

Specifying and Authoring Multimedia Scenarios

Michalis Vazirgiannis

Athens University of Economics and Business, Greece

Ioannis Kostalas and Timos Sellis

National Technical University of Athens, Greece

An authoring methodology and a set of checking tools let authors specify the spatial and temporal features of an application and verify the application prior to its execution. The checking tools include an animation tool, spatial and temporal layouts, and the execution table of the application.

A multimedia application involves a variety of individual media objects, called actors, presented according to the application's scenario. The term scenario covers two areas:

- the spatial and temporal ordering of actors within the application and the relationships among them, and
- the way that users will interact with the application and how the scenario will treat application or system events.

Actors participating in a multimedia application are usually transformed either spatially and/or temporally to meet the author's requirements. For instance, we may want to present part of a video clip faster or slower in a bigger or smaller window.

The authoring procedure for complex multimedia applications involving a large number of actors proves a very complicated task. Authors must keep in mind the large set of possible events that may be encountered in the application context, the number of actors and relationships, and the potential combinations of these factors.

The potential high complexity of multimedia applications demands substantial effort in designing and developing them. In real-life applications,

usually only programmers can develop multimedia application scenarios, since current authoring tools provide rather low-level specification languages. Moreover, these languages don't adequately describe the scenario aspects. Therefore, the lack of an integrated mechanism for a high-level, complete, multimedia application scenario specification proves problematic. Moreover, in current multimedia application development tools, authors mix the script with the application content (actors). This prevents the explicit reusability of scenarios in other multimedia applications with different content but similar functionality.

Checking multimedia application scenarios for integrity during development remains another important issue. The term checking in this context implies the various procedures that let authors review the result of their work prior to producing and executing the application. This enables revisiting the application and adjusting the spatiotemporal specification in order to align to the document style the authors had in mind, or fixing specification errors that result in spatial and/or temporal exceptions.

Figure 1 shows the multimedia application development procedures in terms of modules. You can see the two main phases, namely the specification and the checking phase. In the first, authors specify the transformation of the original multimedia objects in order to participate in the application. In the second phase, authors may verify several aspects of the application by obtaining spatial and temporal layouts or viewing an animation of it prior to execution.

In this article we present an authoring and checking methodology for developing multimedia application documents. The application design builds on a theoretical model for spatiotemporal compositions in the context of multimedia presentations.¹ The tool may be used for both prototyping and checking of multimedia presentations or for spatiotemporal compositions in general.

Regarding the authoring phase, we emphasize the flexible definition of spatial and temporal relationships of the participating entities. In other words, the authoring phase consists mainly of declarative specifications of the spatial and temporal ordering of multimedia objects based on their spatial and temporal relationships.

Multiple tools support the checking procedures by allowing designers to preview their applications in various ways, such as

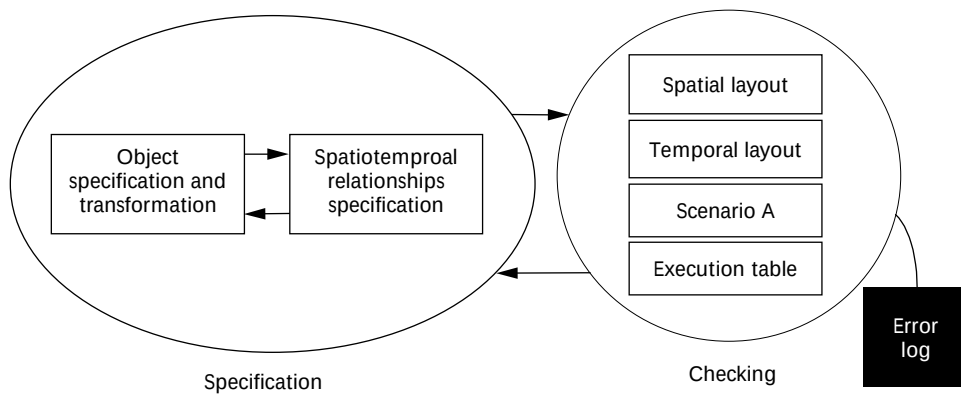


Figure 1. The authoring and checking cycle for a multimedia application.

- viewing the application window's spatial layouts,
- viewing the temporal layout of parts (or the whole application), indicating the temporal duration and relationships among the participating objects, and
- viewing the animation (rendering) of the application (that is, what the execution of the application looks like) in three modes (real time, manual, and snapshots of the application at regular temporal intervals).

Related work and background

Specifying a multimedia application essentially defines the composition of the participating media objects in space and time along with the appropriate transformations of the media object.

In the literature, confusion exists among the concepts of synchronization and temporal specification. Researchers use these concepts interchangeably. Existing multimedia document standards—Synchronized Multimedia Integration Language (SMIL), Multimedia and Hypermedia Experts Group (MHEG), and HyTime—propose varying approaches for modeling multimedia documents' structure and behavior. (For more information on SMIL, see the sidebar.) The proposed standards, however, don't completely specify spatial and temporal composition of actors. Moreover, they don't address the issues of storage, retrieval, execution, and sharing of application scripts. In addition, event modeling and composition schemes don't adequately cover the requirements for the variety of events that might occur in a multimedia application. Finally, no large-scale implementations of these standards exist to prove their credibility and usability.

Bulterman et al.² presented a graphical inter-

Synchronized Multimedia Integration Language

The upcoming standard SMIL merits special attention. Hypertext Markup Language (HTML) succeeded because attractive hypertext content could be created without requiring a sophisticated authoring tool. SMIL¹ aims at the same objective for synchronized hypermedia. It's an upcoming standard for presenting synchronized multimedia in a Web browser. SMIL integrates a set of independent multimedia objects into a synchronized multimedia presentation. A typical SMIL presentation has the following features:

- The presentation consists of several components accessible via a uniform resource locator (URL).
- The components include different media types, such as audio, video, image, or text.
- Presentations have interaction support in terms of simple events. This implies that the begin and end times of different components must be synchronized with events produced by internal objects or by external user interaction. Also, SMIL supports simple user interaction. The user can control the presentation by using control buttons known from video recorders, such as stop, fast-forward, and rewind. Additional functions are random access (the presentation can start anywhere), and slow motion (the presentation plays slower than at its original speed).
- Users can follow hyperlinks embedded in the presentation.
- Rich, high-level temporal composition features such as lip synchronization and expressive temporal relations (including parallel with a master, interruptions with the first ending element (par-min), wall clocks, unknown durations, and so on).

face for creating and playing SMIL documents (Grins) authoring and presentation environment. Grins can be used to create and present SMIL-compliant documents. The temporal and spatial structure model supported is identical to the one of SMIL, while a stand-alone rendering module (called playout engine) presents the document.

