MUSE - An Interactive Networked Multimedia Applications Specification Environment with E-LOTOS Translator

Luciano Paschoal Gaspary
Maria Janilce B. Almeida

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Curso de Pós-Graduação em Ciência da Computação
Campus do Vale, Bloco IV – Bento Gonçalves, 9500 – Agronomia – 91591-970
Porto Alegre, RS – Brazil
E-mail: {paschoal, janilce}@inf.ufrgs.br

Abstract. This work presents MUSE, a graphical environment for modeling interactive networked multimedia applications. Through an advanced graphic interface and a new high-level authoring model, it is possible to create complex systems in a fast and intuitive way. The authoring model proposed in this work and adopted by the environment deals with media objects distributed in a computer network, allowing the definition of acceptable presentation delay thresholds and alternative media objects. Due to the large expressiveness of the model, however, specifications with logical and temporal inconsistencies may be generated. For this reason, the tool also provides E-LOTOS specifications, which may be used to analyze and verify the temporal requirements defined by the author.

1 Introduction

The 90's have been known by the use of multimedia applications in several fields of the human activity such as education, medicine and entertainment. These applications have become increasingly sophisticated along the time, and nowadays they are executed in distributed environments, operating transparently in heterogeneous platforms.

The possibility of having an application with its media objects dispersed in a network influences the creation and modeling of such applications. Users must provide the authoring tools with information like temporal restrictions, defining acceptable delay thresholds to the presentation of the elements that compose the system and establishing the presentation of alternative media objects.

The definition of these restrictions is accomplished based on a synchronization model, which dictates the rules about how the media objects of an application can be related in time. Several synchronization models have been proposed [1]. Most of them are both flexible and very expressive. That is the reason why the resulting specifications can be source of incoherences, where the logical and temporal consistency of the involved media objects can not be assured. An alternative would be to use directly a formal description technique (FDT) to describe the applications, making its analysis possible and so guaranteeing its consistency. The disadvantage of this direct usage, however, is the high complexity inherent to FDTs. So, the need of

having a structured high-level model to specify interactive networked multimedia applications becomes evident. The resulting specifications shall then be translated to an FDT, so that verification and simulation methods can be applied to them.

In this context, an interactive networked multimedia applications authoring model was created. MUSE (MUltimedia Applications Specification Environment) was developed to support this model, allowing the user to easily define a multimedia presentation according to the MHEG-5 standard [2]. The adoption of MHEG-5 allows multimedia information to be shared without worrying about the platform or operating system used, providing specification and development of portable applications. To make the validation process of the specifications possible, the environment automatically generates E-LOTOS specifications.

This work is part of DAMD (Distributed Multimedia Applications Design) project, sponsored by the Brazilian research council. Its main objectives are to provide a methodology to completely cover the distributed multimedia applications development cycle and to allow authors who are not expert in formal methods to easily develop their applications. The project was developed according to figure 1. MUSE, in a certain way, centralizes the process that comprehends modeling and presentation of applications. Specifications created by the user are validated and the obtained results are presented to him in a quite readable way in the own tool. The specification-validation process repeats until the incoherences are eliminated. After that, MHEG-5 applications are generated and can be executed by the engine.

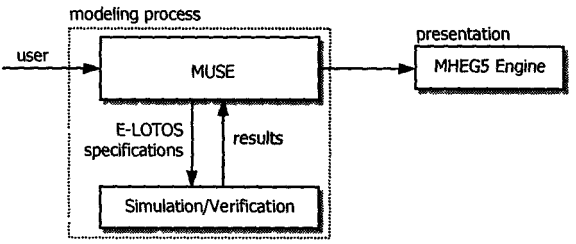


Fig. 1. Structure of the DAMD project

This paper is organized as follows: section 2 presents important aspects to be considered in the applications authoring process, relating them to some multimedia synchronization models pointed by the literature. This section also presents the proposed authoring model. In section 3 basic aspects of the E-LOTOS FDT are presented, as well as a mechanism to represent specifications generated by the authoring model in this formal technique. Section 4 illustrates the functionality of the environment and in section 5, one can read the final considerations.

2 Proposed Authoring Model

The specification of multimedia applications is accomplished with base on three fundamental aspects: logical structuring, establishment of temporal relationships and spatial definition among the elements belonging to the application. The logical

structuring is concerned to offer abstraction mechanisms, providing a wide and structural view of the application. The specification of the temporal behavior involves the definition of synchronization relations among media objects. The spatial synchronization cares about adjusting the positioning of the visible media objects according to the output devices (video).

