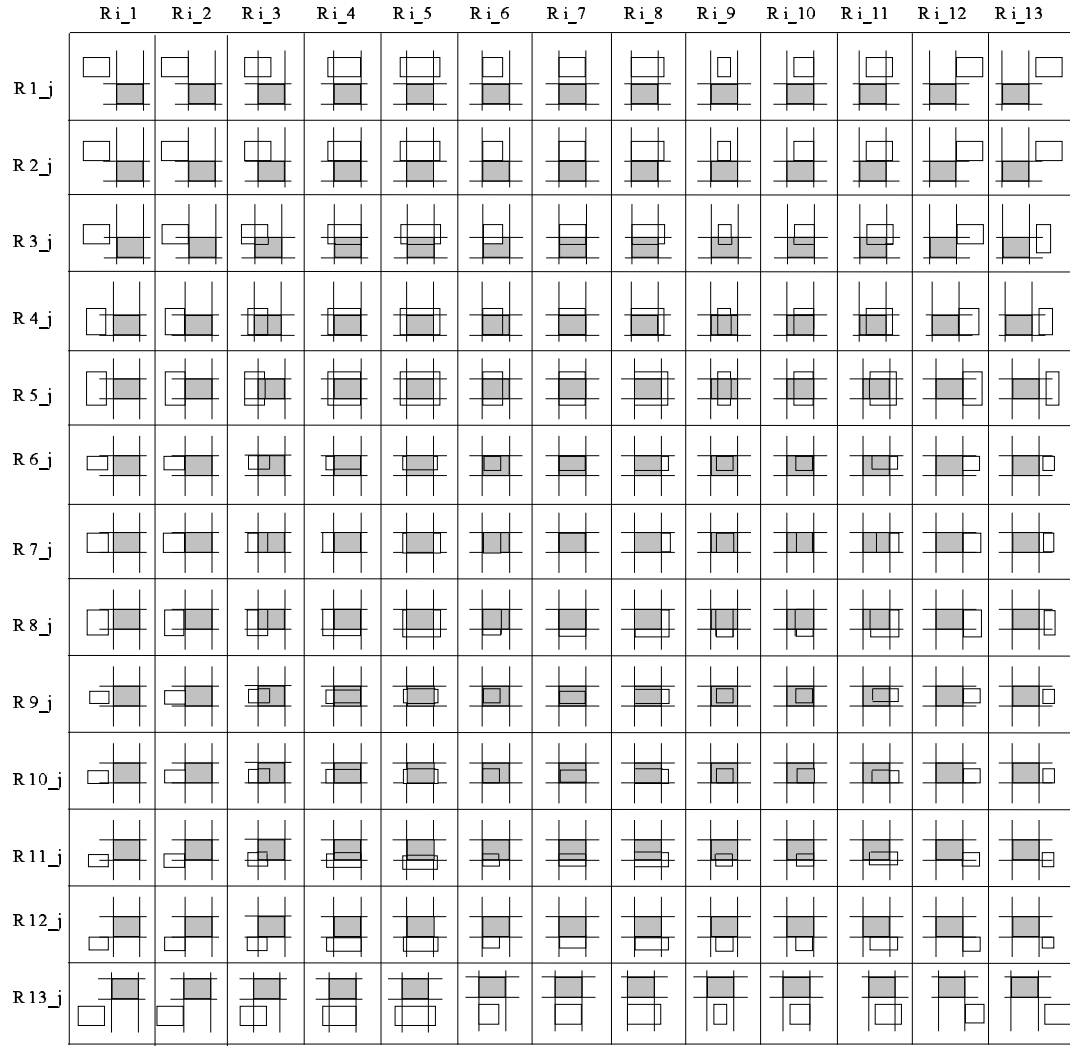


Fig. 3. Directional relationships between two spatial objects (including topological information)

Spatio-temporal distance:

Given two events e_1, e_2 having corresponding spatio-temporal signatures: $(x_{11}, x_{12}, y_{11}, y_{12}, t_1), (x_{21}, x_{22}, y_{21}, y_{22}, t_2)$, then the spatio-temporal distance of the two events defined as : $\text{spatio_temporal_distance} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (t_2 - t_1)^2}$, where $(x_a, y_a), (x_b, y_b)$ are the coordinates of the closest vertices of the two spatial distances.

The generic spatio-temporal composition scheme for events, based on the EBNF notation, is defined as follows:

```

op      ::= e1 Rel e2
Rel     ::= (temporal_relation, spatial_relation, sp_temp_dist)
temporal_relation ::= "after" | "before" | "simultaneously"
spatial_relation  ::= Ri_j
i, j             ::= [1,13]
spatio_temp_dist ::= spatio_temporal_distance

```

By having this kind of event composition scheme we gain simplicity, maintain relationships between events, and facilitate queries and playbacks of existing scenarios or simulation of IMAPs.

5. EXAMPLES OF EVENT DEFINITION AND EVENT COMPOSITION

In this section we present two examples that demonstrate the potential of our proposal. In the following application-specific event definitions, only the appropriate attributes have values attached. For each event the attributes and their **values** are given..