Commercial authoring tools for multimedia application development adopt mostly object-oriented authoring models.

In its current form, SMIL doesn't support relative spatial positioning and spatial embedding of media objects, an aspect extensively covered by our authoring approach. Moreover, the temporal synchronization model doesn't address the causality of temporal relationships. Plus, the interaction model is rather limited regarding the multitude of events that can occur in a presentation. A subset of the Allen relations (sequential, co-start/co-end, and equality in time relationships) represents the temporal composition. Then a temporal graph is constructed and the global set of constraints is solved by offering alternative durations for the objects and the whole presentation.

This approach lacks some features covered in others, such as spatial specifications and interaction. Rather, it verifies temporal features of the presentation instead of executing it.

Commercial authoring tools for multimedia application development such as Assymetrix' ToolBook or Macromedia's MacroMind Director adopt mostly object-oriented authoring models, providing high-level script languages. These tools don't adequately fulfill requirements regarding

- Declarative spatiotemporal composition schemes and modeling of the actor transformations.
- Event representation and composition. These tools support only a limited repertoire of events, but don't allow composition of events. They mostly manipulate events related to the mouse and to the actors' state (start, stop, active, idle, and so on).
- Database support for the application scripts (multimedia data and scripts are bundled together in most cases).

Many existing models for temporal composition of multimedia objects build on Allen's relations.³ However, these relations don't suit composition representations since they're descriptive—that is, they don't reflect causal dependency between intervals. More specifically, problems arise when trying to use these relations because they express relationships between intervals of fixed duration.⁴ Multimedia applications demand that a relationship doesn't change when the duration changes. The descriptive character of Allen's relations doesn't convey the cause and result in a relationship.

Duda and Keramane⁴ classified other models for multimedia composition representation in two categories: point-based and interval-based. In point-based models, the elementary units represent points in time and space. Each event in these models has its associated time point. The models arrange time points according to some relations such as "precedes," "simultaneous," or "after" to form complex multimedia presentations. An example of the point-based approach is a timeline. Interval-based models consider elementary media entities as time intervals ordered according to some relations. Existing models mainly build on the relations defined by Allen for expressing knowledge about time.

Duda and Keramane⁴ also introduced an interesting mechanism for temporal composition. They presented a model that considers the semantics of temporal relationships between objects. A set of operators is defined expressing the causal relations between intervals. Hirzalla, Falchunck, and Karmouch⁵ presented a temporal model for interactive scenarios. This model builds on the timeline approach and provides the primitives for specifying synchronous and asynchronous interactive as well as temporal multimedia compositions. The timeline approach extends to a tree of timelines. Each branch of the timelines represents the different scenarios that the user may select.

Other approaches use Allen's relations to specify a multimedia database schema. Little and Chafoor⁶ proposed an Object Composition Petri Nets (OCPN) model equivalent to Allen relationships. This approach, though, doesn't take into account the possible unknown durations of intervals. Thus, to prepare an instantiated presentation, the tree of interval relations must be traversed to set deadlines in the presentation schedule. Iino, Day, and Ghafoor⁷ developed a model for spatiotemporal multimedia presentations. The model handles temporal composition in terms of Allen relationships and treats spatial aspects in terms of

a set of operators for binary and unary operations. However, the model lacks the following features:

- No indication of the temporal causal relationships exists. For example, 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 (object A is to the right of object B) and the distance information (object A lies 10 cm away from object B).
- The modeling formalism is oriented towards executing and rendering the application rather than authoring it.

Yu and Xiang⁸ introduced the Hypermedia Presentation and Authoring System. In that effort, the temporal synchronization focuses on defining serial (objects one after the other) and parallel relationships (objects starting or stopping at the same time). The system considers each object to be active when presented on the screen and deactivated when removed from the screen. As for spatial synchronization, this approach defines an object's positioning in a rectangular area that may be scaled for the needs of the presentation. The system applies special constraints so that objects will not overlap during their presentation.

The approach doesn't handle the intermediate pause and resume actions common in a presentation of time-dependent objects. Nor does it handle the causality features of the temporal relationships (the end of video A causes the start of video B). Finally, the approach doesn't represent spatial relationships and embedded objects, and interaction handling is limited to anchors.

Kim and Song⁹ provided a temporal specification methodology that considers the temporal length of objects. It also recognizes that the presentation may vary between two values. Thus the concept of elastic time is introduced, which allows stretching or shrinking objects temporally.

Karmouch and Emery¹⁰ developed a model for multimedia applications that relate to multimedia documents. Thus, given that a document will be based on textual resources, the model tries to make an interactive multimedia book containing some form of multimedia objects like images, sound, and video. The book divides into chapters, and the

Many existing models for temporal composition of multimedia objects build on Allen's relations.

screen layout resembles word processors (along with their temporal information). The model takes into account temporal relationships, but not the spatial ones, since it assumes that they're solved depending on the text flow on the page.

Researchers have also studied the issue of scenario integrity checking. Courtiat and De Oliveria¹¹ presented a synchronization model for the formal description of multimedia documents. This model automatically translates the user formalization into a real-time LOTOS formal specification and verifies a multimedia document aiming to identify potential temporal inconsistencies. Described through a hierarchical model, multimedia documents allow incomplete timing. The model also represents user interaction and expresses a media object as one logical unit. The model provides a set of synchronization patterns, formal semantics, and a verification technique. In the proposed hierarchical model, start and end events are mandatory, while external and internal events are optional. The synchronization is facilitated by the names of the events or by explicit scripting. A presentation library provides monolithic and stream media objects. In addition, the model provides a constraint library. Constraints deal with temporal equalities like simultaneous events and precedence of a given event on another one or within a given temporal interval. Constraints also handle temporal inequalities such as the precedence of an event over another, an event within an unspecified amount of time, or an event limited by a minimum and maximum length. Constraints based on interval relationships (before, while, overlaps, and so on) and termination (Wait Latest, Wait Earliest, Wait Master) are also given.