The temporal relations are established according to a synchronization model, which imposes rules on how these elements can relate to each other. Several models have been proposed in the literature. One of the most adopted by existent authoring tools is the time-line based one [3]. However, it presents many limitations such as the difficulty both to modularize the application and to establish relations among elements with variable or unknown duration like user interaction [4].

The hierarchical model also presents deficiencies. The most important one is that the construction and reading process of the specifications is not natural. It is not clear the order in which the media objects will be presented. Besides, the model does not allow the establishment of some synchronization rules [1], which restricts the expression power of this model. Models based on references to points are not adequate to model distributed multimedia applications, because there is no an explicit time notion. Thus, temporal restrictions can not be expressed and temporal irregularities (common in distributed systems) are ignored in this model.

In synchronization models based on Petri nets, it is possible to specify most of the important synchronization rules required for modeling multimedia applications [5]. Among the models up to now presented, this one provides the largest expression power and flexibility. Moreover, as Petri net is a formal model, it makes applications analysis possible, allowing its consistency to be guaranteed. Its largest disadvantage, however, is its complexity; the manipulation of large specifications may become difficult because of the state explosion problem.

In this work, an authoring model that joins mechanisms for logical structuring the applications to a synchronization model similar to HTSPN is proposed. The logical structuring level is based on the concept of scenes and groups, providing a broad view of the application. The definition of temporal synchronizations is done in each scene by means of a simplified graph. The spatial synchronization allows media objects to be positioned considering the output device (see figure 2).

2.1 Logical Structuring

The complexity of multimedia applications increase according to the growth of the number of involved media objects and, consequently, to the several temporal relationships established among them. This is the fundamental reason why the specification of these applications in only one plane is inappropriate. To solve this problem, the concept of scenes was incorporated into the model considering the MHEG-5 standard. Multimedia applications can be organized as a group of scenes related by events, which provide the navigation among them. Each of these scenes can be seen as a black box with an internal behavior that, under certain conditions, enables the presentation of other scenes.

The use of this concept, however, does not solve completely the problem of complexity, since a specification with many scenes will be hardly understood. Trying

to make easier the understanding of so large applications, a hierarchy mechanism was added to the model through the concept of group of scenes. The top of figure 2 illustrates the logical structure of an application, composed by four scenes (Scene1, Scene2, Scene3 and Scene4). Three of them (Scene2, Scene3 and Scene4), due to the cohesion established among them, were gathered in Group1.

The arcs that link groups and scenes in the logical structure do not represent temporal synchronizations, but choices. For example, a scene A tied up to two scenes B and C indicate that the temporal behavior of the scene provides two ways for the application to evolve: either to B or to C, only depending on the dynamic behavior of the application. This evolution is materialized by the use of the transition icon, to be mentioned in the following section.

