JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 24, 1377-1395 (2008)

Extension of Timeline-based Editing for Non-deterministic Temporal Behavior in SMIL2.0 Authoring*

CHUN-CHUAN YANG, CHEN-KUEI CHU AND YUNG-CHI WANG
Department of Computer Science and Information Engineering
National Chi Nan University
Puli, 545 Taiwan

Timeline-based editing provides nonprofessional users an intuitive and friendly way for multimedia authoring, but the schedule-based and deterministic property of timeline results in the lack of the ability for supporting non-deterministic temporal behavior, which is one of the key features of SMIL2.0. This paper presents our elaborate effort of supporting non-deterministic temporal behavior by timeline-based editing. The concept of Dividable Dynamic Timeline (DDTL) is proposed in the paper, which includes two novel features: dividable timeline and dynamic section. With DDTL, authors can create interactive multimedia presentations while enjoying the convenience of timeline. Mechanisms of converting from DDTL editing results to SMIL2.0 and the reuse of SMIL2.0 scripts are presented in the paper. By the reuse of existing SMIL2.0 scripts and the flexible features of DDTL, an efficient and friendly authoring environment for SMIL2.0-based interactive multimedia presentations can be provided. Implementation of the system provides a friendly WYSIWYG environment and multiple views/windows are provided by the systems to help SMIL2.0 authors compose multimedia presentations efficiently.

Keywords: SMIL2.0, multimedia authoring, interactive presentation

1. INTRODUCTION

Synchronized Multimedia Integration Language (*SMIL*) [1-3] developed by *WWW Consortium* (*W3C*) provides Internet users a mechanism to compose multimedia documents. With SMIL, authors can create multimedia presentations integrating video, audio, animation, image, text, *etc*. There are two versions of SMIL specification that had been released. The current version of SMIL (SMIL2.0 and up) enhances the previous scheduled-based version (SMIL1.0) by a strong support of user interaction with a declarative event-based timing.

The direct way to create a SMIL document is to use a text editor and start writing SMIL tags as most of the programmers do. For nonprofessionals, it is much better to have an authoring system that helps users compose SMIL documents in a visualized (WYSIWYG) way. Two major categories of visualized SMIL1.0 authoring are (1) structure-based editing, and (2) timeline-based editing. Structure-based editing is primarily based on the visualization of SMIL temporal relations (*i.e.* <seq> and <par>), and users need to organize nested <seq> and <par> blocks. On the other hand, timeline-based editing hides the language structure of SMIL by visualizing the playback time and duration

Received November 17, 2006; revised July 17, 2007; accepted March 6, 2008.
Communicated by Chung-Sheng Li.
* Preliminary versions of the paper were published in the proceedings of IEEE ICME 2004 [29, 30].

of each object in the timeline manner providing users a more intuitive way to understand and easily control the timing of each object.

In our previous work, we proposed an efficient modeling technique [4] for SMIL1.0, and based on that technique, we designed and implemented a friendly and powerful timeline-based SMIL1.0 authoring system called *SMILAuthor* [5]. In light of easy learning of timeline-based editing, we intended to extend our effort in SMIL1.0 authoring to support SMIL2.0. However, the feature of event-based timing in SMIL2.0 introduces non-deterministic temporal behavior in a presentation which means the accurate playback time (and duration) of some media objects as well as the total length of the presentation cannot be determined before run-time. Apparently, the original timeline-based scheme cannot support authoring of non-deterministic temporal behavior. Thus, two novel features, namely *dividable timeline* and *dynamic section*, are proposed in this paper to support non-determinism in timeline-based editing. The new editing scheme is thus called *Dividable Dynamic Timeline-based* (*DDTL-based*) authoring.

Moreover, in order to reuse a SMIL2.0 script in the authoring process, the script must be converted to the form of DDTL. There are two steps involved in the conversion. First, the script is converted to our previously proposed model namely *Extended Real-Time Synchronization Model* (*E-RTSM*) [6], which provides a systematic view for the temporal information in the script. In the second step, the temporal information of each object in the script is extracted by processing E-RTSM and represented in the form of DDTL.

The rest of the paper is organized as follows. Definition of E-RTSM is presented in section 2. The basic concept of DDTL as well as the conversion mechanisms from DDTL to SMIL2.0 are presented in section 3. Reuse of existing SMIL2.0 scripts in DDTL-based authoring are presented in section 4. Implementation of the proposed authoring system is presented in section 5. Related work of the paper is discussed in section 6. Finally, section 7 concludes this paper.

2. EXTENDED REAL-TIME SYNCHRONIZATION MODEL (E-RTSM)

The previous version of E-RTSM, RTSM was proposed to address the lack of *Petri net* based models such as *OCPN* (*Object Composition Petri Net*) for dealing with real-time synchronization. There are two kinds of places in RTSM, *regular places* and *enforced places*. The firing rule of RTSM specifies that once an enforced place becomes unblocked (*i.e.* related action associated with the place is completed), the following transition will be immediately fired regardless the states of other places feeding the same transition. With the enforced firing rule, temporal relationship of objects in a SMIL1.0 document can be easily represented by RTSM. Please refer to our previous work [4, 7] for more detailed definition, properties, and application of RTSM.

E-RTSM was proposed to equip RTSM with the ability of modeling event-based timing in SMIL2.0. Major differences between SMIL1.0 and SMIL2.0 in timing control include: (1) Values of <*begin*> and <*end*> attributes for an object (or time containers <*par*> and <*seq*>) can be non-deterministic events, *i.e.* Events with unknown occurring times such as *Mouse-Click* events or *Key-Pressed* events. (2) Multiple values for <*begin*> and <*end*> attributes are allowable for media objects and time containers, *i.e.* the start or

the end of a SMIL2.0 object can be controlled by more than one event. (3) Some complicated synchronization features such as <*restart*> and <*min/max*> attributes are also defined in SMIL2.0.

In order to cope with the non-deterministic synchronization behaviors of SMIL2.0, two new features are added in E-RTSM: (1) allowing a place in E-RTSM to be mapped to a non-deterministic event (denoted by a "?" in a place). (2) *Run-time controllers* for complicated synchronization features are defined.

Introducing association of non-deterministic events with E-RTSM places increases the flexibility of the model, but it also increases the difficulty in processing the model, such as the estimation of the firing time of each transition. On the other hand, a non-deterministic event is normally associated with an enforced place in the application of converting SMIL2.0 scripts to E-RTSM. However, from the viewpoint of modeling, a regular place can also be mapped to a non-deterministic event, and in such case the non-deterministic event is not dominating the firing of the following transition.