Blakowski and Steinmetz¹² recognized an event-based representation of a multimedia scenario as one of the four categories for modeling a multimedia presentation. Events are represented in the Hypermedia/Time-Based Structuring

Language (HyTime) and Hypermedia Office Document Architecture (HyperODA). Events are defined in HyTime as presentations of media objects along with the playout specifications and finite coordinate system (FCS) coordinates. HyperODA events happen instantaneously and mainly correspond to start and end of media objects or timers.

All these approaches suffer from poor semantics conveyed by the events. Moreover, they don't provide any scheme for composition and consumption architectures.

You'll find an interesting survey on authoring models and approaches elsewhere.¹³

The composition model

The "action" concept relates to the presentation of actors (multimedia objects) participating in multimedia applications. An application specification should describe both temporal and spatial ordering of actors. We claim that the term synchronization is poor for multimedia applications and, instead, propose the term composition to represent both the temporal and spatial ordering of actors.

Temporal relationships

Allen has already addressed the topic of relations between temporal intervals.³ We exploit the temporal composition scheme defined in our earlier work.¹⁴ Here we briefly introduce that scheme for representing the temporal composition of multimedia objects that also captures the causality of the temporal relationships. In our scheme we exploit the start and end points of a multimedia instance as events. The end of a multimedia object presentation is either natural (when the media objects finishes its execution) or forced (when an event explicitly stops the execution of a multimedia object). Moreover, we also take into account the well-known pause and resume procedures.

We consider temporal instance an important concept because it serves as an arbitrary temporal measurement that's relative to some reference point, the application temporal starting point in our case, hereafter Θ .

Based on the above descriptions we define the following operators attached to the corresponding events:

Definition: Let A be a multimedia instance, then $A>$ represents the start of the multimedia instance, $A<$ the natural end of the instance, $A!$ the forced

stop, $A||$ the pause, and $A|>$ the resume actions.

Definition: Let A, B be two multimedia instances. Then the expression $Aop1\ t\ Bop2$ represents all the temporal relationships between the two multimedia instances, where $op1 \in \{>, <, ||, |>\}$, $op2 \in \{>, !, ||, |>\}$, and t is a vacant temporal interval.

Definition: Let A be a multimedia instance. We define as t_{Aop} the temporal instances corresponding to the events Aop , where $op \in \{>, <, !, ||, |>\}$.

Definition: Let A be a multimedia instance. We define as d_A the temporal duration of the multimedia instance A .

Spatial relationships

Another aspect of composition of multimedia objects in multimedia applications relates to the spatial layout of the application—the spatial arrangement and relationships of the participating objects. The spatial composition aims to represent three aspects:

- topological relationships between the objects (disjoint, meet, overlap, and so on),
- directional relationships between the objects (left, right, above, above-left, and so on), and
- distance characteristics between the objects (outside 5cm, inside 2cm, and so on).

Spatiotemporal composition model

Here we briefly present the theoretical foundations of our spatiotemporal checking framework. The model we propose will translate spatiotemporal relationships among multimedia objects into minimal and uniform expressions, as imposed by the requirements for correct and complete representations.

For uniformity reasons, we define an object named Θ that corresponds to the spatial and temporal start of the application (that is, the upper left corner of the application window and the temporal start of the application). We also assume that the objects included in the composition include their spatiotemporal presentation characteristics such as size, temporal duration, and so on.

Definition: Assuming two spatial objects A, B , we define the generalized spatial relationship between these objects as: $S_R = (x_{ij}, v_i, v_j, x, y)$

where r_{ij} is the identifier of the topological-directional relationship between A and B, v_i, v_j represent the closest vertices of A and B respectively and x, y are the horizontal and vertical distances between v_i, v_j .

Hereafter, we define a generalized operator expression to cover the spatial and temporal relationships between objects. Note that in some cases we don't need to model a relationship between two objects, we need to declare the spatial and/or temporal position of an object relative to the application's spatial and temporal start point Θ . For example, object A appears at the spatial coordinates (110, 200) on the tenth second of the application.

Definition: We define a composite spatiotemporal operator that represents absolute spatial/temporal coordinates or spatiotemporal relationships between objects in the application, **ST_R(sp_rel, temp_rel)**, where **sp_rel** is a spatial relationship (**S_R**) as defined above, and **temp_rel** is a temporal relationship.

The spatiotemporal composition of a multimedia application consists of several independent fundamental compositions. The term independent implies that actors participating in them aren't related explicitly (either spatially or temporally) except from their implicit relationship to the start point Θ . Thus, all compositions are explicitly related to Θ . We call these compositions **composition_tuples**, and these include spatially and/or temporally related objects.

Definition: We define the **composition_tuple** in the context of a multimedia application as **composition_tuple = A_i [{ ST_R A_j }]**, where **A_i, A_j** are objects participating in the application and **ST_R** is a spatiotemporal relationship (as defined above).

Definition: We define the composition of multimedia objects as a set of **composition_tuples**: **composition = C_i {C_j}**, where **C_i, C_j** are **composition_tuples**.

The Extended Backus Normal Form (EBNF) definition of the spatiotemporal composition based on the above definition follows:

```
composition ==:
    composition_tuple
```

```
{[,composition_tuple]}
composition_tuple ==:
    Θ {[,spatio_temporal_
        relationship actor |
        composition]}
spatio_temporal_relationship ==:
    "[("[spatial_operator |
        spatial_instance")" ,
    ("temporal_operator |
        temporal_instance")]"

temporal_operator ==: Θ | t_event
                    t_interval
                    TAC_operation
t_event ==: ">" | "<" | "!" | "|>" |
            "||"

spatial_operator ==:
    (rij, vi, vj, x, y)
x ==: INTEGER
y ==: INTEGER
```

where **rij** denotes a topological-directional relationship between two objects and **vi, vj** denote the closest vertices of the two objects (see definition above).