5.1 Example 1

Assume the following IMAP scenario:

“Assume an IMAP that presents maps of three different countries, Germany, Greece, and Spain consecutively at a random order and for some temporal interval. The user moves a magnifying lens over the application window while at any time may click the left mouse button. We want to detect if the mouse click took place when the lens was over a map, and on which map the mouse click was in, in order to start the appropriate video clip. Moreover, if currently it is summer time the appropriate summer video clips will be presented, otherwise a corresponding cultural video clip will be presented”.

In this example the feature that makes the use of events essential is the random temporal duration of each map presentation. This makes impossible to know at authoring time the exact video presentation times. Here, we are not concerned with the mechanisms that produce the random order and temporal duration of the presentation of the maps. The spatio-temporal ordering of the scenario is illustrated in Figure 4.

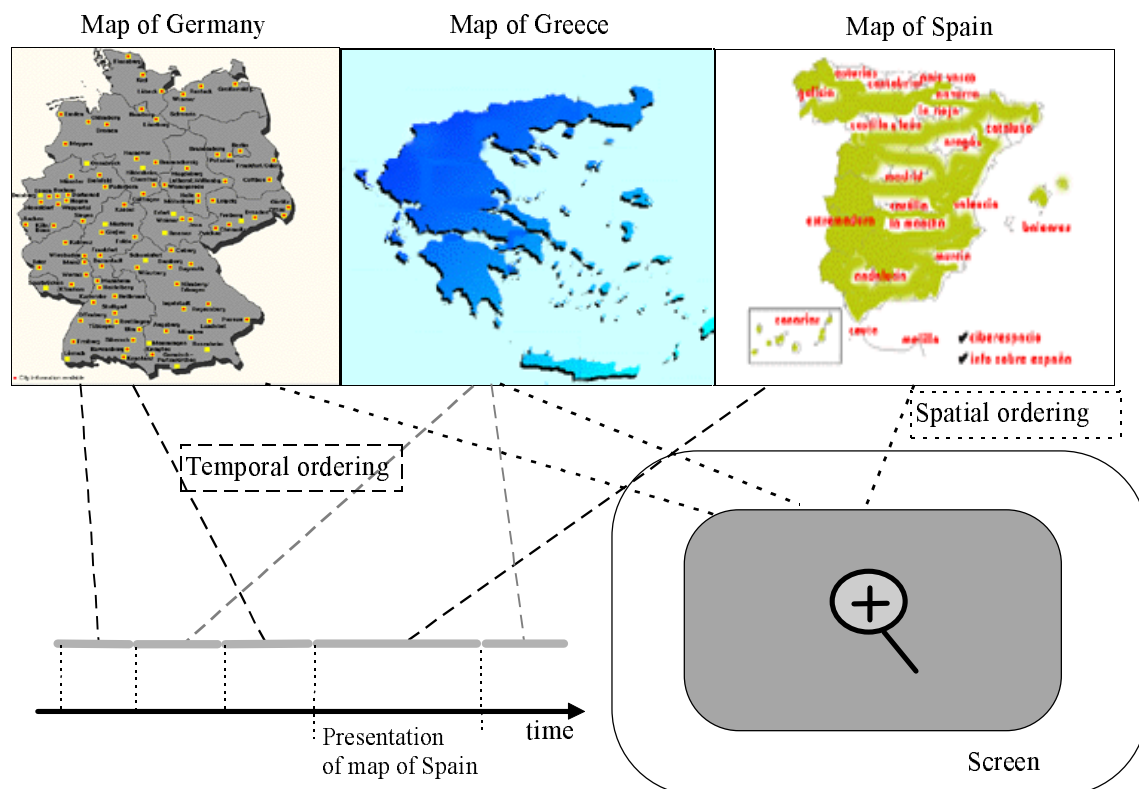


Fig. 4. Temporal and spatial ordering of the maps' presentation in the IMAP

The appropriate application-specific events definition follow. First, we want to represent the event produced when the user clicks the mouse left button:

```
event lens_mouse_click // instantiated at runtime
  attributes
    subject mouse
    action LeftButtonDown()
    object null // no object is affected by the event
    spatio_temporal_signature null// filled on instantiation
  end
```

Then, we are interested to represent the events produced when the „lens“ overlaps any map. One application specific event per map is specified:

```
event lens_overlaps_Greece // instantiated at runtime
  attributes
    subject lens
    action overlaps(map_of_Greece)
    object null // no object is affected by the event
    spatio_temporal_signature null // filled on instantiation
  end
```

In the same way we define the events `lens_overlaps_Germany` and `lens_overlaps_Spain` with respect to `map_of_Germany` and `map_of_Spain`, respectively.

We are also interested in representing the application-specific events produced when the maps are displayed in the IMAP window. Thus, we are able to know at any time during the presentation which map is currently presented:

```
event Greece_map_displayed // instantiated at runtime
  attributes
    subject map_of_Greece
    action display()
    object null // no object is affected by the event
    spatio_temporal_signature null // filled on instantiation
  end
```

In the same way we define the events `Germany_map_displayed` and `Spain_map_displayed` with respect to `map_of_Germany` and `map_of_Spain`, respectively.