Run-time controllers are used to model complicated timing features in SMIL2.0 that are difficult or impossible to be represented by the combination of other basic elements (arc, transition, place). A run-time controller can be placed in between any two transitions (the start transition and the end transition) as places in E-RTSM. *Bi-directional arcs* are used to connect the start transition to the run-time controller and the run-time controller to the end transition. A run-time controller is associated with a set of rules that control the firing of the start and the end transitions. Therefore, the operation of a run-time controller overrides the operation of the places/transitions in between the start and the end transitions of the run-time controller.

By using run-time controllers in E-RTSM, handling of these complicated timing features is delayed until run-time rather than the modeling (parsing) phase. Three run-time controllers have been defined in E-RTSM: *Restart* controller, *Min* controller, and *Repeat* controller. Introduction of run-time controllers in E-RTSM brings some convenience in synchronization modeling. However, due to the inherent limitation of timeline-based editing, the proposed extension of timeline-based editing can not support the complicated temporal relationship presented by the run-time controllers. Definition of E-RTSM is given as follows:

Definition E-RTSM is a 10-tuple $\{T, P, E, \underline{R}, A, \underline{B}, D, M, \underline{N}, X\}$, where (*Note that the differences between E-RTSM and RTSM are underlined*.)

$T = \{t_1, t_2, \ldots, t_n\}$	Transitions
$P = \{p_1, p_2, \ldots, p_m\}$	Regular places (single circles)
$E = \{e_1, e_2, \ldots, e_k\}$	Enforced places (double circles)
$S = P \cup E$	All places
$R = \{r_1, r_2, \ldots, r_i\}$	Run-time controllers
$A = \{T \times S\} \cup \{S \times T\}$	Unidirectional arcs
$B = \{T \times R\} \cup \{R \times T\}$	Bi-directional arcs
$D = S \rightarrow$ *Real number*	Time duration of places
$M = S \rightarrow \{m_1, m_2, \ldots, m_j\}$	Regular types of medium
$N = S \rightarrow \{n_1, n_2, \ldots, n_t\}$	Non-deterministic events
$X = S \rightarrow \{0, 1, 2\}$	State of places

Each place may be in one of the following states:
0: no token
1: token is blocked A "cross" in the place
2: token is unblocked A "dot" in the place

The firing rules of E-RTSM are the same as those of RTSM in the absence of run-time controllers. When a run-time controller is presented between two transitions, the firing of the transitions (the start transition and the end transition) is controlled by the run-time controller, which can override the firing rules associated with places.

3. DIVIDABLE DYNAMIC TIMELINE (DDTL)

Normally authors must specify the exact playback time and duration of each object in timeline-based editing. According to an object's playback time and duration, a time-line segment representing that object is displayed at corresponding position on the time axis. In order to introduce non-deterministic temporal behavior in timeline and decide to what extent timeline-based editing can do for SMIL2.0 authoring, we need to investigate the synchronization characteristics in SMIL2.0.

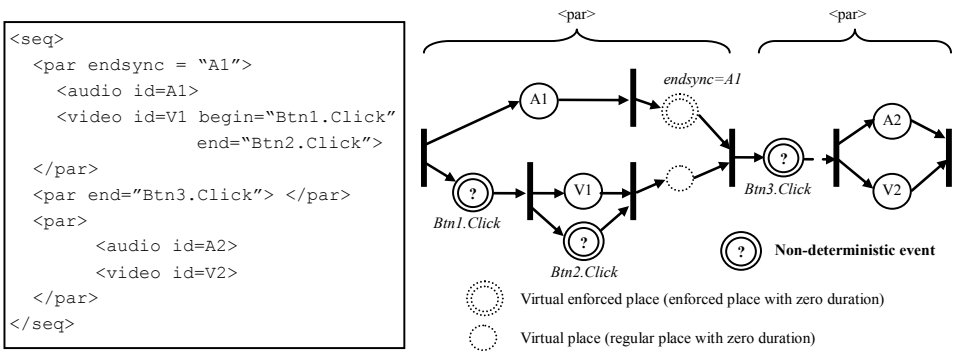


Fig. 1. A sample SMIL2.0 code snippet and its corresponding E-RTSM model.

Fig. 1 shows a sample SMIL2.0 code snippet and its corresponding E-RTSM model. It is easy to understand from the model that non-deterministic temporal behavior in the code snippet comes from the non-deterministic events (denoted by a double circle with a "?"). Moreover, two types of the non-deterministic events are identified in the E-RTSM model: splitting event and non-splitting event. An event is a splitting event when removing the event results in two separate parts in the model. The player must wait for the occurrence of a splitting event before it can continue playing the rest of the presentation. Event Btn3.Click in Fig. 1 is a splitting event. Events Btn1.Click and Btn2.Click are non-splitting events.

Inspired by the two different types of non-deterministic events, we introduce two novel features in timeline-based editing: *dividable timeline* (from the idea of splitting event) and *dynamic section* (from the idea of non-splitting event), which are explained respectively in the following subsections.

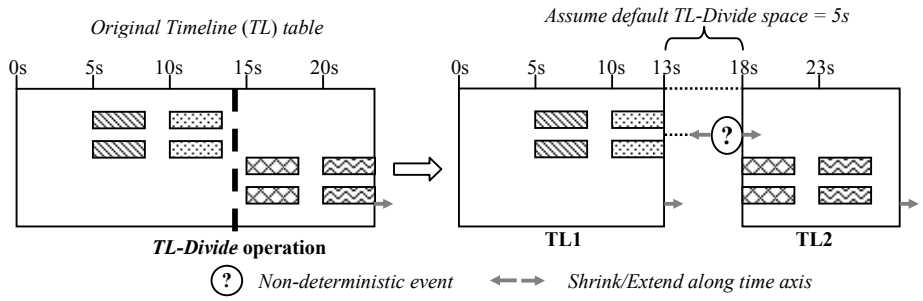


Fig. 2. TL-Divide operation.

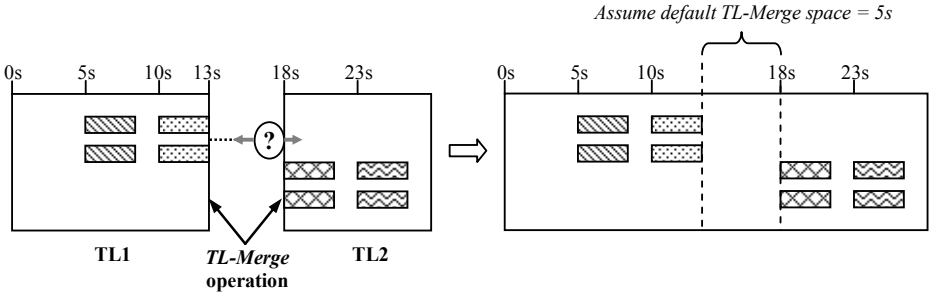


Fig. 3. TL-Merge operation.