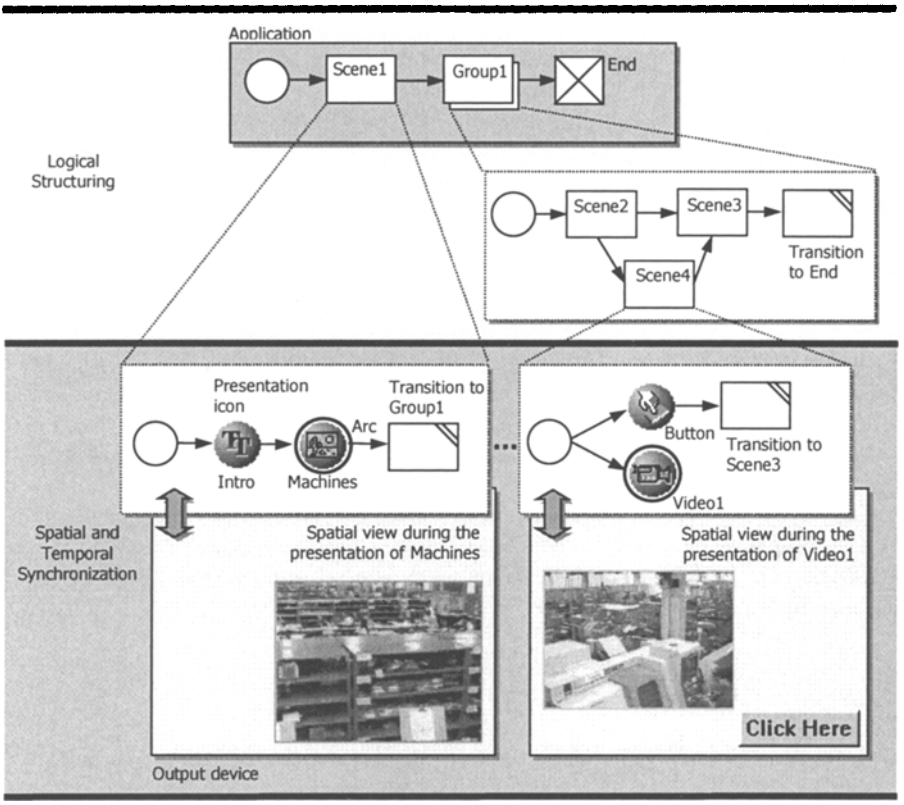


Fig. 2. Structure of an interactive multimedia application

Usually, there are media objects whose presentation embraces several scenes. With the purpose of increasing the expressiveness of the model, the possibility of representing media objects shared by several scenes was created. Figure 3 shows an application organized in three scenes (Scene1, Scene5 and Scene6) and a group (Group1). The image Logo is presented simultaneously to the whole application, and Instructions, during the presentation of Group1 and Scene5.

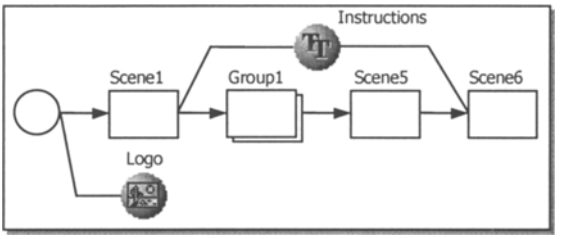


Fig. 3. Media objects shared among scenes and groups

From the authoring process perspective, the proposed structure facilitates the reuse of scenes and groups that repeat in different specifications. Besides, the model allows the development of templates – basic pre-constructed scenes – whose utilization makes the specification process evolving and incremental. One can have a set of templates, so that the specification process, in this case, is reduced to joining these different scenes, lessening drastically the development efforts.

2.2 Temporal Synchronization

The temporal synchronization of an application, as mentioned previously, refers to the ordering of the presentation of its media objects in time. Each media object has a presentation duration that may or may not be foreseen, depending on its nature. The following topics present how the synchronization relationships can be established.

Basic Synchronization. Media objects can be presented sequentially or simultaneously. In the sequential presentation, the playout of a media object depends on the end of another's. In figure 2 both types of basic synchronization appear. In Scene1, the presentation of a text (Intro) is followed by the presentation of an image (Machines). In Scene4, there is the simultaneous presentation of a video (Video1) and a button (Button).

Event Duration and Delay. A minimum and a maximum duration of presentation are associated to each media object. In the case of an image or a text, these values are equivalent because they are time-independent media objects. When one deal with media objects like audio and video, however, it is important to determine both a minimum and a maximum presentation duration, since these media objects will be hardly presented at the nominal rate due to problems like network traffic. The representation of these durations is given by an interval.

To make the modeling of a delay between the presentation of two consecutive media objects possible, a special icon can be used. It does not have any media object associated to itself but only a specific value representing how long it has to wait to start the presentation of its successive media. Figure 4 illustrates three slides (Slide1, Slide2 and Slide3) being presented sequentially with a delay of three seconds between the first and the second and a delay of five time units between the second and the third one.

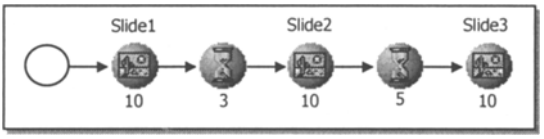


Fig. 4. The delay icon

User Interaction and Scene Transition. User interaction corresponds, for instance, to a button click or an object selection. It is represented in this model as a constructor whose presentation duration is uncertain, varying between the minimum and maximum values associated to it. When the maximum threshold is reached, the scene continues with its presentation. It is still possible to specify a button without maximum duration; in this case, its evolution will only happen after the interaction.

The user interference is normally associated to a scene transition. Transition is the constructor that makes the navigation among scenes possible. Its execution involves both the immediate suspension of the presentation of all the media objects belonging to the current scene and the beginning of a new scene presentation. In Scene4 (see figure 2), the transition to Scene3 occurs after the hitting of the button; if the video (Video1) is still being presented at that instant, it is interrupted.

The connections described in the logical structure and the transitions used in the temporal structure must be consistent to each other. In Scene1, for example, the only acceptable transition is to Group1, once in the logical structure the scene only has connection to the icon that indicates this group.