A sample multimedia composition

Here we describe a multimedia application corresponding to a TV news clip in terms of spatiotemporal relationships as defined above. The high-level scenario of the application follows:

“The News clip starts with presentation of image A (located at point 50,50 relative to the application origin Θ). At the same time, background music E starts. Ten seconds later video clip B starts. It appears to the right side (18 cm) and below the upper side of A (12 cm). Just after the end of B, another multimedia application related to fashion (Fashion_clip) starts. Fashion_clip consists of a video clip C that shows the highlights of a fashion show and appears 7 cm below (and left aligned to) the position of B. Three seconds after the start of C, a text logo D (the designer's logo) appears inside C, 8 cm above the bottom side of C, aligned to the right side. D will remain for 4 seconds on the screen. Meanwhile, at the tenth second of the News clip, the TV channel logo (F) appears at the bottom-left corner of the application window. F disappears after 3 seconds. The application ends when music background E ends.”

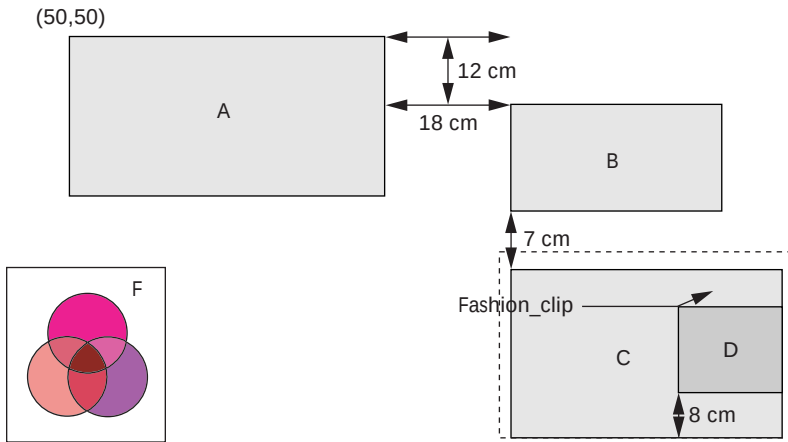
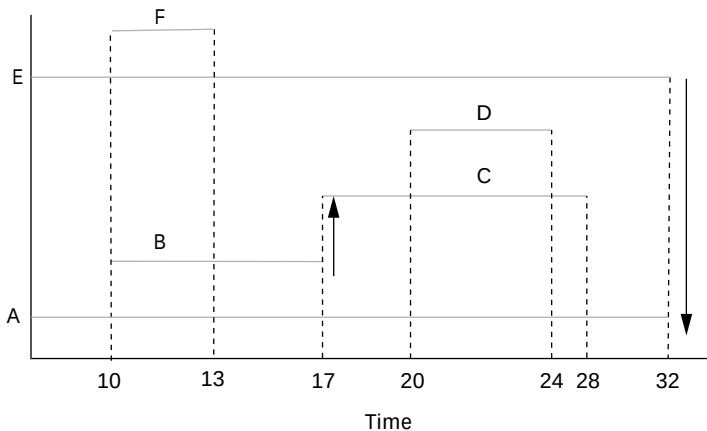


Figure 2. The spatial composition of the News multimedia application.

Figure 3. The temporal composition of the News multimedia application.



The spatial composition (screen layout) appears in Figure 2, while the temporal one appears in Figure 3. The objects to include in a composition tuple of a multimedia application are those spatially and/or temporally related. In our example (News clip), A and B and Fashion_clip should be in the same composition tuple, since A relates to B and B relates to Fashion_clip. On the other hand, F doesn't relate to any other object, neither spatially nor temporally, so it composes a different tuple.

We transformed the above spatial and temporal specifications defined in a high-level graphical user interface (GUI) into the following representation according to the model primitives defined previously:

```
// News Clip
composition = {r1, r2}
r1 =  $\Theta$  [(_, _, _, _), (>0>)]
E [(_, _, _, _), (<0!)]
```

```
News
r2 =  $\Theta$  [(r1,1, _, v2, 5, 5), (>0>)]
A [(r11,13, v3, v2, 18, 12),
(>10>)]
B [(r13,6, v1, v2, 0, -7),
(>0>)]
Fashion_clip
r3 =  $\Theta$  [(_, _, v1, 0, 300),
(>10>)]
F
// Fashion clip
composition = {r4}
r4 =  $\Theta$  [(_, _, v2, 0, 0), (>0>)]
C [(r9,10, v4, v4, 0, 8), (>3>)]
D
```

Expressive power of the model

In this section we evaluate the expressiveness of the proposed authoring model.

The model can represent in an integrated way the spatial and temporal composition of objects. A spatial/temporal composition is essentially a chain of objects where each one relates to the previous one with a spatial and/or temporal relationship, which explicitly reflects the author's composition requirements. The same application may contain several composition tuples with each one having as the first object the spatiotemporal origin of the document (Θ). A composition tuple only includes objects explicitly related by the author. Thus objects that belong to the same application document but not explicitly related belong to different composition tuples.

The cornerstone of the model's expressiveness is that spatial/temporal relationships among participating objects are explicitly represented in a declarative way.

As for interaction, in this model we deal only with a multimedia application's internal interaction (for example, the end of object A triggers the start of object B). The issue of user interaction is not handled in this effort, though this model has served as the basis for a scenario model that handles rich internal and external interactions at the modeling,¹⁵ authoring,¹⁶ and rendering¹⁷ levels.

In some cases objects have uncertain starting or ending points (that is, partially unspecified durations). Our model handles cases like this to the degree that the starting or ending point depends on the start/end points of other objects. Appropriate algorithms compute the exact absolute temporal points, relative to the application's temporal starting point, based on the relative temporal/spatial positioning.

Another issue we can address here is the aggregation of more than one object in a self-standing presentation unit. Our model allows aggregating several objects in a composition tuple. Moreover, relations between aggregations can be expressed, since composition tuples are allowed as members in other composition tuples. This is illustrated in composition tuple π_2 in the above example.

An important issue in such a complex spatial and temporal composition is that inconsistencies may arise. We deal with this problem at the object and application levels, which we'll describe later. However, the consistency checking problems are much harder to tackle due to the indeterminate nature of interactions. (We addressed this problem in other research efforts.¹⁰) Currently, we're addressing the integrity checking in the temporal domain, but such checking in the spatial domain remains an issue for further research.