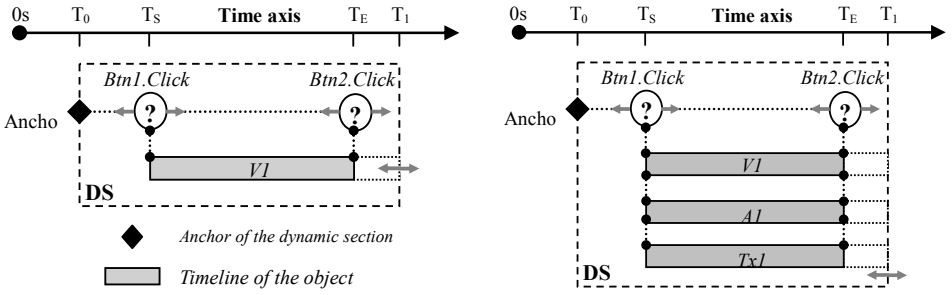
3.1 Dividable Timeline

We define *TL-Divide* operation enabling users to divide a timeline table into two sequentially separated timeline tables and associate a non-deterministic splitting event (*e.g.* user's mouse action) with the beginning of the latter timeline table. An example of TL-Divide operation is shown in Fig. 2. To divide a timeline table, the author has to specify a proper cutting point on the time axis such that no objects will be cut into two pieces. Moreover, in order to properly display information of the time axis for the latter timeline table, we need to choose a preset (default) value (*e.g.* 5s in Fig. 2) for the occurrence time of the splitting event. The preset value is only for reference in authoring stage since the exact occurrence time of the event is unknown before run-time. For more flexibility, the authoring system should allow users to change the reference occurrence time of the event as shown in Fig. 2.

The reverse of TL-Divide is *TL-Merge* operation. The author uses the operation to remove the non-deterministic event and merge two separated timeline tables. Fig. 3 gives an example of TL-Merge operation. Note that the default space in time for TL-Divide and TL-Merge depends on the content of the document being composed. Thus, in addition to using a general default value, the authoring system should also provide a friendly way for the user to dynamically change the value of the space.

3.2 Dynamic Section

A dynamic section (DS) is defined for a dynamic object whose beginning and/or



(a) For a single object. (b) For a group of parallel objects.

Fig. 4. Dynamic section in a timeline table.

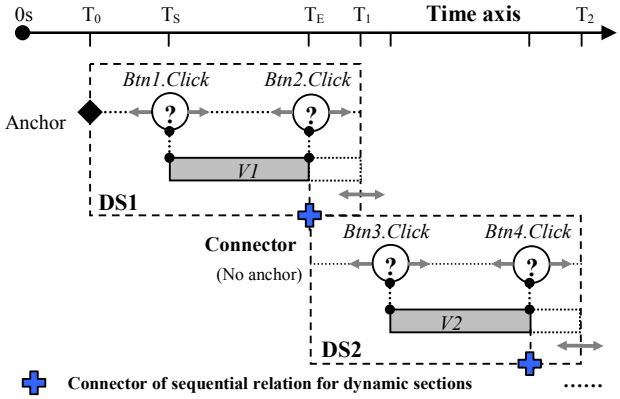


Fig. 5. Connecting two dependent DS.

ending are triggered by non-splitting events. As shown in Fig. 4 (a), to create a DS for an object, the user has to specify the position (anchor) and the length of the DS on the time axis. Two events are associated with beginning and ending of the object in the DS. That is, DS defines the time range of playback for an object, but the actual starting time and ending time of the object are not determined until run-time. Thus, the positions of the two events on the time axis are only for reference in authoring stage. As illustrated in Fig. 4 (b), DS can also control a group of parallel objects such that the group acts just like a single object in the DS. The authoring system should provide the option to disable/enable events in a DS for more flexibility.

Given that the author may expect a dynamic object depending on a former dynamic object. We propose a DS operator called *connector* to support dependency between different dynamic sections. As illustrated in Fig. 5, beginning of DS2 depends on DS1's ending, and DS1's ending depends on event *Btn2.Click*. By using DS connector, the author can easily create a sequence of dependent dynamic objects.

3.3 Converting DDTL to SMIL2.0

The editing result of DDTL needs to be converted to SMIL2.0 format and saved as a SMIL document for future playback on a SMIL player. We had developed a converting

```
<seq>
      <TL1>
      <par end="Btn.Click"> </par>
      <TL2>
</seq>
```

Fig. 6. SMIL code snippet for TL-Divide.

```
<par begin= "T0-0s"end="Btn2.Click;
T1-0s">
     <video id=Vl begin="Btn1.Click" >
</par>
/*** note that DS's time base is 0s ***/
```

(a) Single object in the DS.

```
<par begin="T0-0s"end="Btn2.Click; T1-0s">
     <par begin="Btn1.Click">
          <video id=Vl>
          <audio id=A1>
          <text id=Tx1>
     </par>
</par>
/*** note that DS's time base is 0s ***/
```

(b) Group of objects in the DS.

Fig. 7. SMIL code snippet for the DS in Fig. 4.

algorithm from deterministic timeline data to SMIL1.0 in our previous work [5]. We only present the converting mechanisms for dividable timeline and dynamic section in this paper, and the mechanisms can be easily integrated into the original converting algorithm in our previous work.

Since the two separated timeline tables resulted from TL-Divide operation have a sequential relation, a parent <*seq*> time container is created for the two timeline tables, and a <*par*> time container with its <*end*> attribute set as the non-deterministic event is added in between the two timeline tables. Fig. 6 shows the code snippet for the result of TL-Divide in Fig. 2.

To convert a DS to SMIL2.0, we need to know that there are two run-time cases ending a DS as well as the dynamic object in the DS: (1) the viewer triggers the ending event for the object, or (2) no ending event triggered but the preset ending time of the DS (which is according to the length of the DS) is reached. Therefore, in the conversion, we need to create a parent time container <*par*> for the object and set the <*end*> attribute of the <*par*> to reflect the above two cases. Fig. 7 (a) shows the code snippet for the DS in Fig. 4 (a), in which the <*end*> attribute of the parent element <*par*> is a list of two cases to end the DS (*i.e.* event *Btn2.Click* or time T_1 is reached). The <*begin*> attribute in the parent <*par*> specifies the location of the DS on the time axis, and the <*begin*> attribute of the object defines the triggered event to start the object. Similarly, the conversion of the DS containing a group of objects (Fig. 4 (b)) is shown in Fig. 7 (b).