Eventually, we need to specify the events that are raised when the user clicks the mouse over a map and this takes place during summer time, so that the summer or winter video clip can be played, respectively. The appropriate temporal interval summer is defined by `t_int summer("01/Jun", "31/Aug")`. Moreover, here we exploit the aforementioned composition operators:

```
event mouse_click_over_Greece_in_summer // instantiated at runtime
  attributes
    subject mouse
    action in(
      (( lens_mouse_click and lens_overlaps_Greece)
        (after,_) Greece_map_displayed),
      summer)
    object null // no object is affected by the event
    spatio_temporal_signature null // filled on instantiation
  end
```

In the same way we define the events `mouse_click_over_Germany_in_summer` and `mouse_click_over_Spain_in_summer` with respect to the events that concern the presentation of the map of Germany and the map of Spain, respectively.

The aforementioned events may be exploited in a scenario scheme that represents the sample IMAP as described previously (see section 3.3). Then the scenario tuple for displaying the summer video about Greece would be:

```
scenario tuple Greece_summer_video
  start_time    = mouse_click_over_Greece_in_summer
  stop_time     = Germany_map_displayed or Spain_map_displayed
  action_list   = Greece_summer_video.start()
  content_condition = null
  synch_events  = (_,_)
  constraints    = null
  is_active     =
end
```

With this scenario tuple, essentially the presentation of the `Greece_summer_video` is triggered when the user clicks the mouse over the map of Greece during summer. The tuple's execution is to be interrupted in the case that either the German or the Spanish map is displayed.

5.2 Example 2

Our event composition scheme may cover complex interactive scenarios with multiple users participating in a shared environment [ACM95]. Assume the following application scenario:

“Two users share the same application: The buttons (A, B) may be dragged in the video area VA. When a button enters (spatially) the VA area the corresponding video (A or B) is started. If the button is dragged out of VA, the video shifts to a suspended state. If it re-enters, it resumes”

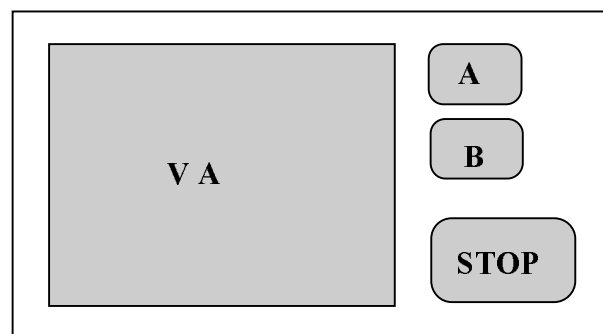


Fig. 5. The spatial layout of the sample application

The above scenario is highly interactive since it includes sharing of the application among multiple users, incorporates motion of objects and multiple dependencies among them, and incorporates events that occur in the application. We apply the following constraints that reflect user participation and dependencies among participating object states:

- If a button is occupied by a user (event `button_pressed` raised) it cannot be manipulated by any other user.
- If a video is played back in the VA area then no other video may be played back in VA before it ends or before it is explicitly stopped (by pressing button STOP).

The objects participating in the application may be in various states, considering mostly the buttons and the participating video clips. From each state there are certain actions (transitions) that lead to another state. The corresponding state-transition diagrams appear in Fig. 6a and Fig. 6b respectively. We exploit the information depicted from these diagrams in the definition of the scenario so that the above mentioned constraints are fulfilled.

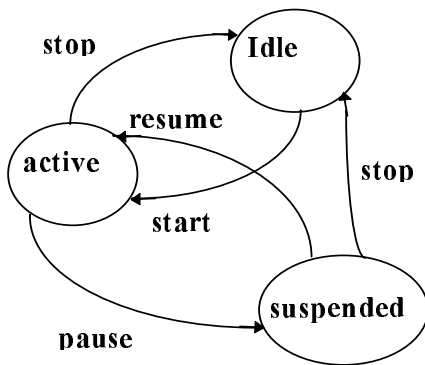


Fig. 6a: State-transition diagram for a video clip

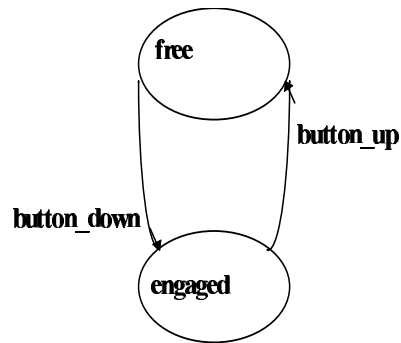


Fig. 6b: State-transition diagram for a push button depending on mouse clicks

The application definition is based on the definition of events of interest and of the resulting scenario tuples [VAZ96]. First we define the application events related to buttons and their states:

event button_A_pressed attributes subject mouse action left_button_down object button A spatio_temporal_signature end	event button_A_released attributes subject mouse action left_button_up object button A spatio_temporal_signature end
event button_STOP_pressed attributes subject mouse action left_button_down object button STOP spatio_temporal_signature end	event button_STOP_released attributes subject mouse action left_button_up object button STOP spatio_temporal_signature end
event button_B_pressed attributes subject mouse action left_button_down object button A spatio_temporal_signature end	event button_B_released attributes subject mouse action left_button_up object button B spatio_temporal_signature end
event button_A_to_drag attributes subject mouse action (mouse_move (after, ...))	event button_B_to_drag attributes subject mouse action (mouse_move (after, ...))