Synchronization Points. Synchronization points allow the beginning of the presentation of one or more media objects to be associated to different policies related to the end of the presentation of other media objects that converge to these points. To simplify the graphical representation of the authoring model, synchronization points involving only two presentations are not shown. For instance, in figure 4 the synchronization points between Slide1 and the delay and between the delay and Slide2 are not presented.

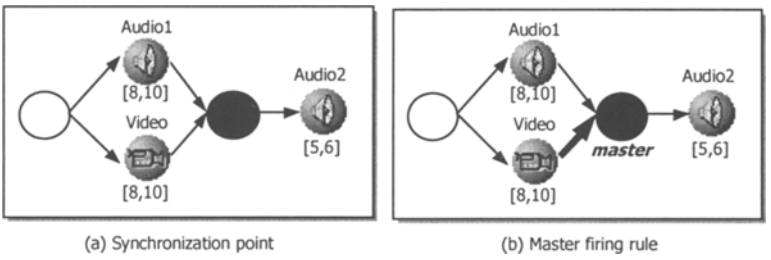


Fig. 5. Synchronization point and firing rules

To increase the specification power, the model has adopted some policies widely commented in the literature. They allow the association of different behaviors to the synchronization points [6]. For simplification, the model only supports three of them:

- *Master*: the synchronization point is fired when the presentation of a master media object is finished, interrupting all the others. This rule could be used in the example of figure 5a above if one wishes that the end of the presentation of Video (master) causes the interruption of Audio1, starting Audio2 (see figure 5b). The master media object is identified by the presence of the character *m* or the word *master* close to it.
- *Earliest*: the synchronization point is fired when the presentation of the first media object is finished, resulting in the interruption of the others. This rule is graphically represented by the presence of the character *e* or the word *earliest* close to the synchronization point.
- *Latest:* the absence of an indication close to the media object or to the synchronization point means that all the media objects that precede this point will be executed (or they will conclude due to the elapsing of their maximum presentation duration) before the synchronization point is fired (figure 5a).

Instants of Synchronization. In MUSE, the synchronization among media objects in other instants than the beginning and end of its presentations requires the division of these media objects in parts, creating a set of segments. The granularity of this division is associated to the precision degree desired for the synchronization. Figure 6 shows the synchronization of two subtitles (Subtitle1 and Subtitle2) with a video (VD), where the latter is divided into four segments. The first subtitle is presented simultaneously to the second video segment and the second subtitle together with the third segment.

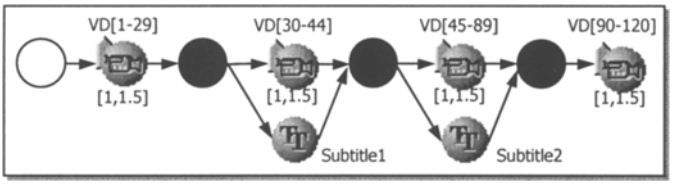


Fig. 6. Synchronization of a video with subtitles

2.3 Spatial Synchronization

The spatial synchronization allows the author to visualize the positioning of the visible media objects of a scene. It is not possible to accomplish the spatial structuring considering a certain time elapsed after the beginning of the scene execution. It is so because each of the executions of the application, due to the acceptable temporal variations, the media objects can be presented in different instants. For this reason, the spatial synchronization is always accomplished with relation to the presentation of a media object. The spatial arrangement of the media objects of Scene1 (see figure2)

during the presentation of Machines, for example, will only allow the bitmap Machines to be organized. On the other hand, the spatial view of Scene4 during the presentation of Video1 will present the media objects Video1 and Button. The appearance of Button occurs because it is defined to be simultaneously presented with Video1.

2.4 Example of Model Usage

The example illustrated in figure 7 models the application proposed in [1], where initially a video (VD) and an audio (AU) are executed simultaneously. Following, a recorded user interaction (RI), a sequence of three slides (P1-P3) and an animation (ANI) which is partially commented by an audio sequence (Audio2) are presented sequentially. During the animation, a multiple-choice question is presented to the user (Interaction). If the user makes the selection, a final image (P4) is presented. This is just one of several ways of representing this application. The ease in understanding it is obtained mainly by the user's good sense in the moment of its specification.

