Multimedia Tools and Applications, 18, 213–231, 2002

Dealing with Uncertain Durations in Synchronized Multimedia Presentations

NABIL LAYAÏDA Nabil.Layaida@inrialpes.fr
LOAY SABRY-ISMAIL Loay.Sabry@inrialpes.fr
CÉCILE ROISIN Cecile.Roisin@inrialpes.fr
Opéra Project, INRIA Rhône-Alpes, 655 Avenue de l'Europe, F-38330 Montbonnot Saint-Martin

Abstract. In this paper, we discuss the effect of the uncertainty in the duration of some multimedia objects on the quality of the presentation of multimedia scenarios. This uncertainty can be due to external factors such as the access delay over internet or the user interaction. An uncertain duration is often followed by a period of desynchronization during which the presentation deviates from the desired scenario. To solve this problem, we present in this paper a solution that integrates two complementary approaches which reduce the desynchronization and limit its propagation in the rest of the presentation. These approaches are non-blocking and use the flexibility in the duration of multimedia objects in order to resynchronize the presentation as rapidly as possible.

Keywords: multimedia, scheduling, synchronization, uncertain durations, dynamic formatting

1. Introduction

A multimedia presentation is a set of media objects rendered over a pre-established temporal scenario. The different objects are presented according to predefined durations set at the authoring stage [8]. This scenario is contained in a multimedia document defined by an author and remotely accessed by users spread over the internet. In such an environment, some objects behave in an indeterministic manner and their duration does not necessarily match the predefined values. Audio and video streams suffering from remote access delays are common examples of such objects. These delays can propagate in the scenario affecting the global synchronization and the presentation quality.

There are two ways for dealing with the presentation's indeterminism. In the first case, each time the presentation gets out of synchronization, some actions are taken immediately to bring the scenario back to the predefined case. Usually this operation is achieved at the cost of either blocking some of the objects, skipping the content of some objects or delaying some others. In most cases, the author specifications are violated. In the second case, the rendering engine attempts first to determine if these delays can be taken into account while maintaining or at least remaining close to the scenario specified by the author. The later method requires some means to adjust the scenario according to the author's intent. The adjustment can be carried out by modifying the durations of some objects having some temporal flexibility [8]. These modifications must be achieved in a non blocking manner that avoids the desynchronization of the presentation. Notice that event-based synchronization, such as "when A finishes, B must be stopped," do not require handling object durations and are therefore not considered hereafter.

In this paper, we propose an intelligent scheduler which allows rendering of a multimedia presentation while handling the indeterministic behavior of objects. It was developed within the Madeus authoring and presentation environment [9] to handle scenario-based documents. The proposed scheduler is particularly suited for declarative document formats like SMIL [18] accessed through the network. Our solution combines two approaches: the dynamic formating and the reparation of indeterminism. The dynamic formatting approach operates at a given instant ahead of time to achieve the resynchronization. While the reparation approach is applied in case of failure or partial success of the dynamix formatter, it allows the effect of desynchronization to be minimized.

The paper is organized as follows: firstly, we present the related work in Section 2 and some basic concepts of temporal scenario in Section 3. In Section 4, we identify the consequences induced by the presence of indeterministic objects on multimedia presentations. Section 5 is devoted to the presentation of the two complementary techniques that we defined to manage the indeterminism: the *dynamic formatting* and the *reparation of indeterminism*. Finally, we conclude and give our perspectives for the research in this domain.

2. Related work

Research in the field of scenario-based scheduling covers two inter-related aspects. First, the design of verification tools which allows the document consistency to be checked at authoring stage. Second, the design of schedulers (called also controllers) which handle the synchronization during the presentation. Verification tools were generally added to authoring tools or models were the flexibility of the objects were formally described as Petri-Nets with duration ranges [15]. Other systems such as [11, 14] are rather event-based scheduling with less focus on the duration constraints but on the causal relationships between the different scenario events. The work of [5] uses flexibility to check for the consistency and to generate a retrieval schedule which allows to enhence the overall quality of the presentation.

Courtiat [6] tried to tackle consistency validation by using formal methods based on model checking applied to state automatons specified in RT-LOTOS. A reachability analysis is applied to these automatons in order to check for the consistency. The scenario representation corresponds to all possible states of presentation. In [13], the analysis is extended to presentations with different resource conditions. The validation is carried out with the assumption that duration constraints are actually fulfilled. But in a networked environment, with variable delays and resources, the automaton representing the scenario differs from one environment to another according to the available resources. Encompassing all the situations together with the uncertainty in the durations causes an exponential explosion in size of the automaton. In a recent effort, new methods are proposed to overcome this problem by integrating the uncertainty both at the verification phase and at the scheduling. In this case, the automaton serves not only for validation purposes but also for scheduling strategies which prevent the presentation from entering in desynchronized states.

In [7, 10, 17] a similar effort was carried out in order to define verification methods based on constraint propagation under incomplete knowledge of the durations. The consistency of a scenario is checked using constraint propagation techniques. In this case, the representation of the scenario is a constraint graph which is smaller and more manageable than state automatons. The notion of controllability [17] is also introduced for scenarios with uncertain durations. The drawback is that the proposed verification process is incomplete and does not detect all the inconsistencies.

The previous approaches attempt to prevent synchronization errors at the specification level. They are all based on predictive reasoning about all the possible cases produced by variable durations and limited resources at presentation time. With a lack of guarantees from the current network and system infrastructure, desynchronizations are in fact unavoidable. In this context, the design of "best effort" schedulers seems to be a more realistic approach. The consistency is checked independently from the resources and enforced dynamically by the scheduler [2]. In addition, when an unpredicted desynchronization occurs, it is also very useful to rapidly bring the presentation as close as possible to the author's specification. In this paper, our contribution is the design of such a scheduler for scenario-based presentations.

3. Multimedia presentations: Definitions

In this section, we present some useful definitions related to multimedia presentations and scenario scheduling in an indeterministic environment.

First, we define the basic units of a multimedia presentation as the *media objects* which can have different forms, such as text, images, graphics, video, audio and user interaction buttons. These units are combined in a *multimedia scenario* by different types of relations between them, such as temporal relations [1], spatial relations and hyperlinks. The media objects can be classified into two main classes: controllable and incontrollable objects.