but_A_pressed) object button_A spatio_temporal_signature end	but_A_pressed) object button_B spatio_temporal_signature end
event button_A_enganged attributes subject button_A action but_A_pressed object spatio_temporal_signature end	event button_A_free attributes subject button_A action but_A_pressed (after,_,_) but_A_released object spatio_temporal_signature end
event button_B_enganged attributes subject button_B action but_B_pressed object spatio_temporal_signature end	event button_B_free attributes subject button_B action but_B_pressed (after,_,_) but_B_released object spatio_temporal_signature end

Here after we define the events that relate buttons to VA:

event button_A_inVA attributes subject button_A action inside(VA) object spatio_temporal_signature end	event button_A_outVA attributes subject button_A action disjoint(VA) object spatio_temporal_signature end
event button_B_inVA attributes subject button_B action inside(VA) object spatio_temporal_signature end	event button_B_outVA attributes subject button_B action disjoint(VA) object spatio_temporal_signature end

The application scenario

Above we have exploited the scenario tuple structure as defined in [VS96] The scenario is represented by the the following senario tuples as defined above:

tuple_id	start_time	stop_time	constraints	action list	synch_events
t1	button_A_inVA	but_STOP_pressed	vidB.idle()= TRUE	VidA.start()	(t1_s, t1_e)
t2	button_B_inVA	but_STOP_pressed	vidA.idle()= TRUE	VidB.start()	(t2_s, t2_e)
t3	button_A_outVA	but_A_inVA	t1_s	VidA.pause()	(t3_s, t3_e)
t4	button_B_outVA	but_B_inVA	t2_s	VidB.pause()	(t4_s, t4_e)
t5	button_A_inVA	but_A_outVA	t3_s	VidA.Resume()	
t6	button_B_inVA	but_B_outVA	t4_s	VidB.Resume()	
t7	button_A_to_drag	button_A_released	button_A_free	button_A.moveto(mouse_pos)	
t8	button_B_to_drag	button_B_released	button_B_free	button_B.moveto(mouse_pos)	

Table 2. The scenario tuples that represent the scenario of the application and the related constraints

6. IMPLEMENTATION ISSUES

Our implementation integrates the event concept proposed in previous sections into an object-oriented multimedia database system. We envisage a client/server based architecture manages events in the context of IMAPs persistently on the server and provides a multimedia database client that generates, evaluates, and processes the events produced in the context of the execution of an IMAP. We are based on the AMOS system for the implementation of events. The AMOS project (Active Media Object Stores) at GMD-IPSI currently develops a multimedia database management system (MMDBMS) which allows for the integrated management and presentation of multimedia data and multimedia applications [RNL95, BKL96, TK96]. The AMOS prototype is based on a client/server architecture. It allows for the management of multimedia data and currently supports storage, retrieval, and execution of pre-orchestrated multimedia presentations [BKL96], however with limited interaction support. But since an IMAP may have extended repertoire of interactions, the new event concept proposed in this paper is implemented in this multimedia database environment to support the definition and execution of multimedia applications with highly interactive features. With the implementation of the event concept we aim at extending the interactive features of the AMOS prototype considerably.

Hereafter, we elaborate on the modifications and extensions of the AMOS system components in order to support IMAPs. The integrated approach of the AMOS system is to model, store, and execute IMAPs within *one* system. The distribution of the functionality of the MMDBMS locates the modeling and management of data at the server, whereas the presentation of the multimedia data takes place at the database clients. In section 6.1 we address the parts dealing with the modeling of events and, based on events, the modeling of IMAPs in the database schema. Section 6.2 explains the way from IMAP modeling on the database server to the execution of an IMAP at a database client. The actual execution of IMAPs is presented in section 6.3 and emphasizes the generation, evaluation, and consumption of events during the execution of an IMAP at the client.

6.1 Modeling of events in the database schema

The central multimedia database server forms the core of the MMDBMS architecture. It manages the data in the multimedia database and controls concurrent access of the multimedia clients to a database. The database schema represents the media objects, their attributes, and behavior accessible by an application. Given this functionality, the database server provides for the modeling and management of multimedia scenarios that involve media objects. We integrate the modeling of events, which facilitate the integration of scenario tuples. This aims at integration of IMAPs into the database schema.

Each interactive multimedia scenario of an IMAP is modeled in the database schema. Each scenario is represented within the database by means of a *multimedia document* that belongs to a certain document model. Several multimedia document models or multimedia exchange formats, respectively, have been proposed so far. Prominent examples are MHEG [ISO93], HyTime [ISO92], and Object Composition Petri Nets (OCPN) [LG93]. The implementation of the (evolving) standards HyTime and MHEG did not seem

to be adequate to us as they do not support the modeling of interaction at all or at least unsatisfactory. In a first approach, we modeled the so called *presentation nets*, which are an extension of OCPN, in the database schema and implemented their interactive presentation executed at a database client [BKL96]. From our implementation experiences, however, presentation nets provide only for simple interaction specification.

To support a higher degree of „interactivity“, we aim at modeling and executing multimedia scenarios on the base of scenario tuples, as has been described in section 3.8. Each IMAP instance, that is a multimedia document, is described by a set of scenario tuples and is stored at and managed by the database server. Events are an essential part of these tuples. The start time and stop time events of a scenario tuple describes the events that trigger/stop the execution of the action part of a scenario tuple if the condition, which is also given in the tuple, holds.