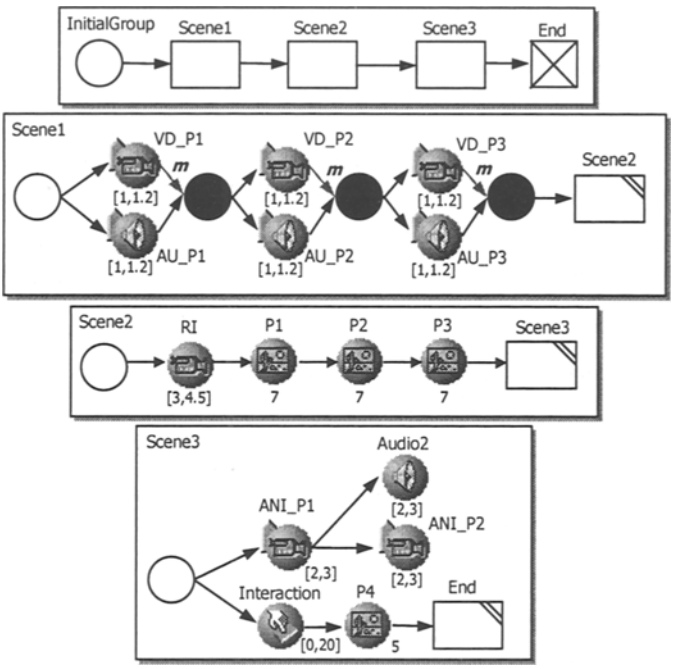


Fig. 7. A simple example of the model usage

3 Representation of Multimedia Applications in E-LOTOS

The formalization of specifications is important for the process of their validation. The proposed authoring model, due to its high flexibility and expressiveness, allows both temporally and logically incoherent specifications to be defined. The analysis process detects, for example, conflicts in resources usage and tests if the application's end can be reached from all the possible navigation paths. Thus, specifications described by an author according to the model presented in the previous section are translated to a formal representation, analyzed and the obtained results are presented to the user, who will make the necessary adjustments.

The formal description technique E-LOTOS (Enhancements to LOTOS) [7] is an enhanced version of LOTOS and is in standardization process. The main innovation of the language is the incorporation of quantitative time notion, allowing the definition of instants in which actions or events may happen. This is a fundamental feature for representing multimedia applications and, for this reason, E-LOTOS was chosen to formally represent them.

The representation of multimedia applications is hierarchical and considers the four essential elements of the authoring model: application, group, scene and media object. All these elements are modeled as processes that evolve according to previously established synchronization relationships. The way of formally represent multimedia applications commented in this section is based on the approach presented in [8]. Further details are presented in the following topics.

3.1 Data Representation and Root Process Instantiation

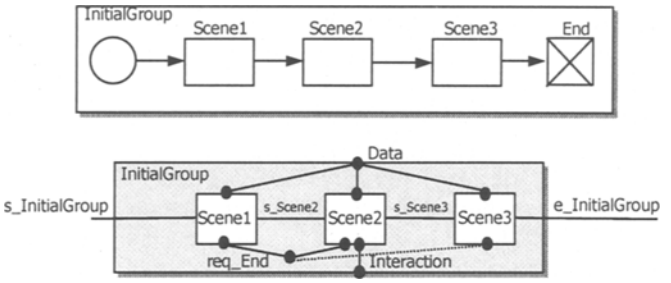
Data representation is done by means of a library called *classes*, which define data types for all possible media objects. There are types like BitmapClass, StreamClass and SwitchButtonClass, whose definition is based on their respective MHEG-5 classes. For example, the fields of BitmapClass are the media object, its position in the output device and its dimensions. The application is started from the instantiation of the root group process. After that, the application is indeed able to evolve.

3.2 Group Representation

In the representation of groups, the hiding operator is used. Taking the example of figure 8, one can see that some internal events like the beginning of both Scene2 (s_Scene2) and Scene3 (s_Scene3) are not visible outside the process (1). These events are used to synchronize the presentation of the scenes belonging to InitialGroup. The synchronization is modeled with the *par* operator (2). For instance, the beginning of Scene2 is associated with the end of Scene1 (s_Scene2) (3 and 4). The same occurs with Scene2 and Scene3: the beginning of the latter is synchronized with the end of Scene2 (s_Scene3) (4 and 5).

The disabling operator must also be mentioned (6). As one can observe, the req_End event reaches all the processes of the group; it is used to model the end of the

application. When it is generated (by a transition to end), groups and scenes are successfully terminated (6).