For a sequence of dynamic sections connected by DS connectors, we need to create a parent <*seq*> for all DS in the sequence. The code snippet for a sequence of DS (Fig. 5) is displayed in Fig. 8. Please note that the time base of the first DS (DS1, with an anchor) and the time base of the other DS (DS2 and the following) are different. Since DS1 is the first child element in the <*seq*> element, DS1's time base is the begin time of the <*seq*> element. DS2 is the succeeding element of DS1 in the <*seq*> element, thus DS2's time base is the end of DS1, which is triggered by either the *Btn2.Click* event or the preset ending time T_1. A sample DDTL editing result is displayed in Fig. 9. The corresponding SMIL2.0 code snippet for that sample is shown in Fig. 10. Note that the code snippet in Fig. 10 only serves for demonstrative purpose, and some of the mandatory attributes such as "region" and "src" in the example are removed for higher compactness and readability.

```
<seq>
  <par id=DS1 begin=T0-0s end="Btn2.Click; T1-0s">
       <video id=V1 begin="Btn1.Click">
  </par>
  <par id=DS2 end="Btn4.Click; T2-TE (DS2 Length)">
       <video id=V2 begin="Btn3.Click">
  </par>
  ……  /* for following DS, if any */
</seq>
  /*** DS1's time base is 0s ***/
/*** DS2's time base is the end of DS1 ***/
```

Fig. 8. SMIL code snippet for the sequence of DS in Fig. 5.

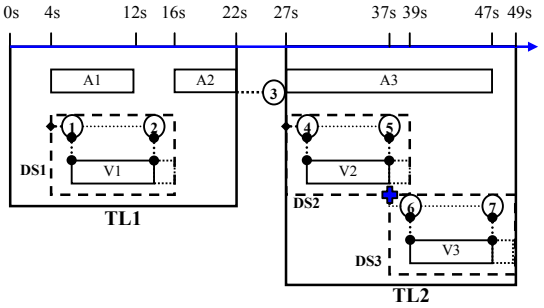


Fig. 9. Sample DDTL editing result.

```
<seq>

  <par id=TL1>
    <seq>
      <audio id=A1 begin="4s" dur="12s-4s">
      <audio id=A2 begin="16s-12s" dur="22s-16s">
    </seq>
    <par id=DS1 begin="4s", end="Btn2.Click;16s">
      <video id=V1 begin="Btn1.Click">
    </par>
  </par>

  <par end="Btn3.Click"> </par>

  <par id=TL2>
    <audio id=A3 begin="0s" dur="47s-27s">
    <seq>
      <par id=DS2 begin="0s" end="Btn5.Click;39s-27s">
           <video id=V2 begin="Btn4.Click">
      </par>
      <par id=DS3 end="Btn7.Click;49s-37s">
           <video id=V3 begin="Btn6.Click">
      </par>
    </seq>
  </par>

</seq>
```

Fig. 10. SMIL code snippet for the example in Fig. 9.

4. REUSE OF SMIL2.0 SCRIPTS

To reuse a SMIL2.0 script in DDTL-based editing, the script must be converted to elements in DDTL, which include original timeline segment, dividable timeline, and dynamic section. Two steps are involved in converting a SMIL2.0 script to DDTL: (1) converting the script to our previously proposed E-RTSM, and (2) Extracting DDTL

```
<seq>

  <par id=TL1>
    <seq>
      <audio id=A1 begin="4s" dur="8s">
      <audio id=A2 begin="4s" dur="6s">
    </seq>
    <par id=DS1 begin="4s">
      <video id=V1 begin="Btn1.Click" end="Btn2.Click;12s">
    </par>
  </par>

  <par end="Btn3.Click"> </par>

  <par id=TL2>
    <audio id=A3 dur="20s">
    <seq>
        <par>
            <video id=V2 begin="Btn4.Click" end="Btn5.Click;12s">
        </par>
        <par>
            <video id=V3 begin="Btn6.Click" end="Btn7.Click;12s">
        </par>
    </seq>
  </par>

</seq>
```

Fig. 11. A sample SMIL2.0 code snippet.

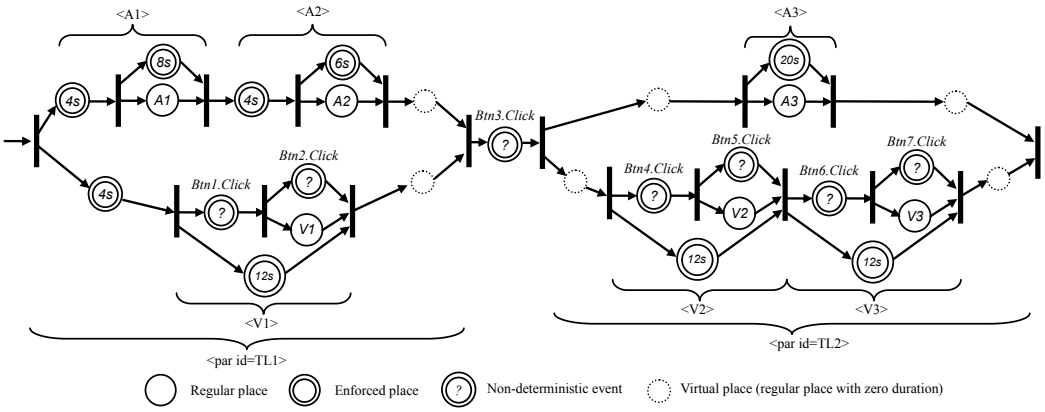


Fig. 12. E-RTSM model for the sample SMIL code snippet in Fig. 11.

elements from E-RTSM. Since E-RTSM and related converting algorithm were proposed in our previous work [6], we only give a typical example for the conversion from SMIL2.0 to E-RTSM and focus on converting E-RTSM to DDTL. A sample SMIL2.0 code snippet is shown in Fig. 11. The corresponding E-RTSM for that sample is displayed in Fig. 12.

For an input SMIL script to be reused in timeline-based authoring, the playback time and duration of each object in the script must be determined. According to an object's playback time and duration, a timeline segment representing that object is displayed at corresponding position on the time axis. In our previous work for SMIL1.0 authoring, we had developed the mechanisms to calculate the playback time and duration (deterministic values) for each object in an input SMIL1.0 script. However, the previously proposed mechanisms cannot be directly applied in the case of SMIL2.0, since part

of the objects in the script may have non-deterministic temporal behavior. Therefore, we propose the modified mechanisms to convert E-RTSM to DDTL in this paper.

4.1 Identifying Splitting Events

As mentioned in section 3, DDTL incorporates the non-deterministic temporal behavior with traditional deterministic timeline segments. Non-deterministic temporal behavior in DDTL comes from the features of dividable timeline and dynamic sections, which can be mapped to two types of non-deterministic events in E-RTSM. Since enforced places dominates the firing time of transitions and to provide a better understanding for the playback of the presentation, an E-RTSM model is reduced by removing the regular places that feed into the same transition with one or more enforced places. The result of the reduction is called the *reduced E-RTSM*. Fig. 13 shows the reduced E-RTSM for the model in Fig. 12. An event dividing a timeline table into two separate tables must be a *splitting event* in the reduced E-RTSM. A splitting event is a non-deterministic event (an enforced place with a "?") and when removing the event will divide the reduced E-RTSM into two separated parts.