Currently, we are extending the database schema by the generic and application-specific event classes which we presented in section 3.1. The generic events are modeled as classes in an inheritance tree in the MMDBMS's data definition language VML [KAN94, Vodak95]. For the definition of an IMAP, the designer first has to specify the events that are of interest to the application. For this purpose, the database schema models application-specific events that each inherit from one of the generic events. For each new application-specific event the designer creates a new event object and assigns to it the application-specific values. These values describe the objects that are involved in the event, the action that caused the event, and the subjects that are effected by the event. Subjects and objects of an event are entities participating in an IMAP. They are so far the presented media objects and in this case the designer assigns the references to media objects in the database. Participating entities can also be input devices such as the mouse which are also represented as possible entities in the schema. The different actions that can be used for an application-specific event are also modeled in the database schema. For the modeling of complex events the database schema additionally models the operators necessary for the temporal, spatial and algebraic composition of events that have been presented in section 4. All applications-specific events are stored persistently in the database in the context of an IMAP object so that they can be manipulated and/or executed.

6.2 From IMAP modeling to IMAP execution

The storage of IMAPs at the server and the execution of IMAPs at the client, requires a suitable and efficient transport mechanism between client and server. The access of database clients to IMAPs is realized with the system's database Application Programming Interface (API). For the execution of an IMAP, a client queries the database and as a result receives a non-persistent copy of the IMAP object. In fact, this means that the scenario tuples (i.e., the events, conditions, and the actions to be triggered) are transferred to the client and are stored in corresponding volatile client objects. An initial design decision is that an IMAP is entirely delivered to the client before the actual execution starts. The transport of the scenario tuples implies the transport of meta-information about the IMAP, the transport of media raw data, however, takes place only when the IMAP is actually executed.

6.3 Event processing scheme for the execution of IMAPs

Once the IMAP object is received at the client, the execution can be started. From then on, the client presentation system generates, evaluates, and consumes events and triggers the corresponding actions. In this section, we describe the system components of the client architecture for the execution of IMAPs, putting emphasis on the parts that relate to event generation, evaluation, and consumption. Figure 7 outlines the tasks that jointly perform the execution IMAP that is the presentation of an interactive multimedia scenario.

6.3.1 *Generation of atomic events*

The events are the driving force of the application. The running instance of an IMAP produces a multitude of atomic events such as mouse down, key up, the origin of a visual media object that is currently dragged, etc. However, before an IMAP is executed the events of the scenario tuples are announced with the presentation system. That is only those atomic events are delivered to the event evaluation component that are involved in the scenario tuples. This is necessary not to make the event evaluation a bottle neck, that is not to overflow these components with atomic events. For instance, if an application is purely monitored by the user, there is no necessity to generate timer events. Otherwise, the necessary timer events are generated indicating that, e.g., the 10th second of the presentation has been reached. Many of the events are produced by the graphical user interface (GUI) performing the actual presentation. The GUI produces atomic events such as mouse-down, screen position of a drag and drop interaction, etc.. On the announcement of the events with the presentation system also the number of events generated by the GUI can be limited to those atomic event types of interest. The event instances of the event generation are input to the event evaluation component and hereby for the formation of instances of application-specific events.

6.3.2 *Evaluation of atomic events and detection of application-specific events*

This component instantiates the application-specific events that have been presented in section 3.1 (see also Fig.2). The instances of application-specific events are not persistent, as they exist only during runtime of the IMAP.

The event evaluation and detection module evaluates the atomic events delivered by the event generation and detects occurrences of IMAP application-specific events. For each event that is received the event evaluation checks if the event matches with any event expression in the scenario tuples or is a part of a composite event. If the latter applies, the received event is subject to a further evaluation. Herein, the problem of events evaluation arises. This issue is addressed in active databases area. A strategy must be adopted on how to maintain event histories as such an event may participate in more than one complex event of interest at different times. A solution proposed is the handling events history with a petri nets for each complex event [GD94]. In this approach, if a primitive event is registered the corresponding input place of the petri net is marked. Then the petri net possibly fires transitions and an output place of the net can be marked. This means that the corresponding complex event was fulfilled.

If one or more event expressions are totally matched, the corresponding action executions are triggered. The events evaluation and detection should not decrease the presentation quality due to the evaluation procedure computational overhead so if not to delay the specified temporal course of the IMAP.

6.3.3 Consumption of events - action execution

The IMAP execution is the result of the execution of the action part of the scenario tuples. The execution of an action implies the execution of operations on presentation objects such as start, stop, move, hide, etc. The action execution has to take into account the network delays in delivering multimedia data from the server. If more than one complex event is detected, then the corresponding actions are executed in parallel. Here a multithreaded architecture should be considered.