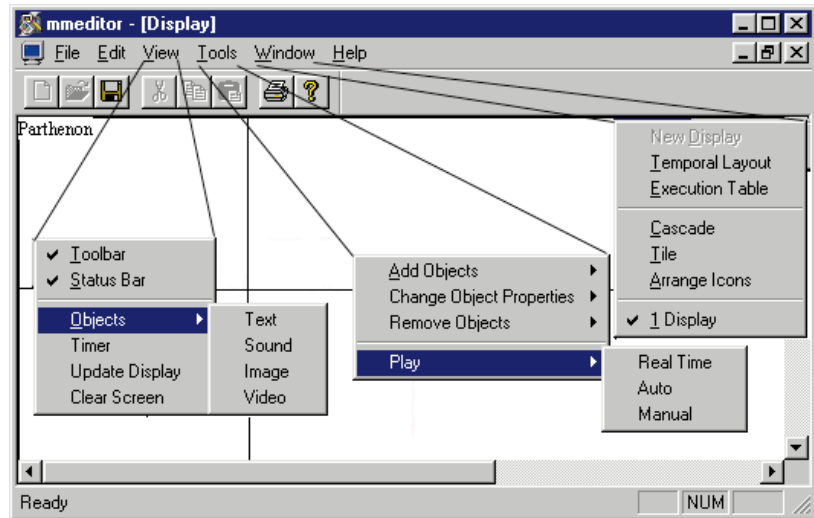
Authoring compositions

Our authoring methodology builds on the model introduced in the previous section. A multimedia application document contains objects composed in space and time according to a set of spatial and temporal relationships. Authors build scenarios in a stepwise procedure. The first step requires specifying the objects that participate in the application along with their spatial and temporal transformations. The second step defines the actors' spatial and temporal position in the application document in terms of absolute or relative coordinates. The authoring tool transforms these specifications and produces the spatiotemporal scenario—that is, when, where, and for how long each object will be presented.

We distinguish the actors in four main categories: text, sound, image, and video. We assume that each object (except sound) has some spatial extent so it can be represented by a rectangle in which the image, text, and/or video information is presented. The sound objects have only temporal aspects.

Authoring environment

The authoring interface supports the multimedia scenario's visual definition. Conceptually, the scenario specification should start at the temporal start of the application. Thus, the authoring procedure starts at the beginning of the application (time = 0) and specifies the participating actors in increasing temporal order. Figure 4 gives an overview of the working environment.



Composition specifications. Initially authors may insert or remove media objects from the actor's list. Each actor object can be further defined by assigning values to its spatial and temporal attributes.

Each actor includes identification data such as name, media type and media file corresponding to the actor, and the object's temporal and spatial coordinates in the application. An actor's name must be a unique name in the application context. Though the tools allow defining different actors based on the same media object, the data that authors enter (external) are transformed into absolute coordinates (internal) to produce the scenario. The external data mostly relate the spatial and temporal coordinates of the new object to those of other objects.

Our methodology supports incremental authoring by adding new actors to the existing spatiotemporal composition and relating it to them spatially and/or temporally.

Temporal attributes specification. An object's temporal attributes include its temporal start and end points. The author may define the beginning of an object in relation to another object that's already, or will be, active. Thus the starting time relates to the temporal application origin Θ or to the start/end point of another object (in this case, we don't distinguish between the pause and stop events or between the start and resume event). The same applies to the definition of the end time of the object as long as the object doesn't have an internal (natural) duration (like video and sound).

If the object has a natural length, then the user can either let it end naturally or force it to end

Figure 4. The authoring environment menu options.

before its natural end. That event can be a stop or a pause event. As mentioned before, the author may define the actor's temporal data in relation to the starting/ending point of another actor. We must distinguish here between the event's pause and restart in our design. The pause event represents a temporary stop event, whereas a restart event serves as a start event.

Spatial attributes specification. The multimedia literature has hardly addressed the issue of spatial features of actors and their composition in multimedia applications. Only recently have some research efforts studied this issue.^{7,14} Our authoring tools let authors specify an actor's spatial features either as absolute coordinates or in relation to other objects. We assume that a rectangle bounds each object (whose dimensions the author may change) and that the author defines its position. The actor's upper left corner relates either to the origin of the application (Θ) or to any vertice (upper left, upper right, lower left, lower right) of any other actor already defined. Note that it's possible to relate the actor under concern to different actors as regards to the X and Y axis. For absolute coordinates we define the position of the actor's upper left corner, which relates to the top left corner of the application window.

Initial integrity checks

At this point a set of initial integrity checks can be carried out. A list of nodes represents the results of the authoring procedure regarding internal representation of the data. It contains for each participating object all the defined attributes (described above). The nodes contain information such as

- Relative spatial and temporal position data, relating the object to others participating in the multimedia application.
- Absolute coordinates (spatial and temporal), when available. The absolute data are the spatial/temporal distances of the object from Θ (the temporal and spatial start of the application).

The consistency is syntactic and semantic, defined at the object (actor) and document (application) level.

An object is syntactically correct if

- it contains a multimedia application unique name,

- the necessary spatial and temporal coordinates (absolute or relative) are given, and
- in the case of relative placement, all referenced actors still exist in the application (since actors can be deleted).

The check at the object level occurs when authors define a new object or modify an existing one.

Semantic checks include the following criteria:

- an object's start time must be before its end time,
- an object's spatial extent must be within the limits of the specified display window (violations reported as warnings), and
- an object's natural duration must not be less than the duration resulting from the objects temporal specifications (violations reported as warnings).

Syntactic and semantic integrity is ensured at the application level if

- all the participating objects are consistent (according to the above definitions) and
- no temporal overlaps exist among objects including sound (video or sound object).

Navigation and querying tools

During the authoring procedure authors might want to query the scenario, especially if it's extended and complicated in terms of presentation actions. This can help

- inform the authors about the scenario's underlying spatiotemporal constraints and
- modify the scenario and create, correct, or improve the scenario incrementally.

In the last case, the author, depending on the answer of the query, can modify the scenario currently under creation. The queries divide into the following categories:

- *Point queries.* Finds relationships between events such as "Does actor Starting Logo start before Tour video?" or "Which objects appear at the position (50, 50) before the fifth second of the application?"

- *Relationship queries.* Finds relationships among the temporal intervals that represent the presentations of actors³ or alternatively, the relationships between the spatial extents of the actors. Such queries would include “Are the presentations of objects ‘tour_video’ and ‘agora’ simultaneous?” or “Does ‘tour_video’ overlap with ‘agora’?”
- *Layout queries.* Finds the spatial and temporal relationships among presented objects. For example, “Show the temporal layout of the application between the second and tenth second of the presentation.”