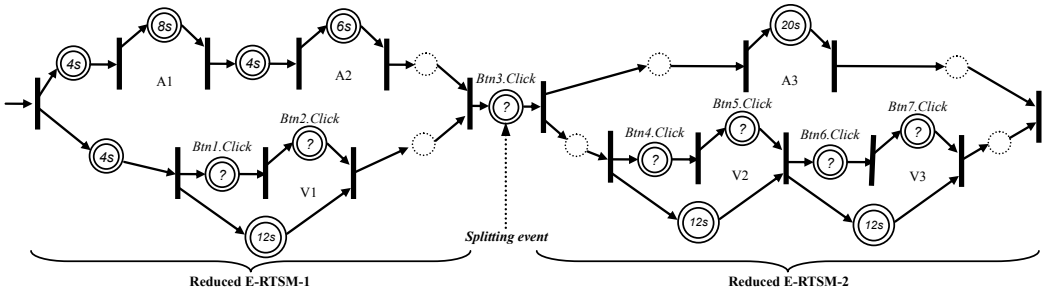


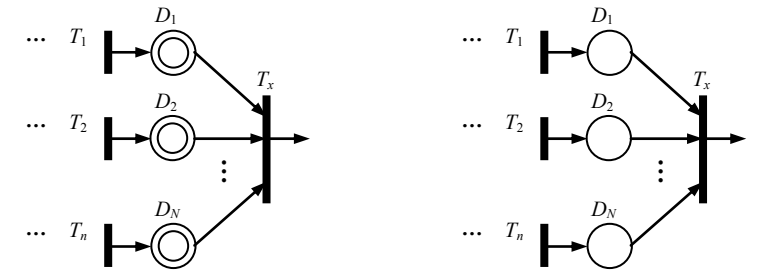
Fig. 13. Reduced sub-E-RTSMs of the model in Fig. 12.

We have to find all splitting events in the reduced E-RTSM in order to properly identify all divided timeline tables in the input script. The simplest way to decide whether an event is a splitting event or not is to temporally remove the event and check the reachability of the end of the model. If the end of the reduced E-RTSM is unreachable when removing an event, the event is a splitting event. For example, *Btn3.Click* in Fig. 13 is a splitting event, and thus two timeline tables (from the two sub-models E-RTSM-1 and E-RTSM-2 respectively) emerge.

4.2 Identifying Dynamic Sections

After the sub-E-RTSM models for different timeline tables are determined, the next step is to traverse these sub-models (in the reduction form) respectively in order to calculate the firing time of each transition. The firing time of the transition right before an object is the starting time of the object, and the firing time of the transition followed by the object is the ending time of object.

There are only two cases for one transition in a reduced E-RTSM: (1) places that



(a) $T_x = \text{Min}(T_1 + D_1, \ldots, T_n + D_n)$. (b) $T_x = \text{Max}(T_1 + D_1, \ldots, T_n + D_n)$.

Fig. 14. Determine the firing time of transition T_x.

E-RTSM-1: 0s

ID	Beginning time	Ending time
*A*1	4s	12s
*A*2	16s	22s
*V*1	4s + *Btn1.Click*	4s + MIN ("*Btn1.Click* + *Btn2.Click*", 12s)

E-RTSM-2: *Btn3.Click*

*A*3	0s	20s
*V*2	*Btn4.Click*	MIN ("*Btn4.Click* + *Btn5.Click*", 12s)
*V*3	END (V2) + *Btn6.Click*	END (V2) + MIN ("*Btn6.Click* + *Btn7.Click*", 12s)

Fig. 15. Beginning and Ending time for each object in Fig. 10.

feed to the transition are all enforced places, or (2) places that feed to the transition are all regular places. For case (1), the firing time of the transition is the minimal value of "the firing time of the preceding transition" plus "the duration of the following place of the preceding transition", which is illustrated in Fig. 14 (a). Fig. 14 (b) shows case (2), in which transition T_x is fired only after all its preceding regular places finish playing. Therefore, for case (2), the firing time of transition T_x is the maximum value of "the firing time of the preceding transition" plus "the duration of the following place of the preceding transition". The duration of each place depends on the type of the media object. For an enforced place of time medium, the duration of the place is the value of the duration. For static media objects, such as <img> and <text>, the duration of the place is zero. For continuous media objects, such as <audio> and <video>, the duration of the place is the implicit duration of the object that is provided by the data server. Since the objects stored in a data server are all pre-orchestrated, it is easy for the data server to obtain the implicit duration of a continuous object. However, for a non-deterministic event, the duration of the enforced place is non-deterministic and is represented by a variable in the firing time calculation.

Fig. 15 shows the beginning time and ending time of each object in the two sub-E-RTSM models in Fig. 13. For those objects (*A*1, *A*2, *A*3) with deterministic beginning time and ending time, a timeline segment is displayed at corresponding position on the time axis in the timeline table. Dynamic sections are used to represent those objects that do not have deterministic beginning time and ending time.

As mentioned in section 2, DS is used to define an object with a beginning event and an ending event. The general mathematical expression for the beginning time of the object in a DS (Fig. 4 (a)) is T_0 + *BeginEvent*, and the ending time of the object, T_0 + MIN (*BeginEvent* + *EndEvent*, *Length of the DS*). (MIN is the function that returns the smaller one from two given variables/values) Therefore, for those objects with beginning time and ending time in the form of above expressions can be represented by DS. For example, objects *V*1 and *V*2 (Fig. 15) are converted to dynamic sections in their respective timeline tables.

For an object whose playback depends on others is converted to a dependent DS. A dependent DS is connected to a former DS by a DS connector, therefore, the general form of the beginning time and ending time for the object in a dependent is as follows:

Beginning: END(the former DS) + *BeginEvent*.
Ending: END(the former DS) + MIN(*BeginEvent* + *EndEvent*, *Length of the DS*).

For example, object *V*3 in Fig. 15 has the form of dependent DS (and its former DS is *V*2's DS). Therefore, a DS connector is created to connect *V*3's DS (DS3) to its former DS (DS2). The final DDTL result of converting the sample code snippet in Fig. 11 is the same as the one in Fig. 9.

4.3 Discussion

As mentioned in section 3, the idea of DDTL comes from E-RTSM. It is easy to know that an object in an E-RTSM model (*e.g*. run-time controllers) cannot always be represented by elements in DDTL. That is, DDTL cannot cover the whole set of temporal non-determinism supported by SMIL2.0, which means for an input SMIL2.0 script there may be some objects that cannot be represented by DDTL elements. Those objects that cannot be DDTL-ized cannot be reused in DDTL-based authoring process. The power of DDTL in dealing with non-deterministic temporal behavior needs to be further explored in the future work of the research.