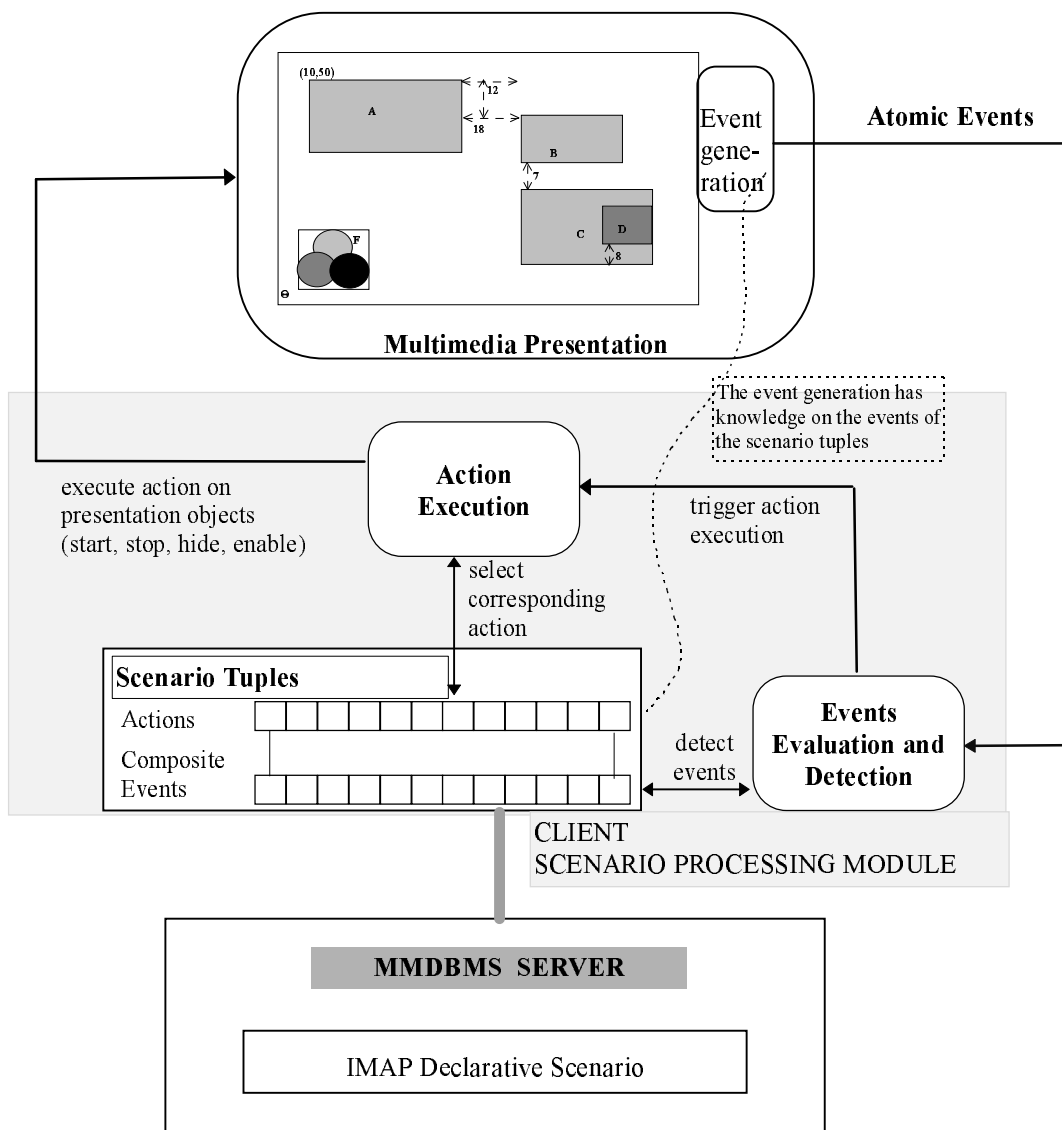


Fig. 7: Event processing architecture

7. RELATED WORK

In this section we review Multimedia synchronization modeling approaches and Multimedia document standards regarding the interaction support they provide.

In [IIN94] a model for spatio-temporal multimedia presentations is presented. The temporal composition is handled in terms of Allen relationships, whereas spatial aspects are treated in terms of a set of operators for binary and unary operations. The model lacks of the following features: there is no indication of the temporal causal relationships (i.e., what are the semantics of the temporal relationships between the intervals corresponding to multimedia objects). The spatial synchronization essentially addresses only two topological relationships: overlap and meet, giving no representation means of the directional relationships between the objects (i.e. Object A is to the right of object B) and the distance information (i.e. object A is 10 cm away from object B). The modeling formalism is rather oriented towards execution and rendering of the application rather than for authoring.

In [Ha96] a synchronization model is presented. This model covers many aspects of multimedia synchronization like: incomplete timing, hierarchical synchronization, complex graph type of presentation structure with optional paths, presentation time prediction and event based synchronization. As regards the events, they are considered merely as presentations constrained by unpredictable temporal intervals. There is no notion of the events semantics and also no notion of composition scheme.

In [SD96] a presentation synchronization model is presented. Important concepts introduced and manipulated by the model are the objects states (Idle, ready, In-process, finished, complete). Although events are not explicitly presented, user interactions are treated. There are two categories of interaction envisaged: buttons and user skips (forward, backward).

As mentioned in [BLA96], event based representation of a multimedia scenario is one of the four categories for modeling a multimedia presentation. There it is mentioned that events are modeled in HyTime and HyperODA. Events in HyTime are defined as presentations of media objects along with its playout specifications and its FCS coordinates. As regards HyperODA events are instantaneous happenings mainly corresponding to start and end of media objects or timers. All these approaches suffer from poor semantics conveyed by the events and moreover they do not provide any scheme for composition and consumption architectures.