3.1. Controllable and incontrollable objects

A *controllable object* O is a media object whose duration of presentation can be chosen by the presentation engine within a given interval $[l_O, u_O]_c$ of durations having lower and upper boundaries (l_O and u_O respectively).

The difference between the upper and the lower duration boundaries defines the *flexibility* of the media object. Examples of controllable objects are discrete objects such as text, images and graphics. Continuous objects such as audio and video can be considered controllable if the system resources (CPU, bandwidth) are available and sufficient to their rendering process.

On the other hand, an *incontrollable object* O is a media object whose duration can be any value in the interval $[l_O, u_O]_i$ but the exact value of duration cannot be known until the end of its presentation. In some cases the values of l_O and u_O are 0 and ∞ respectively, unless the user specifies explicitly lower and upper acceptable boundaries. Examples of incontrollable objects are user interaction buttons, remote media objects that undergo network accessing delays and media objects with unknown durations.

Given these definitions, a temporal scenario can be characterized by its *flexibility* that measures its ability to be adapted to the numeric constraints of durations specified by the user while maintaining the global consistency of the multimedia presentation [8]. This flexibility depends on the flexibility of the controllable objects as well as their position in the graph (as it will be shown in Section 5.1).

The duration boundaries can be defined at the user level (in the source format of the scenario) or at the presentation level. In the first case they are set explicitly by the author in the document, while in the later case, the presentation engine can define these boundaries according to the media types and other contextual information (availability of the CPU, network bandwidth).

3.2. Nominal and effective duration and date values

A correct *multimedia presentation* is the execution of the media objects in a way that respects the specified relations between the objects. Objects are defined by their interval of duration together with a set of temporal relations between them. This data is then passed to a specific component, namely the *temporal formatter*, which statically computes a schedule for the corresponding presentation [3].

We can identify two variables for the duration of the presentation of a media object O: the nominal duration and the effective duration.

The *nominal duration* D_{nom} is the duration statically computed by the temporal formatter before starting the presentation, where $D_{\text{nom}} \in [l_O, u_O]$.

The *effective duration* D_{eff} of an object is the value of the object's duration observed at the end of its presentation, where $D_{\text{eff}} \in [l_O, u_O]$.

A controllable object is characterized by D_{eff} equals to D_{nom}, while for an incontrollable object, this equality cannot be guaranteed because of the indeterminism. Taking into account the nominal duration of objects, the flexibility of a controllable object O can be split into two parts: the part corresponding to its stretching capability $[0, u_O - D_{\text{nom}}]$ and the part corresponding to its shrinking capability $[l_O - D_{\text{nom}}, 0]$.

The presentation can be seen as a sequence of presentation instants, where at each *presentation instant* one or several presentation actions occur. For example, a presentation action can be the start or termination of a media object. Similarly to durations, we can define two dates for the presentation instants: the nominal date and the effective date.

The *nominal date* date_{nom} is statically calculated by the temporal formatter, taking the presentation's starting instant as a reference.

The *effective date* date_{eff} is the exact value of the instant's date, known when the presentation reaches this instant.

A multimedia presentation can be modeled as a directed acyclic *graph* where nodes represent presentation instants and arcs represent objects labelled by their durations (see figure 1).

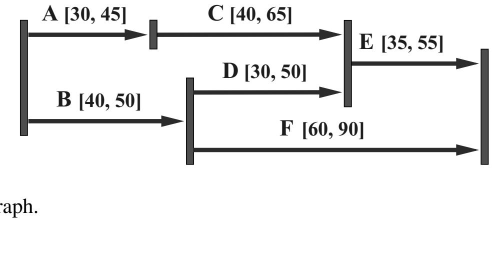


Figure 1. Presentation graph.

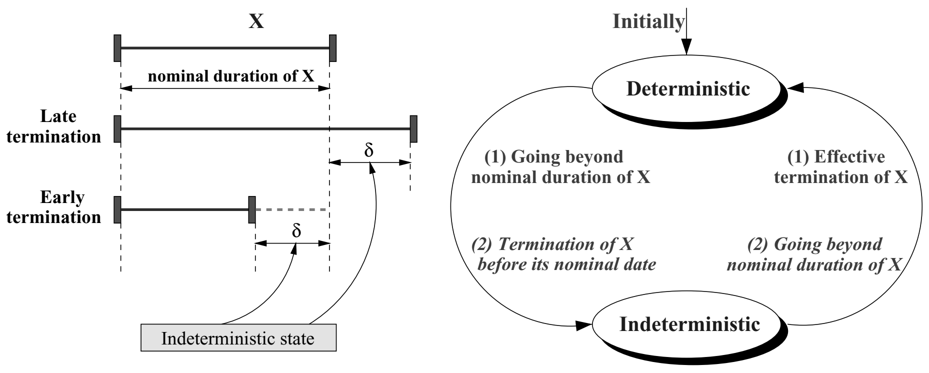

X
nominal duration of X
Late termination
δ
Early termination
δ
Indeterministic state
Initially
Deterministic
(1) Going beyond nominal duration of X
(2) Termination of X before its nominal date
(1) Effective termination of X
(2) Going beyond nominal duration of X
Indeterministic

Figure 2. The indeterministic state of an incontrollable object.

A *chain* is a contiguous sequence of arcs such as each arc is connected to one and only one successor and predecessor arc (objects A and C in figure 1 form a chain).

4. Multimedia presentations in the presence of indeterminism

4.1. Deterministic and indeterministic state

During a multimedia presentation, an incontrollable object can be in one of the following two states: either deterministic or indeterministic (see figure 2). The indeterministic state, corresponding to an *indeterministic period* δ, can occur in two cases:

- The *behind of schedule indeterminism* (BSI) or *late termination*, where the effective duration of the object is greater than its nominal duration. The object returns to its deterministic state when it effectively terminates.
- The *ahead of schedule indeterminism* (ASI) or *early termination*, where the effective duration of the object is less than its nominal duration. This indeterministic state finishes when the nominal termination date of the object is reached.