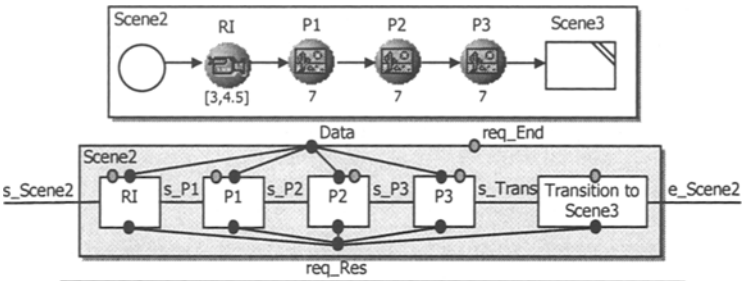

```
process InitialGroupl[s_InitialGroup,e_InitialGroup,Interaction,Data]
  (...,RI:StreamClass,d1:Time,d2:Time,P1:BitmapClass,
   dP1:Time, P2:BitmapClass,dP2:Time,P3:BitmapClass,
   dP3:Time,...):exit is
  hide s_Scene,s_Scene2,s_Scene3, req_End in                          (1)
    s_InitialGroup;
    par s_Scene2#2,s_Scene3#2                                         (2)
      [s_Scene2]→Scene1[s_Scene,s_Scene2,Data,req_End](...)           (3)
      [s_Scene2,s_Scene3]→Scene2[s_Scene2,s_Scene3,Interaction,Data,  (4)
                          req_End] (RI,d1RI,d2RI,P1,dP1,P2,dP2,P3,dP3)
      [s_Scene3]→Scene3[s_Scene3,e_InitialGroup,Data,req_End](...)    (5)
    endpar
    [>req_End;exit                                                    (6)
  endhide
endproc
```

Fig. 8. InitialGroup modeling in E-LOTOS.

3.3 Scene Representation

Scene modeling differs in many aspects from group representation. One of the differences is that scene processes instantiate media objects and restrictions instead of groups and scenes. The presence of the *loop* operator in the representation is another important difference (1) (see figure 9). It is used to allow a scene to be presented more than once, which may happen when the application is logically organized as a net. Figure 9 shows Scene2, previously instantiated in figure 8.

The req_Res event is responsible for restarting the media objects of the current scene when a transition to another scene occurs. The code that models a scene transition is composed of three events: s_Trans, req_Res and f_Scene (see figure 10a). The former denotes the occurrence of the transition. The second invokes the media objects of the scene to be reset. The third one indicates the end of the scene presentation. As the transition is an endless process, it is also disabled by the occurrence of the req_End event. When the transition is to the end of the application, the req_Res event is replaced by the req_End event (see figure 10b).




```
process Scene2 [s_Scene2,e_Scene2,Data,req_End]
  (RI:StreamClass,d1:Time,d2:Time,P1:BitmapClass,dP1:Time,
   P2:BitmapClass,dP2:Time,P3:BitmapClass,dP3:Time)
  hide s_Scene,s_P1,s_P2,s_P3,s_Trans,req_Res in
    loop forever in                                          (1)
      s_Scene2;
      par s_P1#2,s_P2#2,s_P3#2, s_Trans#2
        [s_P1]→RI [s_Scene, s_P1,Data,req_End,req_Res] (RI,d1,d2)
        [s_P1,s_P2]→P1[s_P1,s_P2,Data,req_End,req_Res](P1,dP1))
        [s_P2,s_P3]→P2[s_P2, s_P3,Data,req_End,req_Res](P2,dP2))
        [s_P3,s_Trans]→P3[s_P3,s_Trans ,Data,req_End,req_Res](P3,dP3))
        [s_Trans]→Transition[s_Trans,e_Scene2,req_End,req_Res]()
      endpar
    endloop
    [> req_End;exit
  endhide
endproc
```

Fig. 9. Representation of Scene2

```
process Transition [s_Trans,f_Scene,req_End,req_Res]:exit is
  loop forever in
    s_Trans;req_Res;f_Scene
  endloop
  [> req_End;exit
endproc
```

(a) Scene transition

```
process Transition [s_Trans,f_Scene,req_End]:exit is
  s_Trans,req_End;f_Scene;exit
endproc
```

(b) Transition to the end of the application

Fig. 10. Representation of transitions

3.4 Basic Objects and Temporal Restrictions

Basic or monolithic objects were defined by [3] and model the presentation of simple media objects. These media objects are defined by the occurrence of synchronous (beginning, end) and asynchronous (user interaction) events. Several combinations of these events can be formulated, but only eight are pointed as important in the definition of interactive multimedia scenes. This work presents three of these combinations (see table 1). The fourth object presented in this table (pSsSe - Synchronous start Synchronous end) does not appear in [3]. It allows time-dependent media with both minimum and maximum presentation durations to be modeled. In the definition of the processes, the Data event was used to represent the presentation of the media object.