To respond to the above requirements requires transforming the spatial/temporal content of the multimedia application into convenient memory resident structures (such as sorted lists of actors, where the sorting is based on space and time).

Sorting the participating objects

To carry out the aforementioned checks requires placing the application (actors) content in absolute temporal and spatial coordinates.

We designed a tool that targets preorchestrated multimedia applications with minimum user-program interactivity. Thus for each object, the application’s absolute spatial and temporal coordinates can be computed. The tool transforms relative spatial and temporal data to absolute values and creates the final scenario. If authors request changes and alter the relative data, then the transformation process takes places again. An object participating in a multimedia application has, roughly, the following structure:

```
Object
  // absolute positioning
  Sp_coordinates
sp_absolute_data
  Time start_time_absolute
  Time end_time_absolute
  // relative spatial positioning
  Object* sp_relative_obj
  Sp_coordinates
sp_relative_data
  // relative temporal positioning
  Object*
temp_relative_obj
  Time start_time_rel
  Time end_time_rel
  ...
end
```

The attributes `sp_absolute_data`, `start_time_absolute`, and `stop_time_absolute` represent the absolute coordinates. The terms `rel_absolute_data`, `start_time_rel`, and `stop_time_rel` represent the object’s *relative_data*. When authors specify an actor’s position, the values provided are always attributed to the relative members (`sp_relative_data`, `start_time_rel`, and `end_time_rel`). Then the tool can transform the absolute coordinates and update the appropriate members (`sp_absolute_data`, `start_time_absolute`, and `end_time_absolute`).

To simplify the algorithm description, we assume that an object in a composition tuple relates at most to one object in time (`temp_relative_obj`) and to one object in space (`sp_relative_obj`).

The objects reside in a list structure (`object_list`), which serves as the basic structure for sorting them and answering queries. Then, the algorithm in Figure 5 (next page) transforms the relative data into absolute coordinates.

After applying the algorithm in Figure 5 to the `object_list`, every object “knows” its absolute spatial and temporal coordinates. This makes it easy to generate the list in any spatial or temporal order.

Temporal layout tool

The first checking tool is the temporal layout that displays, graphically, the temporal order and the duration of the actors (see Figure 6, next page). This facility gives an overview of the temporal configuration of the multimedia application and provides support to queries of the type “Which objects are active at a specific time?” or “Which objects are active at a specific period of time?” (that is, during the temporal interval in which another object is active). The temporal layout may refer to a part of the application’s temporal duration or to its total duration. The layout sorts the list of objects (in absolute temporal coordinates) according to their starting time. The other checking tools—the execution table and the spatial layout—use this list.

The temporal layout uses the absolute data of the objects resulting from the actor-sorting algorithm (Figure 5). However, the objects must be listed based on their absolute start time. The following bubble-sort algorithm sorts this list:

```
if object_list has one object then
  sorted
else
  goto second object
```

```

If (object_list changed due to object addition)
  and (rest of object_list is ordered)
  do transform_new_data(object_list)
else
  for every object o in object_list
    initialize o.absolute data to (-1)

object_list transform_new_data(object_list)
//for temporal data
  for every object o in object_list
    if o.temp_relative_obj is  $\Theta$ 
      copy o.start_time_rel to o.start_time_absolute
    else
      find o.temp_relative_object
      if o.temp_relative_object.temp_relative_obj is  $\Theta$ 
        compute o.start_time_absolute
      else recursive call until  $\Theta$  is reached

  if o.end_time_rel is a natural end
    (where permitted)
    o.end_time_absolute = o.start_time_absolute +
      o.duration
  else
    find o.temp_relative_object
    if o.temp_relative_object.temp_relative_obj is  $\Theta$ 
      compute o.end_time_absolute
    else recursive call until  $\Theta$  is reached
//for spatial data
  if o.sp_relative_obj is  $\Theta$ 
    copy o.sp_relative_data to o.sp_absolute_data
  else
    find o.sp_relative_object
    if o.sp_relative_object.sp_relative_obj is  $\Theta$ 
      compute o.sp_absolute_data
    else recursive call until  $\Theta$  is reached
// check the results
  if for all objects (absolute data are set)
    or (no object set at this repetition)
    stop and notify for errors (find objects not set)
  else
    do transform new data(object_list)
    until (all set) or (none set) or (number of
      repetitions = number of objects)

```

Figure 5. Algorithm for transforming relative data into absolute coordinates.

```

sort list:
  if this object starts after the
    previous
    goto next
  else
    move it back until it does
    goto next object
do sort list

```

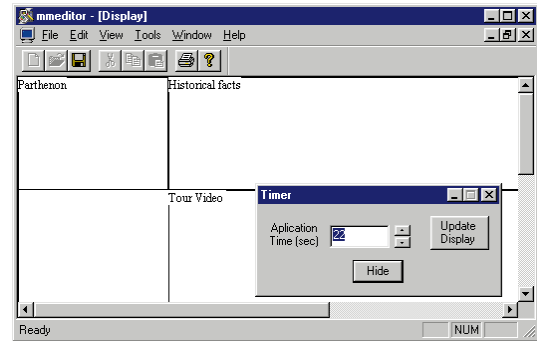


Figure 6. The temporal layout of a multimedia application.

Spatial layout

The term spatial layout describes the appearance of the multimedia application window, conveying information about the position and dimensions of the actors participating in the application. Authors must be able to preview the application's spatiotemporal layout at any time during development, so they can modify it accordingly. The spatial layout tool lets authors view how the application window will look at any temporal point during a potential application execution. Each object's temporal duration derives from the temporal layout.

Authors may set the desired time point (see Figure 7) through the timer, then view the screen's layout with the update display option. Thus authors can check the display before inserting a new object and find out where it should be placed.

The spatial layout tool uses a list of objects. It checks for each of them if the desired presentation time occurs between the object's absolute start and end times, and determines whether to display the object.