This definition can be generalized to a multimedia presentation: a presentation is in an indeterministic state when there is at least one incontrollable object in the indeterministic state. Otherwise the presentation is in the deterministic state. Initially, the multimedia presentation is in the deterministic state.

4.2. Sequential and parallel desynchronization

Two types of desynchronization can be introduced between media objects due to the indeterminism: the *sequential desynchronization* and the *parallel desynchronization*.

In order to explain these types of desynchronization, let us take the example of figure 3. The objects A and B are related by a sequential relation (*meets*), i.e., object B starts immediately after the termination of object A. On the other hand, A and X are related by a parallel

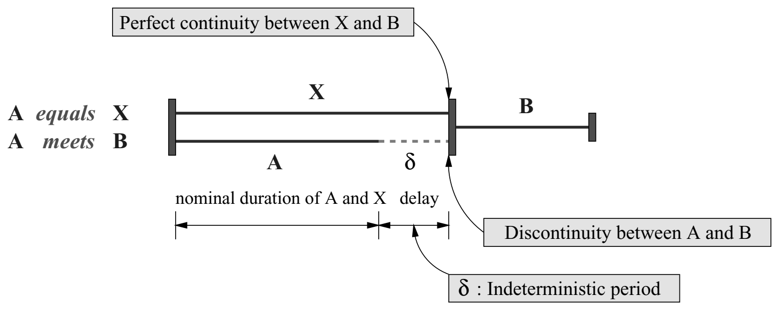

Perfect continuity between X and B
A *equals* X
A *meets* B
X
B
A
δ
nominal duration of A and X
delay
Discontinuity between A and B
δ : Indeterministic period

Figure 3. Sequential and parallel desynchronization.

relation (*equals*), i.e., they start at the same instant and terminate at the same instant. Such a scenario can be expressed in SMIL with the statements: SEQ (PAR (AX, end(A) = end(X)) B). Consequently, by transitivity, the two relations generate an implicit relation between X and B of sequential type (*meets*). Supposing that the presentation of the incontrollable object X goes beyond its nominal duration and lasts for a longer time than A which is not flexible, we have to wait for the termination of X before starting object B. Consequently, a delay δ is introduced between A and B leading to a *sequential desynchronization* between them. So instead of an immediate start of B after the termination of A, the user perceives that the object B waits for a delay δ before starting. In addition, during the indeterministic period in figure 3, there is a *parallel desynchronization* which appears in the form of a shift between the termination instants of objects A and X. Accordingly, the user perceives that the object A has terminated but the object X is still playing during a period of δ.

4.3. Desynchronization period in multimedia presentation

The desynchronization effects due to an indeterministic state usually last longer than the duration of the indeterministic state itself. This is due to a desynchronization propagation along the scenario. So we define a *desynchronization period* (Δ_{desync}) as the period during which the presentation is desynchronized with respect to the specified scenario (see figure 4).

To illustrate the influence of indeterminism and desynchronization propagation on a multimedia presentation, let us consider the example shown in figure 4. In figure 4(b), the incontrollable object X terminates earlier than its nominal termination (*ASI* case). In this case, the node *n*, corresponding to the end of X, must wait for the termination of the other objects incoming at this node (A and B) in order to start the objects outgoing from *n*. Consequently, a sequential desynchronization is produced between X and D and a parallel desynchronization is produced between X and the two objects A and B, but they don't affect the remaining part of the presentation. Thus, the desynchronization period is equal to the indeterministic period δ (see figure 4(b)).

In the case of the *BSI*, where X takes an effective duration greater than its nominal duration (see figure 4(c)), a parallel desynchronization between X and the two objects A and B is observed, and the effective date of node n is shifted δ units of time with respect

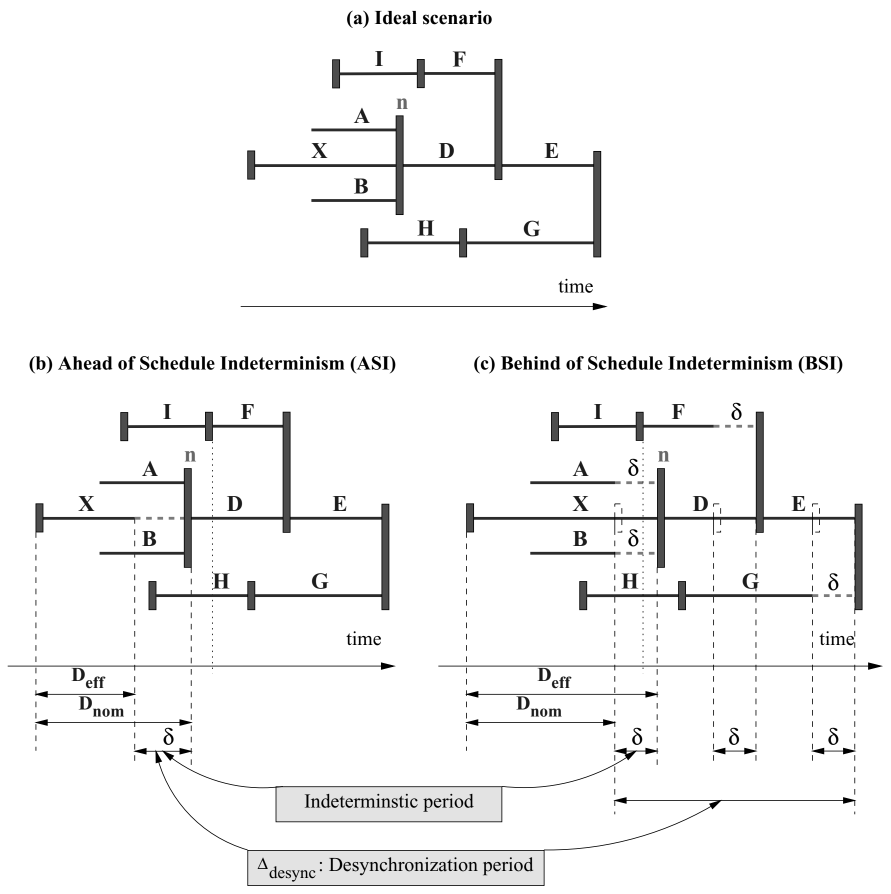

(a) Ideal scenario
I
F
n
A
X
D
E
B
H
G
time
(b) Ahead of Schedule Indeterminism (ASI)
(c) Behind of Schedule Indeterminism (BSI)
δ
D_{eff}
D_{nom}
Indeterminstic period
Δ_{desync}: Desynchronization period

Figure 4. Influence of indeterminism on a multimedia presentation.

to its nominal date (see figure 4(c)). The same shift is observed for the starting and ending instants of the objects D and E. Consequently, a parallel desynchronization will take place between the ending instants of the objects D and F, and similarly for the objects E and G. Thus, the *BSI* is followed by a desynchronization period Δ_{desync} which affects the remaining part of the presentation.