E-LOTOS Code	*Description*
Synchronous start Synchronous end	
process pSsSe[start,end,Data:class] (media:class,d:time):exit is start;Data(!media);wait(d);end@t[t=0];exit endproc	Used to model time-independent media objects like image and text with a known presentation duration.
Synchronous start Asynchronous maximum end	
process pSsAme[start,end,user,Data:class] (media:class,d1,d2:time):exit is start; Data(!media);wait(d1) ; (user@t[t<=d2] [] wait(d2);exit);end@t[t=0] ;exit endproc	Used to model user interaction; if the interaction does not occur during the interval [d1,d2] the process is finished when the maximum time (d2) is reached.
Synchronous start Asynchronous end	
process pSsAe[start,end,user,Data:class] (media:class,d:time) : exit is start;Data(!media);wait(d);user;end@t[t=0];exit endproc	Modeling of user interaction without a maximum time to wait defined. The process finishes only when the interaction occurs.
Synchronous start Synchronous maximum end	
process pSsSme[start,end,Data:class] (media:class,d1,d2:time) : exit is start;Data(!media);wait(d1);end@t[t≤d2];exit endproc	Used to model time-dependent media objects like audio and video, which have a minimum and a maximum duration defined.

Table 1. Representation of basic objects

Figure 11 shows the representation of P2, which appeared in the definition of Scene2 in figure 9. The event req_End (3) can again be observed, because media objects are also always being executed (1); if there is a loop in the scene definition, some media objects may be executed more than once during the presentation of the scene. In the same figure, one can also see the effect of the occurrence of the req_Res event: the restart of the media object to its initial state (2).

```
process P2 [s_P2,e_P2, Data,req_End,req_Res]:exit is
  (P2:BitmapClass,dP2:Time)
  loop forever in                                    (1)
    pSsSe[s_P2, e_P2, Data] (P2,dP2)
    [>req_Res                                        (2)
  endloop
  [> req_End;exit                                    (3)
endproc
```

Fig. 11. Representation of the media object P2

The authoring model and consequently the tool, to be described in the next section, provide the definition of three distinct temporal restrictions: WaitMaster, WaitLatest and WaitEarliest. Their E-LOTOS representation controls the end of the media objects presentation that converge to the synchronization point. Restrictions are not implemented in libraries because their behaviour depends on the number of media objects that converges to the synchronization point. Figure 12 shows the representation of the WaitEarliest restriction.

```
process WaitEarliest[e_A,e_B,e_C, e_Restriction,req_End,req_Res]:exit is
  loop forever in
    (e_A[]e_B[]e_C);e_Restriction@t[t=0]
    [>req_Res
  endloop
  [>req_End;exit
endproc
```

Fig. 12. Representation of the restriction WaitEarliest

4 The Authoring Environment

The creation environment is divided in two units: media repository and specification area. At any moment, the user can insert media objects into the repository. This is done by browsing a local media object or referencing a remote one. Figure 13 shows two windows that allow, respectively, the incorporation of new media objects (*New Media*) to the application and the manipulation (*Medias Palette*) of already existent ones.

Fig. 13. Management of the media objects of the application

The specification area is composed of the scenes and groups of an application. Each scene is represented by both the temporal and the spatial views. The former allows the user to insert icons and synchronization points and to establish their relationship using arcs. The visible elements, used in the temporal synchronization, can be adjusted in the spatial view.

Figure 14 shows the basic functionality of the authoring environment. The toolbar has shortcuts to its main functions (1). Two support windows can be observed: Specification Hierarchy (2) and Icons Palette (3). The former provides the user a general view of the application, presenting all the scenes and groups in a tree and providing a structured view of the relationships among them. In this case, the modeled application was the example presented in section 2 and is composed, therefore, of three scenes: Scene1, Scene2 and Scene3 (4). The latter, in its turn, provides mechanisms to visualize and edit the icons properties. In the same figure, the bitmap icon (P1) of Scene2 is selected and its specific properties are presented in the mentioned window. Icons that have an associated media object (audio, text, image, and video) present a special property called *media*. This property must be filled with a

media object existing in the repository. In this example, icon P1 is associated to the media object *Rio Bonito*.