Execution table

Often, authors must have an overview of the application's structure in ascending temporal order. The execution table tool fulfills this need. This table—generated at any time during the authoring session—includes the temporal and spatial coordinates of each start and stop event in the application (see Figure 8). The table lists contents in ascending temporal order (that is, the start or stop events for actors along with their position and size). The execution table contains the appropriate data resulting from the sorted list of objects. These objects can be sorted again depending on the time that each event occurs.

The sorted list of objects can be used again and

further sorted depending on the time that each event occurs. The resulting table is supported by a C++ structure that contains strings, integers, and pointers to objects in its tuples.

Scenario animation tool

The checking tool lets authors render (animate) the application, giving them an animated view of an application’s spatiotemporal specifications. The play tool accomplishes this. In this section we use the terms rendering and animation interchangeably. The animation simulates an application execution session using consecutive snapshots of the application updated at regular time intervals, which authors can change. This kind of animation lets authors notice any mistakes or misplacements to fix them later.

Authors can change the time limits of the animation and animate only a temporal part of the application (such as for the fifth to the twenty-fifth second). The animation may be interrupted at any time, letting authors make changes in the scenario. They can then resume the animation to verify the changes.

Rendering of the multimedia application relies on the following structures:

- The sorted list of the multimedia objects
- The execution table
- The list that contains the active objects for a specific time

The specific time mentioned is the time of the timer. This is an absolute offset from the application’s starting point. Whenever the timer changes, it updates the active object list to include the correct objects by deleting the objects that finish and adding the ones that start.

Before rendering the application or even obtaining a preview of a specific snapshot, the multimedia editor has to sort the object list and create or update the execution table. When the system performs these two actions, the editor can find the objects active at any given time in the application.

If the timer is set to 0 (the starting point), the editor can access any other time slot of the application by scanning the execution table to the time

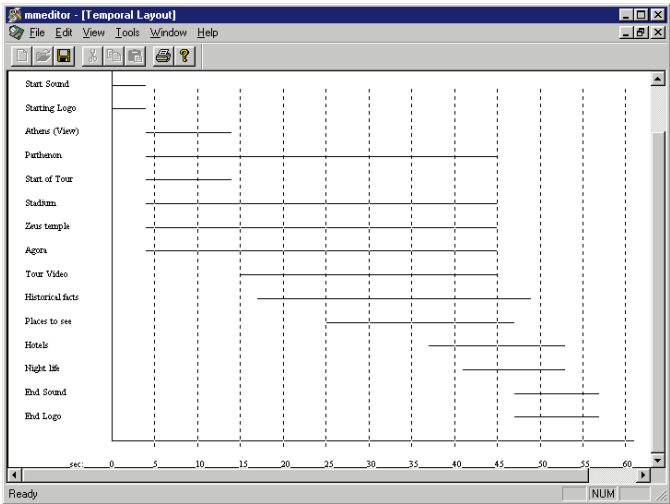


Figure 7. The spatial layout of a multimedia application at any time using the timer tool.

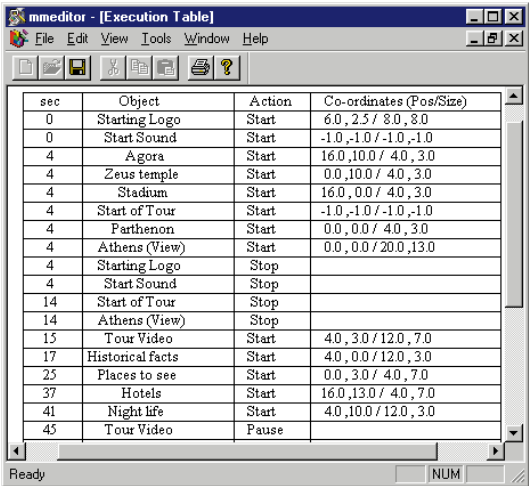


Figure 8. The execution table, which displays presentation actions in temporal ascending order and includes spatial information (position and size).

requested. During the scan, it adds to the active object list all the objects that become active and removes all the ones that become inactive. Because of the possibility of having many objects in the application, the editor just adds references of the objects to the list. The actual objects are stored only once and can be accessed through the object list.

Most of the time the global application time isn’t set to 0, but to some other time. To get to the requested time, the editor can elect to either start from the beginning of the execution table and get to the specific time or move from the current position. The decision is based on the distance of the requested time from the two points (start_time, current).

If the starting point is the closest, then the algorithm mentioned above is used. If the current position is the closest, then the execution table is scanned backwards if the requested time precedes

the current time or forward if it follows. For the forward scan nothing changes from the above algorithm but the fact that the list may not be empty. For the backward scan the objects reported to stop are added to the list and the ones reported to start are removed until the time is reached.

The algorithm for the rendering stays exactly the same. In fact, the same functions perform the rendering. The only difference is that the global time always increases by one and the current time is always closer (it's one time slice back). To render a specific part of the application requires setting the global time to the beginning of the part and updating the active object list. Then a system timer takes care of moving the global time to the next time slot according to the interval set. For each move the active object list is updated and its contents displayed.

The application becomes persistent by saving it to a binary format file that's interpreted only by the tool. Another type of output is the script—adopted for its compatibility among authoring tools. It contains the declarative script representing the spatiotemporal relationships among the participating actors as the user has defined them.

Conclusions

In this article we presented an implemented authoring and checking methodology for developing multimedia application documents. The application design builds on a theoretical model for spatiotemporal compositions in the context of multimedia presentations.¹⁴ The tool can be used for both prototyping and checking of multimedia presentations or spatiotemporal compositions in general.

The advantages of our tools regarding authoring include

- Declarative and visual authoring methodology
- Integrated spatial/temporal application specification
- Relative actor positioning in spatial and temporal domains

The most important advantage of our design is that the authoring and the checking process are well integrated and interleaved. Thus authors can verify their specifications within the authoring procedure and environment.

The tool presented above can be extended in the following directions:

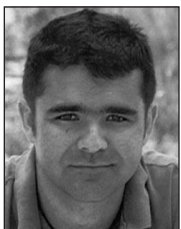
- *Connection to other authoring tools.* It's feasible to export the specification to various authoring tools scripts such as Toolbook, Mac/Director, Hytime, and so on. This would require translating the tool's proprietary script into the authoring tool script language and specifying the mappings.
- *Checking scenarios specified from other tools.* This would involve an import scheme so that our tool could import external tool scripts. Then the application could be animated and verified in relation to the spatial and temporal features.