5. Management of indeterminism

It is possible to address the problem of dealing with indeterministic situations through two complementary approaches: dynamic formatting and repairing technique.

The *dynamic formatting* consists in adapting the scenario dynamically, during the presentation, to the indeterministic behavior of the incontrollable objects, in order to obtain

$\Delta_{\text{desync}} = 0$. The *dynamic formatting technique* presented in Section 5.1 aims at providing partial solutions for indeterministic behaviors by exploiting the flexibility of the scenario.

The *reparation technique* (see Section 5.2) consists in limiting the desynchronization effects (i.e., reducing the period Δ_{desync}) during the presentation by delaying some dates. This technique complements the dynamic formating when it fails.

In the remaining part of this section, we present the two techniques proposed. We show then how we combine them to provide an intelligent scheduler and we give some measures that demonstrate the results of applying these algorithms.

5.1. Dynamic formatting

The technique is based on an analysis of the potential indeterministic behavior of the scenario by using the hypergraph. We can identify three cases that can be handled in an ascending order of complexity [9]. Finally, we propose in Section 5.1.4 a general algorithm that can be applied not only to these three cases but also in any situation where an indeterministic period is detected.

5.1.1. Case 1: Articulation chain. The first case corresponds to the situation in which the presence of an incontrollable object does not affect the synchronization constraints specified by the user. Such a case is illustrated in the figure 5.

In this example, the graph can be subdivided in two sub-graphs G1 and G2 which are only related by a chain X containing the incontrollable object (such a chain is called an *articulation chain*). In such a configuration, it is clear that whatever value the effective duration of this object takes, the scenario remains always consistent. Therefore, no compensation is required. The formatter has only to figure out that the position of the incontrollable object in the graph is on an articulation chain in order to ignore the case.

5.1.2. Case 2: Articulation point. The second case corresponds to the situation in which the incontrollable object is present on a chain whose right terminal node corresponds to an articulation point: an *articulation point* is a graph node that can subdivide the graph in two sub-graphs only related by this node. In such a situation, the incontrollable object does not affect the synchronization constraints specified by the user, with the exception of the constraints of temporal coincidence, such as those of the chains B and D with the

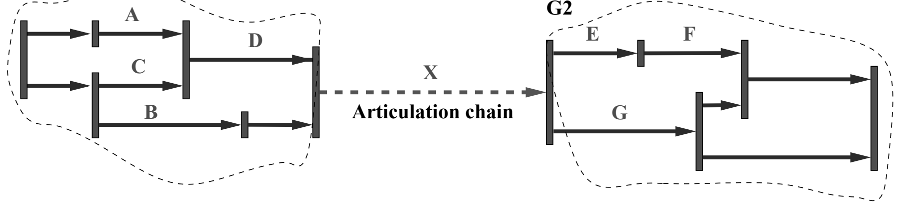


Figure 5. Example of Case 1.

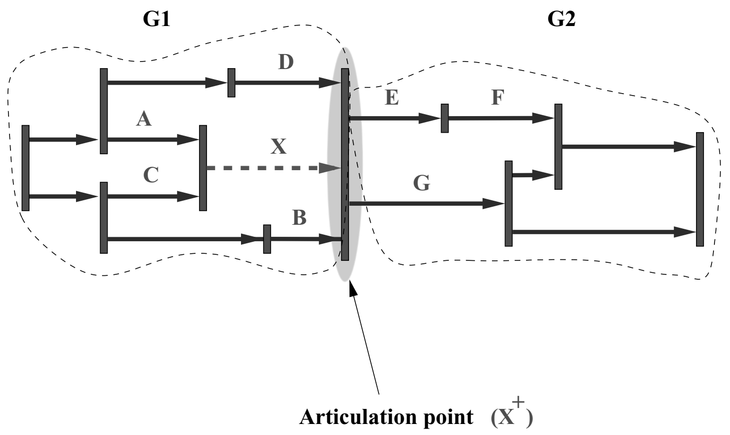


Figure 6. Example of Case 2.

incontrollable chain X in figure 6. Temporal coincidence of two or more chains means that their end instants must occur at the same instant.

In the example of figure 6, the graph can be subdivided also in two sub-graphs G1 and G2 which are only related by the termination instant of the chain containing the incontrollable object X (the articulation point is X^+). This type of situation can not be consistent unless the chains B and D are flexible at their ends, i.e. they end by a discrete object like an image or a text. In this case, the termination of the chain X causes the termination of the other two chains containing B and D. The articulation point provides a re-synchronization point because after it is fired the scenario returns to a consistent state.

5.1.3. Case 3: Compensation on chains. When the ending instant of an incontrollable object occurs, called *observation point*, a more general approach consists in checking if a compensation on the same or on a concurrent chain is possible (see figures 7 and 8). The compensation is based on using the flexibility of the controllable objects whose start and termination instants occur in the future with respect to this observation point. The compensation can be applied on controllable objects during their presentation.