5. SYSTEM IMPLEMENTATION

We extended the previously implemented system namely *SMILAuthor* to support DDTL-based editing. The new version of the system is thus called *SMILAuthor*2. Implementation of SMILAuthor2 follows a similar concept as proposed in [12] to provide a "WYSIWYG" authoring environment. There are five major windows in the system to provide different views for the currently composing presentation. They are (1) *visual layout window*, (2) *timeline window*, (3) *filter window*, (4) *attribute window*, and (5) *preview window*. The display of the system on the monitor is shown in Fig. 16.

The visual layout window is used for the author to edit spatial relationship required by the presentation. The user could use the window to add, delete, resize, and move regions for the visual layout of the presentation. The timeline window displays the playback duration for objects. The users use the timeline windows to perform DDTL-based editing functions as well as SMIL1.0-related editing functions [9] such as *clear*, *cut*,

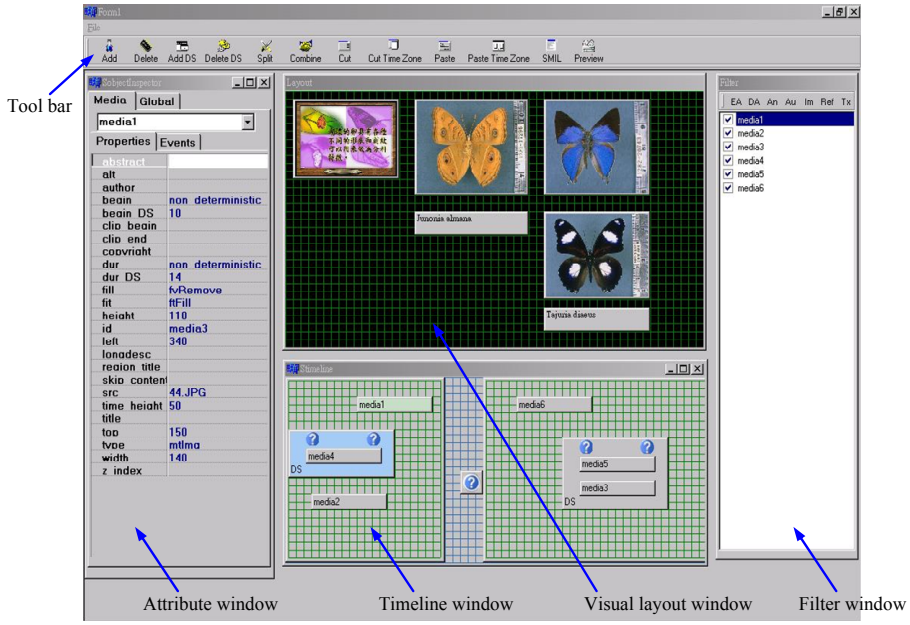


Fig. 16. The display of SMILAuthor2 on the screen.

copy, *paste*, *etc*. In order to reduce the large amount of information that has to display to the author, a filter window is used for the author to set displaying rule for both visual layout window and timeline window. The author could display selected information by specifying either medium type or time duration in the filter window. The attribute window is used to display and modify the attribute information for each object. Finally, the preview window is used to preview the presentation before saving the result to a file. Moreover, the preview window allows the author to preview only part of the presentation by specifying the preview duration, which is called *partial preview function* in the system.

6. RELATED WORKS

Timing issues in multimedia formats had been addressed intensively in the work of Rogge *et al*. [9], in which ten criteria were proposed in their reference model for comparing existing multimedia formats including *SMIL*, *QuickTime*, *Shockwave Flash*, *Realmedia*, *Advanced Streaming Format*, and *MPEG*-4. The authors concluded that SMIL was the only document model supporting all 29 temporal relationships in the reference model. Moreover, SMIL were also designed to have good properties in terms of fine granularity, interactivity with users, extensibility, reusability, adaptability, *etc*. In other words, from the academic viewpoint, SMIL does have what it takes to become one of the most important formats in multimedia presentations.

Multimedia authoring [10-18] have been addressed a lot in the literature for many years. Bulterman and Hardman [10] surveyed selected commercial and research approaches in the context of four different but not mutually exclusive paradigms for au-

thoring multimedia documents: *structured-based*, *timeline-based*, *graph-based* and *script-based*. Although the timeline-based paradigm is easily understood and manipulated for non-interactive, non-adaptive presentations, the authors argued that the structured-based paradigm provides the most useful framework for presentation authoring. Therefore, most of the existing work in multimedia authoring follows the paradigm of structure-based editing. Timeline-based editing in interactive multimedia authoring has not received much of the attention in the literature due to its limitation in supporting complex temporal relationships. Some of the previous authoring systems used the notion of timeline as a supplement in authoring process, in which the timeline view was used only for display, instead of supporting the operations of editing.

Furthermore, most of the authoring systems used proprietary formats in representing multimedia presentations that reduce the popularity of the presentations over WWW. It's worth mentioning that the idea of *partial time chain* proposed in the work of Soares *et al*. [16] is similar to the idea of *dividable timeline* in this paper, since a partial time chain was defined as the time chain in which all its events, except the first one, are predictable in relation to at least one event of the same partial time chain. However, the notion of timeline was used only for display in their work as mentioned above, and the idea of *dynamic sections* in a timeline table was not included in their work.

As the most prestigious commercial product in SMIL authoring, *GRiNS* [19] is an example of a structure-based authoring environment that provides multiple document views. The authoring paradigm incorporated in GRiNS are structure and timeline editing (called *structured timeline*), in which interactive event-based timing is ignored for the timeline (at the authoring stage), objects that start on events are shown as started at the time the event is evaluated. In order to specify the temporal relationship among objects, GRiNS defines different types of containers as displayed in Fig. 17, indicating how child objects are scheduled and activated. By default, GRiNS illustrates the basic structure of the presentation. Although an ordering can be determined, there is no direct representation of the presentation timeline. However, GRiNS provides a timeline view that illustrates the temporal composition of objects once that structure container is activated. In summary, GRiNS does provide a powerful tool for SMIL authoring, but users are required to be equipped with professional knowledge of temporal composition in SMIL for effective use of the product.

Evolved from a structure-based system called *Madeus* [20], *LimSee2* [21-23] (Fig. 18) is an open-source and cross-platform authoring tool for SMIL, in which the timeline paradigm dominates the synchronization control process. Users can adjust media synchronization by moving and resizing the boxes of objects in the timeline view. As pointed out by Bulterman and Hardman [10], the timeline view in LimSee2 does not support asynchronous interaction or adaptive content. There are some elements or attributes in SMIL that can not be edited directly. Editing those elements/attributes is only possible through the hierarchical and attribute views. Moreover, as in GRiNS, the timeline view of LimSee2 adopts a hierarchical structure and the users need to have the knowledge of the synchronization characteristic in SMIL, such as <par> and <seq>. By comparing DDTL with synchronization control in GRiNS and LimSee2, we conclude that DDTL hides the language of SMIL and enjoys more of the benefit of timeline-based editing with partial support of asynchronous (event-based) interaction.