The second direction seems especially fruitful since authors are producing more interactive multimedia content for Web pages and SMIL appears promising. Therefore, we aim to transform SMIL documents in our script language and use our tools to check them. MM

References

1. *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification* (Proposed Recommendation), World Wide Web Consortium (W3C), <http://www.w3.org/TR/REC-smil/>.
2. D. Bulterman et al., "Grins: A Graphical Interface for Creating and Playing SMIL Documents," *Proc. of Seventh Int'l World Wide Web Conf. (WWW7)*, Elsevier Science, New York, April 1998.
3. J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. of the ACM*, Vol. 26, No. 11, Nov. 1983, pp. 832-843.
4. A. Duda and C. Keramane, "Structured Temporal Composition of Multimedia Data," *Proc. IEEE Int'l Workshop on Multimedia Database Management Systems*, IEEE CS Press, Los Alamitos, Calif., Aug. 1995.
5. N. Hirzalla, B. Falchuk, and A. Karmouch, "A Temporal Model for Interactive Multimedia Scenarios," *IEEE MultiMedia*, Vol. 2, No. 3, Fall 1995, pp. 24-31.
6. T. Little and A. Ghafoor, "Interval-Based Conceptual Models for Time Dependent Multimedia Data," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 5, No. 4, Aug. 1993, pp. 551-563.
7. M. Iino, Y.F. Day, and A. Ghafoor, "An Object-Oriented Model for Spatiotemporal Synchronization of Multimedia Information," *Proc. of the IEEE Multimedia Conf.*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 110-119.
8. J. Yu and Y. Xiang, "Hypermedia Presentation and Authoring System," *Proc. of the 6th Int'l World Wide Web Conf.*, Elsevier Science, New York, April 1997, pp. 153-164.

9. M.Y. Kim and J. Song, "Multimedia Documents with Elastic Time," *Proc. of ACM Multimedia 95*, ACM Press, New York, Nov. 1995, pp. 143-154.
10. A. Karmouch and J. Emery, "A Playback Schedule Model for Multimedia Documents," *IEEE MultiMedia*, Vol. 3, No. 1, Winter 1996, pp. 50-63.
11. J.P. Courtiat and R.C. De Oliveira, "Proving Temporal Consistency in a New Multimedia Synchronization Model," *Proc. of ACM Multimedia 1996 Conf.*, ACM Press, New York, 1996, pp. 141-152.
12. G. Blakowski and R. Steinmetz, "A Media Synchronization Survey: Reference Model, Specification, and Case Studies," *IEEE J. on Selected Areas in Comm.*, Vol. 14, No. 1, Jan. 1996, pp. 5-35.
13. M. Jourdan, N. Layaïda, and C. Roisin, "A Survey on Authoring Techniques for Temporal Scenarios of Multimedia Documents," *Handbook of Internet and Multimedia Systems and Applications, Part 1: Tools and Standards*, CRC Press, London, April 1998.
14. M. Vazirgiannis, Y. Theodoridis, and T. Sellis, "Spatiotemporal Composition and Indexing for Large Multimedia Applications," *ACM/Springer-Verlag Multimedia Systems J.*, Vol. 6, No. 4, 1998, pp. 285-298.
15. M. Vazirgiannis and S. Boll, "Events in Interactive Multimedia Applications: Modeling and Implementation Design," *Proc. of IEEE Int'l Conf. on Multimedia Computing and Systems (ICMCS 97)*, IEEE CS Press, Los Alamitos, Calif., June 1997, pp. 244-251.
16. M. Vazirgiannis et al., "i-Muse—Interactive Multimedia Scenario Editor," *IEEE Proc. of the 4th Int'l Workshop on Multimedia Database Management System (MMDBMS 98)*, IEEE Press, Piscataway, N.J., pp. 145-152.
17. D. Tsirikos et al., "A Client-Server Design for Interactive Multimedia Documents Based on Java," *Proc. of Interactive Distributed Multimedia Systems and Services Workshop (IDMS 98)*, Springer-Verlag, Berlin, Sept. 1998, pp. 248-259.
18. I. Mirbel, B. Pernici, and M. Vazirgiannis, "Temporal Integrity Constraints in Interactive Multimedia Documents," *Proc. of IEEE Int'l Conf. on Multimedia Computing and Systems (ICMCS 99)*, IEEE Press, Piscataway, N.J., June 1999, pp. 867-871.



Michalis Vazirgiannis is a lecturer in the Department of Informatics at Athens University of Economics and Business. His research interests and work range from interactive multimedia information systems

to spatiotemporal databases, uncertainty, and data min-

ing. He holds a degree in physics (1986) and an MS in robotics (1988), both from University of Athens, Greece. He also earned an MS in knowledge-based systems from Heriot Watt University, Edinburgh, UK. He acquired a PhD in object-oriented modeling of hypermedia information networks in 1994 from the Department of Informatics at the University of Athens, Greece. Twice he has received postdoctoral fellowships from the TMR Chorochronos project (an European Union-funded research network for spatiotemporal databases). He is a member of ACM and IEEE.



Ioannis Kostalas received a BS in electrical and computer engineering in 1996 from the National Technical University of Athens, Athens, Greece. In 1997 he was accepted as a graduate student to

the same university in the field of signal/natural language processing. His interests include database systems, multimedia, and e-commerce. He has worked on the development of linguistic software using databases and large-scale applications for enterprise-level e-commerce.



Timos Sellis is a full professor in the Computer Science Division of the National Technical University of Athens (NTUA), Athens, Greece. He is also the head of the Knowledge and Database Systems

Laboratory at NTUA. His research interests include extended relational database systems; data warehouses; and spatial, image, and multimedia database systems. He received a BS in electrical engineering in 1982 from NTUA, Athens, Greece. He received an MS from Harvard University in 1983 and a PhD from the University of California at Berkeley in 1986, both in computer science. He received the Presidential Young Investigator award for 1990-1995 and the Very Large Database (VLDB) 1997 10-Year Paper Award. He is a member of the editorial boards of *International Journal on Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies*, and *Geoinformatica*.

Readers may contact Vazirgiannis at the Dept. of Informatics, Athens University of Economics and Business, Patision 76, 10434, Athens, Greece, e-mail mvazirg@aueb.gr.