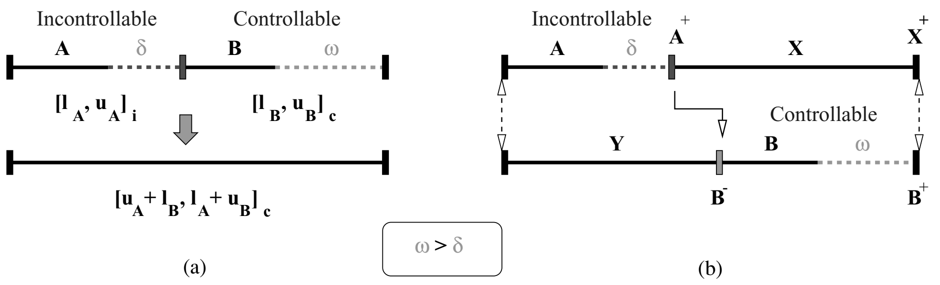
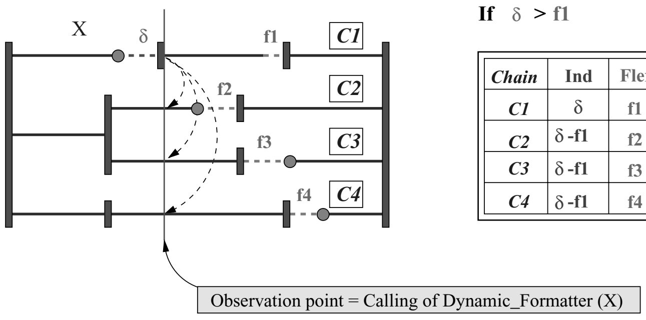


Figure 7. Example of Case 3.



If δ > f1

Chain	Ind	Flex
C1	δ	f1
C2	δ -f1	f2
C3	δ -f1	f3
C4	δ -f1	f4

Figure 8. Compensation on concurrent chains.

As an example, let us consider a chain composed of two objects A and B (see figure 7(a)). A is an incontrollable object A $= [l_A, u_A]_i$ and $\delta = (u_A - l_A)$ represents the value of indeterminism. B is a controllable object B $= [l_B, u_B]_c$ and $\omega = (u_B - l_B)$ represents the amount of available flexibility. If the condition $\omega \geq \delta$ is satisfied, then the indeterminism can be compensated dynamically in a way that renders the whole chain controllable. It is sufficient in this case to use the flexibility ω in a way to make the duration of the whole chain fall in the interval $[u_A + l_B, l_A + u_B]_c$, which is the duration interval of the controllable object equivalent to the chain in question.

If the incontrollable object can not be compensated in its chain, the proposed solution consists in verifying if a usage of the flexibility of the concurrent chains can allow the re-adjustment of the scenario. For example, in the scenario of figure 7(b), the two parallel chains {A, X} and {Y, B} must have equal durations. To verify that this constraint can be satisfied, it is sufficient to make sure that the starting and ending instants of the chains take place at the same date. It is easy to see that if the flexibility ω of the concurrent chain {Y, B} is greater than the indeterminism δ of the chain {A, X} (condition 1), and if the date of the observation instant A^+ (end of A) takes place before the recovery point B^- (start of B) (condition 2), then the temporal coincidence of the instants B^+ and X^+ can be in fact guaranteed.

5.1.4. Dynamic re-formatting of multimedia presentations. The approach we have chosen is a generalization of the cases presented in the previous sections. The algorithm presented here aims at re-adjusting the durations of flexible objects in order to meet the author's specification. It is triggered when an incontrollable object ends. This algorithm minimizes the value of the desynchronization period. The reformatting algorithm is applied on all the concurrent chains of the current execution.

At each instant during the presentation, the scenario represented by the graph is divided in two parts. One part corresponds to the portion of the presentation which has already been rendered to the user, and the second part corresponds to the objects to be presented in the future. The current instant can be represented in the system by the set of current chains (`Ch`), the indeterminism to be compensated (`Ch.Ind`) and the available shrinking and stretching flexibility on each active controllable objects O belonging to the chain:

```
Dynamic_Formatter(Object:X)
1. waiting_list = $\emptyset$;
2. /* Calculation of indeterminism value */
3. $\delta$ = X.end.date$_{eff}$ - X.end.date$_{nom}$;
4. If $\delta \neq 0$ and {Concurrent chains} $\neq \emptyset$
        /* ASI or BSI */
5.     /* Compensation on the same chain */
6.     Remain=Format_Chain(X.Chain,(-1)*$\delta$);
        /* see figure 10 */
7.     If Remain $\neq$0
8. /* Compensation on concurrent chains */
9.       For every concurrent chain Ch {
10.        Ind = Format_Chain (Ch, Remain);
             /* see figure 10 */
11.        If (Ind > 0) Then {
12.          Ch.Ind = Ind;
13.          waiting_list = waiting_list $\cup$ Ch;
14.        }
15. }
16. Return waiting_list;
```

Figure 9. Dynamic formatter.

`O.flex`$^-$ and `O.flex`$^+$. From any object X, we can access the corresponding chain on the graph (`X.Chain`).

The algorithm of figure 9 is based on three phases: first the value of indeterminism due to object X is calculated (line 3). Then, the graph adjustment is taken into account on the chain where this object is found (case (a) of figure 7). If the flexibility of the chain, from this instant until its end, is sufficient, the indeterminism is compensated by reformatting this chain (lines 4–6). This operation consists of shrinking or stretching the chain's duration by a value equals to the observed indeterminism in the case of *BSI* or *ASI* respectively.

Finally, the adjustment is performed on concurrent chains if needed (lines 7–14) such as case (b) of figure 7. In the case of *BSI*, this operation consists of stretching the chains durations by a value equals to the observed indeterminism. In the case of *ASI*, the concurrent chains are shrunk in order to end the indeterministic period. If the reformatting of the chain does not completely compensate for the whole value of indeterminism, then the chain together with the amount of uncompensated indeterminism are added to a waiting list (lines 11–13). The chains in this waiting list are treated later by the technique of reparation of indeterminism presented in Section 5.2. If at least one chain is in the waiting list, this means that the formatter is unable to completely resynchronize the scenario according to the specification.