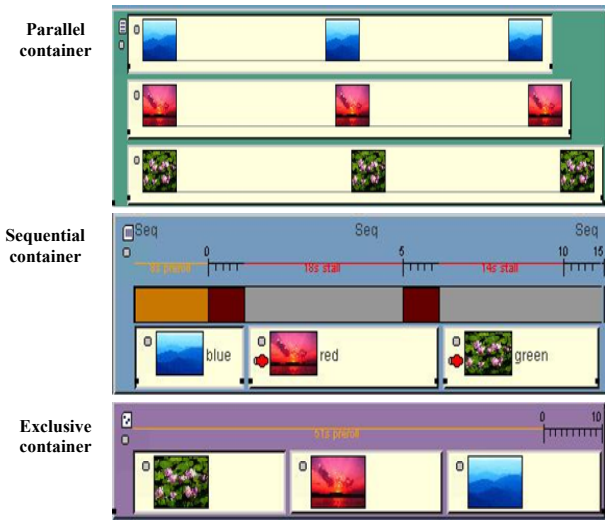


Fig. 17. Examples of GRiNS containers.

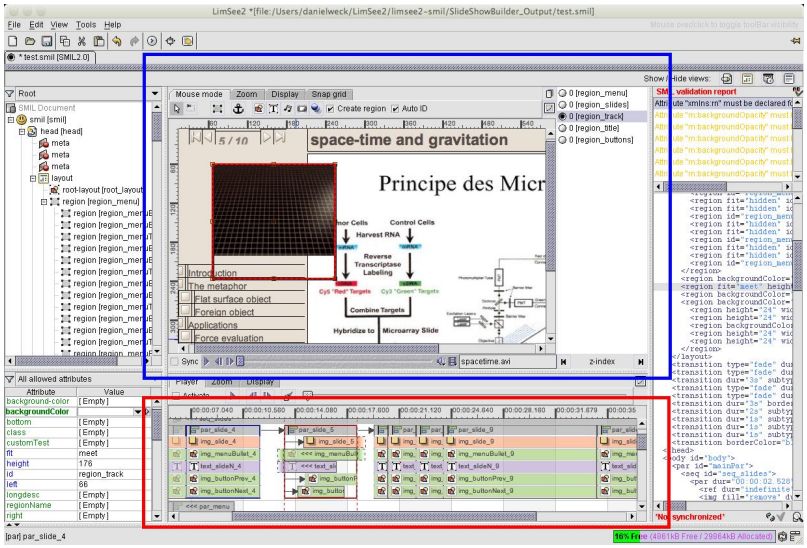


Fig. 18. Snapshot of LimSee2.

Most of the research work supporting SMIL2.0 authoring is non-timeline-based. Sung and Lee [24] developed a collaborative authoring system based on SMIL. Their system provided a unified 3D interface that allowed for simultaneous authoring and manipulation of both the temporal and spatial aspects of a presentation. A timeline-based editor was provided in their system, but their work was mainly focused on SMIL1.0 timing (event-based timing in SMIL2.0 was not addressed). Sampaio *et al*. [25] proposed a *RT-LOTOS* (*Real-Time Language of Temporal Ordering Specifications* [26]) based mechanism for semantic verification of SMIL documents, which is helpful in developing

authoring systems. Non-deterministic events were considered in their work, but their target was mainly SMIL1.0. Chang [27] developed a temporal algebra system to unify media presentation time and interaction event and deal with qualitative and quantitative inconsistency in SMIL2.0 documents, which can be used in semantic verification during paring stage. Editing support in the authoring process was not addressed in their work.

Chung and Pereira [28] proposed a technique based on *Timed Petri Net* (*TPN*) to capture the timing and synchronization information of multimedia objects specified in SMIL2.0. They incorporated a new type of transition (denoted by special transition), a couple of different types of state for places (two tokens can be placed in a place, and each token can be in one of three types), and complicated firing rules in TPN for SMIL2.0 modeling. Although some elaborate features were added in TPN, the authors did not explain how can the proposed scheme model some of the complex timing control behaviors associated with <*restart*>, <*repeatCount*>, and <*repeatDur*> attributes, *etc*. We doubts the feasibility of using a graph-based modeling mechanism such as OCPN, TPN, or even E-RTSM, to completely capture the complex temporal relationships among media objects in SMIL2.0. Moreover, the application of TPN in SMIL2.0 authoring is difficult due to its complicated timing rules.

7. CONCLUSION

Timeline-based editing has the beauty of simplicity and high readability in representing temporal relationships among media objects, and it is easy and friendly for non-professional users in composing multimedia presentations. However, due to its limitation in supporting complex temporal relationships such as event-based timing in SMIL2.0, timeline-based editing has not received much of the attention in the literature. In this paper, we present our effort towards supporting of SMIL2.0 authoring by timeline-based editing. The concept of *Dividable Dynamic Timeline* (*DDTL*) is proposed, which includes two novel features, *dividable timeline* and *dynamic section*, to extend the original timeline scheme with the support of non-deterministic temporal behavior. Mechanisms for converting DDTL data to SMIL2.0 format as well as the mechanisms for reusing a SMIL2.0 script in authoring are presented. DDTL features the easy-learning characteristic of timeline and to some extent allows authors to compose interactive and event-based multimedia presentations. The implementation of extending our previous SMIL1.0 authoring system to support DDTL is also presented.

The future work of the paper is to explore more about the potential of DDTL in supporting SMIL2.0 authoring, in which we will try to identify and define the temporal relationships that DDTL can and cannot achieve. Moreover, impact of DDTL on user friendship in authoring SMIL is also left as the future work.

The contributions of the research in the paper are listed as follows:

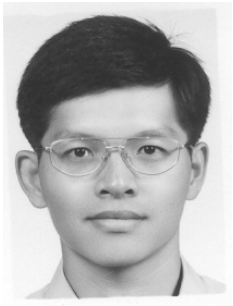
(1) The idea of *dividable timeline* and *dynamic section* to extend timeline-based editing for SMIL2.0 authoring is proposed.
(2) Converting algorithms for DDTL to SMIL2.0 are presented.
(3) Mechanisms for reusing an existing SMIL2.0 script in the DDTL-based authoring process are proposed.

(4) Feasibility of the proposed techniques has been proved by prototype system implementation.