MHEG [ISO93] allows for user interaction between the selection of one or more choices out of some user-defined alternatives. The actual selection by a user determines how a presentation continues. Another interaction type is the modification interaction which is used to process user input. For the modification interaction a content object is offered to a user, e.g., to enter a text with which a video is selected or a number to change the volume of an audio. Temporal access control actions, however, cannot be modeled with MHEG. The hypermedia document standard HyTime [ISO92] offers no modeling of interaction. Object Composition Petri Nets (OCPN) [LG93] do not allow modeling of interaction. The investigations in

[BKL96] that use Presentation Nets, an extension of OCPN, to model interactive multimedia applications have resulted in only limited interaction functionality.

All the aforementioned models and approaches do not capture the rich semantics of events that occur in an IMAP. Moreover, , none of those approaches propose a composition scheme.

8. CONCLUSIONS

In this paper, we presented a classification and a modeling approach of events in IMAPs, which can serve as the basis for the definition of IMAPs in a database environment. Moreover we defined a rich composition scheme for events that covers algebraic and spatio-temporal aspects. Furthermore, we presented a MMDBMS client architecture that generates, evaluates, and consumes events produced in the context of an IMAP. This design corresponds to the need for handling, in a flexible, declarative and rich manner, the interactions in the context of IMAPs.

The overall goal is the complete modeling of IMAPs putting emphasis on interactivity. We claim that events are the appropriate means for modeling interactions. Our contributions can be summarized as follows:

- Formal introduction of event concept into IMAPs. We introduce a formalism for the representation of events in IMAPs based on the object-oriented design paradigm.
- Extension of the event concept towards spatial and temporal domains. We extended the event concept, as defined in the Active Databases area adding spatial semantics.
- Classification of events in the context of IMAPs. In IMAPs a large variety of events of different nature are produced. We attempted two classifications of these events: the first one according to the objects that generated them, and the second one according to the range of their applicability.
- Rich composition scheme covering algebraic, spatio-temporal aspect of composition. It is important to provide the tools to the authors for the definition of composite events. The algebraic composition in the context of an IMAP is the composition of events according to algebraic operators. With the spatio-temporal composition events can be related spatially and temporally. We provided a composition scheme that integrates the three aspects and provides a powerful tool to authors of complex scenarios for IMAPs.
- Implementation design. We presented an implementation design that is based on and extends an existing MMDBMS architecture for processing of events.

There are further areas for research and development as regards events for modeling IMAPs in a MMDBMS:

- Full specification of IMAPS scenarios based on the events scheme. We are working on a complete IMAP specification scheme that is based on ECA-type of scenario tuples. The E part represents the

event (s) that will trigger the execution of the tuple, the C part is the condition(s) that must be fulfilled while the A part is the presentation (simple or complex) that will be executed.

- Extensions so as to cover motion and the resulting events. Motion is one of the under-addressed issues in Multimedia Synchronization area. We plan to work on this field so that motion of objects (and the related events) in IMAPs is integrated in an overall modeling.
- Complete design of a MM DBMS architecture for execution of IMAPs

REFERENCES

- [Al83] J.F. Allen: "Maintaining Knowledge about Temporal Intervals". *Communications of the ACM*, Vol. 26, No. 11, November 1983, pp. 832-843.
- [BKL96] S. Boll, W. Klas, M. Löhr. "Integrated Database Services for Multimedia Presentations". In S.M. Chung (editor), *Multimedia Information Storage and Management*, to be published by Kluwer Academic Publishers, 1996.
- [BS96] G. Blakowski, R. Steinmetz, "A Media Synchronisation Survey: Reference Model, Specification, and Case Studies", *IEEE Journal on Selected Areas in Communications*, vol 14, No. 1, Jan. 1996, pp. 5-35.
- [BZ93] M.C. Buchanan, P.T. Zellweger. "Automatic Temporal Layout Mechanisms". In Proc. ACM Multimedia '93, Anaheim, CA, August 1993, pp. 341-350.
- [CM93] S. Chakravarthy, D. Mishra. "Snoop: An Expressive Event Specification Language For Active Databases". Technical Report, UF-CIS-TR-93-00, University of Florida, 1993.
- [DD94] S.J. DeRose, D.G. Durand. "Making Hypermedia Work", Kluwer Academic Publishers, 1994.
- [Ga94] S. Gatzou. "Events in Active Object-Oriented Database System". Technical Report, ETH, Zürich, 1994.
- [GD94] S. Gatzou, K.R. Dittrich. "Detecting Composite Events in an Active Database System Using Petri Nets". In: J. Widom, S. Chakravarthy (editors). *Proceedings of the 4th International Workshop on Research Issues in Data Engineering: Active Database Systems*. Houston, February 1994.
- [GJS92a] N.H. Gehani, H.V. Jagadish, O. Shmueli, "Composite Event Specification in an Active Databases: Model & Implementation". *Proceedings of VLDB Conference*, 1992, pp. 327-338.
- [GJS92b] N.H. Gehani, H.V. Jagadish, O. Shmueli, "Event Specification in an Active Object Oriented Database", In *Proceedings of the ACM/SIGMOD Conference*, 1992.
- [Ha96] M. Handl, "A New Multimedia Synchronisation model", *IEEE Journal on Selected Areas in Communications*, vol 14, No. 1, Jan. 1996, pp. 73-83.
- [HFK95] N. Hirzalla, B. Falchuk, A. Karmouch. „A Temporal Model for Interactive Multimedia Scenarios“. *IEEE Multimedia*, Fall 1995, pp. 24-31.
- [Ho92] P. Hoepner, "Synchronizing the Presentation of Multimedia Objects", *Computer Communications*, Vol. 15, No. 9, November 1992, pp. 557-564.
- [IDG94] M. Iino, Y.F. Day, A. Ghafoor. "An Object Oriented Model for Spatio-Temporal Synchronization of Multimedia Information". In Proc. IEEE Int. Conf. Multimedia Computing and Systems, Boston MA, 1994, pp. 110-119.
- [ISO92] International Standard Organization. "Hypermedia/Time-based Document Structuring Language (HyTime)", ISO/IEC IS 10744, April 1992.
- [ISO93] ISO/IEC. JTC 1/SC 29, "Coded Representation of Multimedia and Hypermedia Information Objects (MHEG)", Part I, Committee Draft 13522-1, June 1993. ISO/IEC 10031.