For example, in figure 8, the indeterminism δ, is observed on object X of chain C1. The formating of the chain cannot be totally achieved because the flexibility f1 is less than indeterminism δ. The concurrent chains C2, C3 and C4 are consequently reformatted in order to be elongated by a value of δ- f1.

```
Format_Chain(Chain:Ch,integer:Ind)
1. For every object O $\in$ Ch |
       Current_time $<$ O.end.date$_{nom}$
2. And Ind$\neq$0 {
3.     If (O.type = discrete) or
        (Current_time $<$ O.begin)
4.      Ind = Modify_Durations(Ind, O);
5. }
6. Return Ind;

Modify_Durations(integer:Ind,Object:O)
1. If ind $>$ 0 Then
       /* if O $\in$ X.Chain it's a ASI */
2.    /* otherwise it's a BSI*/
3.    /* stretch object O */
4.    If O.Flex$^+$ $\geq$ Ind Then {
5.      O.D$_{nom}$ = O.D$_{nom}$ + Ind;
6.      O.Flex$^+$ = O.Flex$^+$ - Ind;
7.      Return 0; /* compensation is done */
8.    } Else { /* partial compensation */
9.       O.D$_{nom}$ = O.D$_{nom}$ + O.Flex$^+$;
10.      Ind = Ind - O.Flex$^+$;
11.      O.Flex$^+$ = 0;
12.      Return Ind;
          /* remaining indeterminism */
13.    }
14. Else /* if O $\in$ X.Chain it's a BSI */
15.     /* otherwise it is a ASI */
16.     /* shrink object O */
17.    If O.Flex$^-$ $\leq$ Ind Then {
18.      /* Equivalent to lines 5 to 12 but
          with O.Flex$^-$ */
19. }
```

Figure 10. Chain formatting.

The algorithm of formatting chains is presented in figure 10. For each chain, the algorithm modifies the duration of the controllable objects who have not yet terminated and of the continuous objects who have not yet started (lines 1–3). If the indeterminism is not completely absorbed (ind $\neq$ 0, line 2), the remaining uncompensated indeterminism is passed to the next controllable object on the chain and so on until the end of the chain.

The Modify_Durations procedure considers two cases depending on the sign of the indeterminism ind to be compensated: if ind is a positive value, the object must be elongated so O.Flex$^+$ is used (lines 4–12). Otherwise it is a shrinking operation that takes into account the negative flexibility O.Flex$^-$ of the object (lines 14–19). Each object tries to compensate for the indeterminism by providing the maximum of flexibility (lines 4–7).

The advantage of this technique is to provide an automatic re-establishment of the synchronization between the different chains in a local and non-blocking way. As mentioned earlier, this dynamic formatting technique does not cover all the cases of indeterminism. When it fails, a complementary algorithm (as presented in Section 5.2 below) is necessary to dynamically minimize the desynchronization period resulting from the uncompensated indeterminism.

5.2. *Reparation of indeterminism*

We can point out one important remark that the reparation of indeterminism should take into account: the case of *BSI* is more crucial to compensate than the *ASI* (see figure 4). The desynchronization due to the *BSI* can propagate till the end of the presentation.

In this section, we present the algorithm of reparation of indeterminism (see figure 11) whose objective is *the reduction of the desynchronization period following an indeterministic period of a behind schedule indeterminism* (BSI). The proposed solution is based on the postponement of the presentation dates of media objects whose starting or ending instants come during or after an indeterministic period. This postponement is carried out in order to re-establish the synchronization specified in the scenario.

The repairing formatter, presented in figure 11, treats every chain in the waiting list as follows. For each chain, the algorithm finds out the node n on the chain that comes directly after the nominal termination date of the incontrollable object X (line 2). Then, handling of this node n depends on the value of its date with respect to the nominal termination date of X. Two cases can occur:

– Case 1 (line 4): If the nominal date of node n comes before the effective termination date of X, then the algorithm will handle all the outgoing objects from node n (object O in the algorithm).

```
Repairing_formatter(Object:X,Waiting_List:WL)
1.  For every chain ch $\in$ WL {
2.    n = Get_Nearest_Future_Node
          (ch, X.end.date$_{nom}$);
3.    For every object O such that
4.    ((O.start = n) And
          (n.date$_{nom}$ $\leq$ X.end.date$_{eff}$))
5.    Or
6.    ((O.end = n) And
        (n.date$_{nom}$ $>$ X.end.date$_{eff}$)) {
7.      If (O.type = discrete) Then
8.        O.end.date$_{nom}$ = O.end.date$_{nom}$ + ch.Ind;
9.      Else If (O.type = continuous) Then
10.       Insert_Delay (O.end, ch.Ind);
11.   }
12. }
```

Figure 11. Repairing formatter.

– Case 2 (line 6): If the nominal date of node n comes after the effective termination date of X, then the algorithm will handle all the incoming objects to node n (object O in the algorithm).

In both cases, the object O is handled according to its type. If object O is discrete, its nominal termination date is postponed by the value of the indeterminism *Ind* (lines 7–8). Otherwise, if the object is of a continuous type, a delay of the value *Ind* is inserted after its termination (lines 9–10).

5.3. Combination of both approaches

The dynamic formatter and the repairing formatter are combined in a complementary way. Once the presentation of an incontrollable object (say X) terminates, an event is generated and the indeterminism handler is called.

As a first step, the indeterminism handler calls the dynamic formatter which applies the technique presented in Section 5.1. The dynamic formatter answers back to the handler whether it has fully treated the observed indeterminism or not. If not, the dynamic formatter creates a waiting list where each entry represents a chain associated to which the value of uncompensated indeterminism on this chain. The waiting list is then passed to the repairing formatter in order to handle the remaining uncompensated indeterminism.

5.4. Experimental evaluation

This algorithm has been implemented in our multimedia authoring and presentation system Madeus. It allowed the desynchronization period to be reduced by a factor up to 58.82% (see figure 13). In the general case, a scenario authored with sufficient information on the flexibility it contains will be better handled at runtime by our scheduler. In the other hand, a scenario which includes a significant number of parallel chains with no flexibility is likely to be desynchronized when a delay occurs. The source of flexibility comes from the nature of media because discrete objects are less sensitive to delays or duration changes. The desynchronization reduction on a larger set of tests including half of the scenarios with no flexibility gives around 35% of desynchronization reduction. In these cases, we prevented the dynamic formatter from using the media flexibility to resynchronize the presentation. Therefore, the repairing formatter was a key component in this later cases as the dynamic formatter was only used to identify articulation points and chains. In general, it is difficult to provide general measures of performances which cover all the possible scenarios and presentation conditions.