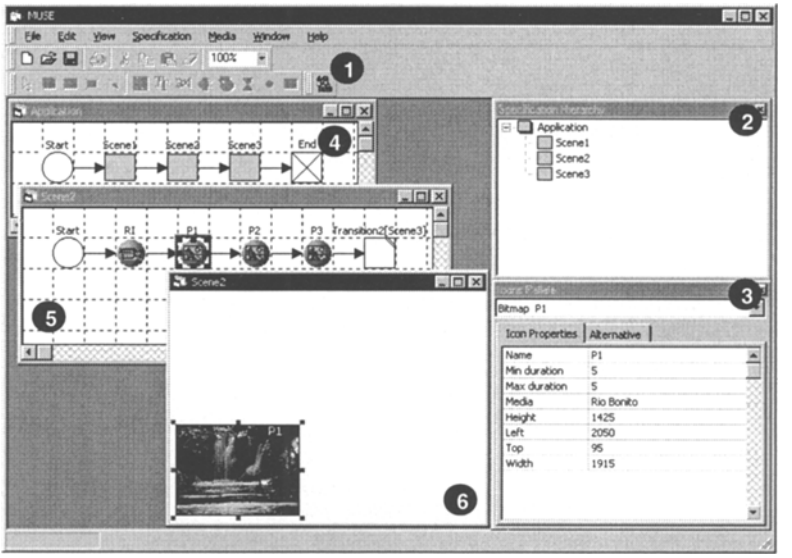


Fig. 14. Interface of the authoring environment

In figure 14, one can also observe the specification of Scene2 (5). It is composed of video (RI) followed by the sequential presentation of three images (P1, P2 and P3). By the end of the presentation of the last media object, a transition to Scene3 occurs. These information are presented by the temporal view. At the same time, the spatial view of Scene2 taking the icon P1 as reference is showed (6). It is possible to move or resize the visible media objects. Their properties related to coordinates and dimensions are automatically updated.

Time-dependent media objects, like video, can be divided in smaller segments, allowing the synchronization of other elements with specific points of them. The environment provides mechanisms that make the process of fragmentation of these media objects easy. MUSE also provides means to reuse scenes and groups. It can be done by browsing the group or scene to be retrieved. It is necessary to redefine the transitions, defining where the application should evolve to after its presentation. Finally, it is also important to highlight the functionality of E-LOTOS code generation. This is obtained through the special saving option *Save as E-LOTOS*.

5 Conclusions and Future Work

This work initially proposed a new model for specifying interactive networked multimedia applications. Besides, mechanisms for mapping this model to the E-LOTOS language were presented. Finally, the developed environment was described. The main contribution of this work is, therefore, the construction of an environment

turned to both ease of use and good expressiveness. At the same time, means to provide the formal representation of applications aiming at its analysis is also a great contribution.

The model proposed distinguishes intentionally the concepts of logical structuring and temporal synchronization. The logical structure of the applications facilitates its organization in chapters, sections or in any other unit. For this reason, the application becomes modular, which contributes to lessen the complexity of the scenes and to avoid the occurrence of the state explosion problem.

Future works include the creation of mechanisms that allow the user to define in the own environment parameters of quality of service, which will be used during the execution of the application. The possibility to define alternative media objects is also an important future task.

It is important to highlight that the use of this environment integrated to the other tools under development in the project provides a complete framework, covering all the steps involved in the design of distributed multimedia applications: specification, verification and presentation. The ease of the authoring model and the use of a formal description technique to validate the applications turn the environment attractive and easy to use, without restricting the expressiveness of the environment.

References

1. G. Blakowski and R. Steinmetz. A Media Synchronization Survey: Reference Model, Specification, and Case Studies. IEEE Journal on Selected Areas in Communications, 14(1): 5-35, January 1996.
2. ISO/IEC DIS 13522-5. Information Technology - Coding of Multimedia and Hypermedia Information, Part 5: Support for Base-Level Interactive Applications, 1995.
3. N. Hirzalla, B. Falchuk and A. Karmouch. A Temporal Model for Interactive Multimedia Scenarios. IEEE Multimedia, 24-31, fall 1995.
4. L. Soares e R. Rodrigues. Autoria e Formatação Estruturada de Documentos Hipermídia com Restrições Temporais. In Proc. of the 3rd Workshop on Multimedia and Hypermedia Systems, São Carlos, Brazil, May 1997. (In Portuguese)
5. P. Sénac, R. Willrich, P. de Saqui-Sannes. Hierarchical Time Stream Petri Nets: A Model for Hypermedia Systems. In Application and Theory of Petri Nets, 1995.
6. P. Sénac, M. Diaz and P. de Saqui-Sannes. Toward a formal specification of multimedia synchronization scenarios. Ann. Telécommun. No. 49, pp 297-314.
7. ISO/IEC JTC1/SC21/WG7. Enhancements to LOTOS. Revised Working Drafts on Enhancements to LOTOS (V4), Project WI 1.21.20.2.3, January 1997.
8. J. P. Courtiat and R.C. de Oliveira. Proving Temporal Consistency in a New Multimedia Synchronization Model. ACM Multimedia, Boston, 1996.