- [KAN94] W. Klas, K. Aberer, E. Neuhold. „Object-Oriented Modeling for Hypermedia Systems using the VODAK Modeling Language (VML)“. In A. Dogac, M.T. Özsu, A. Biliris, T. Sellis (editors). *Advances in Object-Oriented Database Systems*, volume 130 of NATO ASI Series F. Springer, Berlin, 1994.
- [LG93] T. Little, A. Ghafoor. “Interval-Based Conceptual Models for Time-Dependent Multimedia Data, *IEEE Transactions on Data and Knowledge Engineering*, Vol. 5, No. 4, August 93, pp. 551-563.
- [LR95] M. Löhr, T.C. Rakow. “Audio Support for an Object-Oriented Database Management System”. *Multimedia Systems Journal*, 3, 1995.
- [OS95] G. Ozsoyoglu, R. Snodgrass, “Temporal & Real Time databases: A Survey”, *IEEE-TDKE, Special Issue on Temporal Databases*, vol 7(4), Aug. 1995, pp 513-532.
- [RNL95] T. Rakow, E.J. Neuhold, M. Löhr. “Multimedia Database Systems - The Notions and the Issues”. In G. Lausen (editor). *Tagungsband GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*. Dresden, Springer Verlag, Informatik Aktuell, März 1995, pp. 1-29.
- [SD96] J. Schnepf, D.H-C Du, "Doing FLIPS: Flexible Interactive Presentation Synchronisation", *IEEE Journal on Selected Areas in Communications*, vol 14, No. 1, Jan. 1996, pp. 114-125.
- [SKD96] J. Schnepf, A. Kosntan, D.H. Du. “Doing FLIPS: FLexible Interactive Presentation Synchronisation”, *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 1, January 1996, pp. 114-125.
- [SW95] G. Schloss, M. Wynblatt. “Providing definition and temporal structure from multimedia data”, *Multimedia Systems Journal*, Vol. 3, 1995, pp. 264-277.
- [TK96] H. Thimm, W. Klas. “Playout Management in Multimedia Database Systems. In: K.C. Nwozu, P. Bruse Berra, B. Thuraisingham (editors). *Design and Implementation of Multimedia Database Management Systems*. Kluwer Academic Publishers, 1996.
- [TKWPC96] H. Thimm, Wolfgang Klas, Jonathan Walpole, Calton Pu, Crispin Cowan. “Managing Adaptive Presentation Executions in Distributed Multimedia Database Systems”. In: *Proceedings of International Workshop on Multimedia Database Systems*, Blue Mountain Lake, NY, 1995.
- [TR94] H. Thimm, T.C. Rakow. “A DBMS-based Multimedia Archiving Teleservice Incorporating Mail. In: W. Litwin and T. Risch (editors). *Proceedings of the First International Conference on Applications of Databases (ADB)*, Vadstena, Sweden, 1994. *Lectures Notes in Computer Science* 819, Springer Verlag, 1994, pp 281-298.
- [Vodak95] DIMSYS. „VODAK 4.0, User Manual“. Technical Report No. 910. GMD, St. Augustin, 1995.
- [VH93] M. Vazirgiannis, M. Hatzopoulos. “A Script Based Approach For Interactive Multimedia Applications”, in *Proc. of the 1st Multimedia Modelling Conference*, Singapore, 11/1993.
- [VS96] M. Vazirgiannis, T. Sellis. “Event and Action Representation and Composition for Multimedia Application Scenario Modelling”. *ERCIM Workshop on Interactive Distributed Multimedia Systems and Services*, BERLIN, 3/1996.
- [VTS96] M. Vazirgiannis, Y. Theodoridis, T. Sellis. “Spatio Temporal Composition in Multimedia Applications”. In: *Proceedings of the IEEE - ICSE '96 International Workshop on Multimedia Software Development - BERLIN*, 3/1996.
- [WR94] T. Wahl, K. Rothermel, “Representing time in multimedia systems”. In *Proceedings of the IEEE Int. Conference on Multimedia Computing and Systems*, Boston MA, pp. 538-543.
- [WRW95] S. Wirag, K. Rothermel, T. Wahl. “Modelling Interaction with HyTime”. In *Proceedings of the kivs 95*.
- [WWR95] T. Wahl., S. Wirag, K. Rothermel. “TIEMPO: Temporal Modelling and Authoring of Interactive Multimedia”. In *Proc. IEEE Int. Conference on Multimedia Computing and Systems*, Washington D.C., 1995, pp. 274-277.