We made different scheduling tests on some scenarios and we applied a desynchronization function during the execution. This function gives the instantaneous desynchronization at each instant of an execution. It is defined as the sum of the sequential and parallel desynchronizations occurring at that instant. In figure 12, the following executions are represented together with their desynchronization function and desynchronization period Δ_{desync}:

(a) The ideal execution (the incontrollable object X respects its nominal duration).
(b) The execution with BSI desynchronization due to object X *without* applying neither dynamic formatting nor repairing formatting.

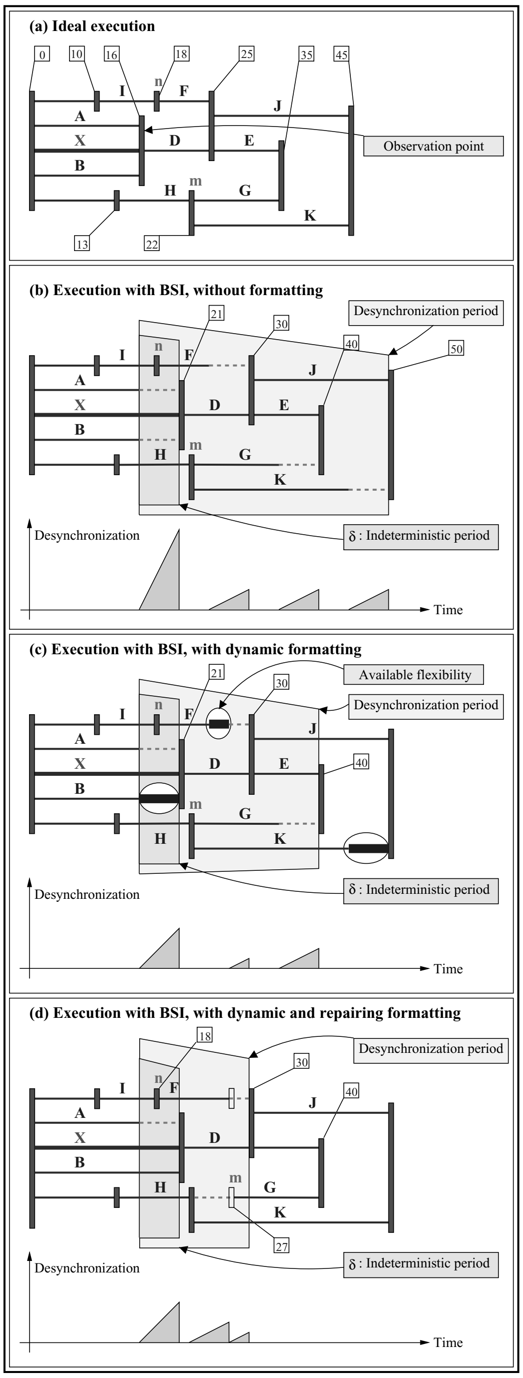

(a) Ideal execution
0
10
16
18
25
35
45
13
22
I
n
F
J
A
X
D
E
B
H
m
G
K
Observation point
(b) Execution with BSI, without formatting
21
30
40
50
Desynchronization period
δ : Indeterministic period
Desynchronization
Time
(c) Execution with BSI, with dynamic formatting
Available flexibility
Desynchronization period
δ : Indeterministic period
(d) Execution with BSI, with dynamic and repairing formatting
27

Figure 12. Indeterminism with a single incontrollable object.

Number of incontrollable objects	**δ (sec)**	**Δ_{desync}**			**Reduction of desynchronization (%)**
		Without formatting (sec)	**With dynamic formatting (sec)**	**With dynamic and repairing formatting (sec)**	
1	**5**	**34**	**24**	**14**	**58.82%**
2	**8**	**35**	**15**	**15**	**57.14%**
3	**14**	**37**	**27**	**17**	**54.05%**

Figure 13. Results of the algorithm on different scenarios.

(c) The execution with BSI desynchronization due to object X while applying *only* the dynamic formatting (objects F and B are flexible).
(d) The execution with BSI desynchronization due to object X while applying *both* the dynamic formatting and the repairing formatting.

This example shows that the *dynamic formatting* reduces the desynchronization level thanks to object's flexibility it uses while the *repairing formatting* shifts the remaining desynchronization ahead and so reduces the desynchronization period Δ_{desync}.

We have applied the repairing algorithm on scenarios that have several incontrollable objects. As expected, best results are obtained when only one incontrollable object is involved at a time (figure 13). In fact, the most interesting effect of this algorithm is to stop the cumulative propagation of the indeterminism along the scenario thanks to an incremental application of the reparation. This advantage is partially lost in very uncertain situations where successions of incontrollable objects occur unless they are located on articulation points or chains. But even in the first case, the algorithm enhances the resynchronization by bringing the scenario as close as possible to its nominal schedule.

The experimental environment used in this paper was based on the Madeus system running on an Ultra-sparc 1 machine from Sun Microsystems and the solaris 2.6 operating system. The Madeus client is a fully functional multimedia authoring and presentation system capable of rendering remote documents and media objects using HTTP and RTSP protocols. The client is written in C language and the scenario specification is XML-based mark-up where the objects durations are defined as minimum and maximum bounds and are of two types: controllable and incontrollable. In the experiment, the delays were introduced in two manners: first by accessing media items located in remote sites and second by artificially overloading the system's CPU. The banchmarks are documents written by several authors. They did not necessarily use the capabilities of the tool to describe precisely the flexibility contained in the documents.

6. Conclusion and perspectives

This paper addressed scenario-based scheduling of multimedia presentations in the presence of indeterministic durations. The presented algorithms are adaptive in that they use the flexibility of media object and the temporal relations given by the author to resynchronize the presentation. Furthermore, this operation is achieved dynamically each time a desynchronization occurs. The algorithms are based on two complementary phases:

– The reformatting which attempts to modify the duration of the objects while maintaining the author's specifications. The resynchronization is achieved in a non-blocking manner and even when it partially succeeds (due to the lack of flexibility), it reduces the postponement delays required by the reparation phase.
– The reparation process which is activated when the reformatting does not totally succeed, i.e., when author's specifications cannot be respected. This process aims at minimizing the period of desynchronization resulting from the indeterminism.

Both algorithms take full advantage of the predictive knowledge of the temporal scenario represented as a graph. Furthermore, the organization of the graph as a set of concurrent chains allows to measure the impact of the indeterminism on the presentation. It allows also to identify the durations to modify in order to re-synchronize the presentation.

Our algorithms can be easily integrated in other media players where the document is described in terms of media objects and relations. It is particularly suited for environments subject to dynamic load conditions and bandwidth resulting in variable object durations. Examples of players are SMIL document format players like RealPlayer G2 [12], GRiNS [4] and SOJA [16]. SMIL documents are already in use and disseminated through the internet and intelligent scheduling policies are highly needed in that context. The flexibility on which our algorithms are based can be affected by the presentation engine depending on the nature of the objects, their locations, etc.

Directions of future work might include a fine tuning of the reformatting on the different chains. For example, flexibility is in our case taken from objects in the nearest future independently from the media types. A better method would be to set priorities between media objects in a manner that affects the least the presentation quality. Another solution would be to capture more information at the documents level about the author's intents and to translate them in terms of priorities.

References

1. J.F. Allen, "Maintaining knowledge about temporal intervals," Communications of the ACM, Vol. 26, No. 11, pp. 832–843, 1983.
2. K. Altisen, G. Gössler, A. Pnueli, J. Sifakis, and S. Tripakis, "A framework for scheduling as controller synthesis," VERIMAG, Grenoble, France, www-verimag.imag.fr, Internal Report, 1998.
3. M.C. Buchanan and P.T. Zellweger, "Automatic temporal layout mechanisms," in Proceedings of the First ACM International Conference on Multimedia, ACM Press: Anaheim, Californie, 1993, pp. 341–350.
4. D. Bulterman, L. Hardman, J. Jansen, K. Mullender, and L. Rutledge, "GRiNS: A GRaphical INterface for creating and playing SMIL documents," in Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, 1998, pp. 519–529.

5. K.S. Candan, B. Prabhakaran, and V.S. Subrahmanian, "Retrieval schedules based on resource availability and flexible presentation specifications," ACM/Springer-Verlag Journal on Multimedia Systems, Vol. 6, No. 4, 1998.
6. J.P. Courtiat and R.C. De Oliviera, "Proving temporal consistency in a new multimédia synchronisation model," in Proceedings of the Fourth ACM Conference on Multimédia, ACM Press: Boston, 1996.
7. H. Fargier, M. Jourdan, N. Layaïda, and T. Vidal, "Using temporal constraints networks to manage temporal scenario of multimedia documents," ECAI 98 Workshop on Spatial and Temporal Reasoning, Brighton (UK), 1998.
8. M. Kim and J. Song, "Multimedia documents with elastic time," in Proceedings of the Third ACM International Conference on Multimedia, Polle Zellweger, San Francisco, California, 5–9 November 1995, ACM Press, pp. 143–154.
9. N. Layaïda, "Madeus: un système d'édition et de présentation de documents structurés multimédia," Ph.D. thesis, Université Joseph Fourier, Grenoble, France, 1997.
10. N. Layaïda and L. Sabry-Ismail, "Maintaining temporal consistency of multimedia documents using constraint networks," in Multimedia Computing and Networking 1996, M. Freeman, P. Jardetzky, and H.M. Vin (Eds.), SPIE 2667, 1996, pp. 124–135.
11. L. Li, A. Karmouch, and N.D. Georganas, "Teleorchestra with independent sources : Part 1—temporal modeling of collaborative multimedia scenarios," ACM/Springer-Verlag Journal on Multimedia Systems, Vol. 1, No. 4, 1994.
12. Real G2, RealNetworks Announcing RealSystem G2, on line: http://www.realaudio.com/g2/index.html, 1998.
13. C. Santos, L. Soares, G. De Souza, and J.P. Courtiat, "Design methodology and formal validation of hypermedia documents," in Proceedings of the Sixth ACM International Conference on Multimedia, Bristol, UK, 1998, ACM Press, pp. 39–48.
14. J. Schnepf, J.A. Konstan, and D.H. Du, "Doing FLIPS: Flexible interactive presentation synchronization," IEEE Journal on Selected Areas in Telecommunications, Vol. 14, No. 1, 1996.
15. P. Senac, M. Diaz, A. Leger, and P. Saqui-Sannes, "Modeling logical and temporal synchronization in hypermedia systems," IEEE Journal on Selected Areas in Telecommunications, Vol. 14, No. 1, 1996.
16. SOJA "Barbizon," "SMIL at HELIO," En Ligne: http://www.helio.org/products/smil, 1999.
17. T. Vidal and H. Fargier, "Contingent durations in temporal CSPs: From consistency to controllabilities," in Workshop Proc. TIME-97, Dayton Beach, FL, USA, 1997.
18. World Wide Web Consortium Recommendation, "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification," http://www.w3.org/TR/REC-smil, 1998.

Nabil Layaïda received a Ph.D. degree in computer science from University of Grenoble in 1997. Since 1998, he is working as a reserach officer at INRIA (Institut National de Recherche en Informatique et en Automatique). He is managing the presentation services for multimedia systems group within the Opera project. His research interests are in the fields of interactive structured and multimedia document processing, adaptive documents, multimedia applications for the mobile devices, structure transformations, temporal scheduling and synchronization and their applications on the Web. He is also a member of the SYMM working group at the World Wide Web Consortium and co-author of SMIL 1.0 and SMIL 2.0 recommandations (Synchronized Multimedia Integration Language) of the World Wide Web Consortium.

Loay Sabry-Ismail received a Ph.D. degree in computer science from University of Grenoble in 1999. He is now working as assistant professor at the Faculty of Engineering, Ain Shams University, Cairo, Egypt. His research interests are in the fields of system support for multimedia document processing systems, quality of service, network protocols for multimedia, temporal synchronization.

Cécile Roisin received a Ph.D. degree in computer science from University of Grenoble in 1984 and her "Habilitation à diriger des recherches" in 1999. She has been working as assistant professor at the University of Grenoble since 1987 and works also at INRIA (Institut National de Recherche en Informatique et en Automatique). Her research interests are in the fields of interactive structured and multimedia document processing, mainly in complex formatting, video manipulation and temporal synchronization.