REFERENCES

1. Synchronized Multimedia Integration Language (SMIL) 2.0 Specification, W3C Recommendation, 2001, http://www.w3.org/TR/smil20.
2. D. C. A. Bulterman, "SMIL 2.0 part 1: overview, concepts, and structure," *IEEE Multimedia*, Vol. 8, 2001, pp. 82-88.
3. D. C. A. Bulterman, "SMIL 2.0. 2. examples and comparisons," *IEEE Multimedia*, Vol. 9, 2002, pp. 74-84.
4. C. C. Yang and Y. Z. Yang, "Design and implementation of the just-in-time retrieving policy for schedule-based distributed multimedia presentations," *Journal of Systems and Software*, Vol. 71, 2004, pp. 49-63.
5. C. C. Yang and Y. Z. Yang, "SMILAuthor: an authoring system for SMIL-based multimedia presentations," *Journal of Multimedia Tools and Applications*, Vol. 21, 2003, pp. 243-260.
6. C. C. Yang, Y. C. Wang, and C. W. Tien, "Synchronization modeling and its applications for SMIL2.0 presentations," *Journal of Systems and Software*, Vol. 80, 2007, pp. 1142-1155.
7. C. C. Yang and J. H. Huang, "A multimedia synchronization model and its implementation in transport protocols," *IEEE Journal of Selected Area in Communications*, Vol. 14, 1996, pp. 212-225.
8. M. Jourdan, C. Roisin, and L. Tardif, "Multiviews interfaces for multimedia authoring environments," in *Proceedings of Multimedia Modeling*, 1998, pp. 72-79.
9. B. Rogge, J. Bekaert, and R. van de Walle "Timing issues in multimedia formats: review of the principles and comparison of existing Formats," *IEEE Transactions on Multimedia*, Vol. 6, 2004, pp. 910-924.
10. D. Bulterman and L. Hardman, "Structured multimedia authoring," *ACM Transactions on Multimedia Computing*, *Communication and Applications*, Vol. 1, 2005, pp. 89-109.
11. S. Hudson and C. N. His, "The walk-through approach to authoring multimedia documents," in *Proceedings of the 2nd ACM International Conference on Multimedia*, 1994, pp. 173-180.
12. J. Song, M. Y. Kim, G. Ramalingam, R. Miller, and B. K. Yi, "Interactive authoring of multimedia documents," in *Proceedings of IEEE Symposium on Visual Languages*, 1996, pp. 276-283.
13. J. Freire, R. Lozano, H. Martin, and F. Mocellin, "A STORM* environment for building multimedia presentations," in *Proceedings of the 12th International Conference on Information Networking*, 1998, pp. 329-332.
14. M. Vazirgiannis, I. Kostalas, and T. Sellis, "Specifying and authoring multimedia scenarios," *IEEE Multimedia*, Vol. 6, 1999, pp. 24-37.
15. J. Kim and S. S. An, "Design and implementation of IMAT (Internet Multimedia Authoring Tool) using a unified spatio-temporal relationship model," in *Proceedings of the 3rd IEEE Workshop on Multimedia Signal Processing*, 1999, pp. 617-622.

16. L. F. G. Soares, R. F. Rodrigues, and D. C. M. Saade, "Modeling, authoring and formatting hypermedia documents in the HyperProp system," *Journal of Multimedia Systems*, Vol. 8, 2000, pp. 118-134.
17. M. Vazirgiannis, *et al*, "Interactive multimedia documents: a modeling, authoring and rendering approach" *Journal of Multimedia Tools and Applications*, Vol. 12, 2000, pp. 145-188.
18. R. Willrich, P. Saqui-Sannes, P. Sénac, and M. Diaz, "Multimedia authoring with hierarchical timed stream Petri nets and Java," *Journal of Multimedia Tools and Applications*, Vol. 16, 2002, pp. 7-27.
19. GRiNS, http://www.oratrix.com/GRiNS/index.html.
20. T. T. Tien and C. Roisin, "A multimedia model based on structured media and sub-elements for complex multimedia authoring and presentation," *International Journal of Software Engineering and Knowledge Engineering*, Vol. 12, 2002, pp. 473-500.
21. LimSee2, http://wam.inrialpes.fr/software/limsee2/.
22. C. Roisin, V. Kober, V. Quint, P. Genevès, and P. Navarro, "Editing SMIL with timelines," in *Proceedings of the Synchronized Multimedia Integration Language European Conference*, 2003, http://wam.inrialpes.fr/publications/2003/smileurope_editing-timelines/EditingSMIL.html.
23. R. Deltour, N. Layaïda, and D. Weck, "LimSee2: a cross-platform SMIL authoring tool," *ERCIM News* (the quarterly magazine of the European Research Consortium for Informatics and Mathematics), 2005.
24. M. Y. Sung and D. Y. Lee, "A collaborative authoring system for multimedia presentation," in *Proceedings of IEEE International Conference on Communications*, 2004, pp. 1396-1400.
25. P. N. M. Sampaio, C. A. S. Santos, and J. P. Courtias, "About the semantic verification of SMIL documents," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2000, pp. 1675-1678.
26. J. P. Courtiat, C. A. S. Santos, C. Lohr, and B. Outtaj, "Experience with RT-LOTOS, a temporal extension of the LOTOS formal description technique," *Computer Communications*, Vol. 23, 2000, pp. 1104-1123.
27. A. Y. Chang, "Design of an intelligent distributed multimedia presentation system using temporal algebra and SMIL," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2004, pp. 2211-2214.
28. S. M. Chung and A. L. Pereira, "Timed Petri net representation of SMIL," *IEEE Multimedia*, Vol. 12, 2005, pp. 64-72.
29. C. C. Yang, C. K. Chu, and Y. C. Wang, "Dividable dynamic timeline-based authoring for SMIL2.0 presentations," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2004, pp. 1243-1246.
30. C. C. Yang, Y. C. Wang, and C. K. Chu, "Reuse of SMIL2.0 scripts in dividable dynamic timeline-based authoring," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Vol. 2, 2004, pp. 1235-1238.



Chun-Chuan Yang (楊峻權) received his B.S. degree in Computer and Information Science from National Chiao Tung University, Taiwan, in 1990 and Ph.D. degree in Computer Science from National Taiwan University in 1996. He joined the Department of Computer Science and Information Engineering, National Chi Nan University, Puli, Taiwan, as an Assistant Professor in 1998. Since Feb. 2008, he has been a Full Professor. His research area of interests includes multimedia network protocols, multimedia synchronization control, and multimedia applications.

Chen-Kuei Chu (朱振魁) received his B.S and master degree in computer science and information engineering from National Chi Nan University, Taiwan, in 2002 and 2004. His research topic is multimedia editing and authoring in SMIL2.0.

Yung-Chi Wang (王永吉) received his B.S and master degree in Computer Science and Information Engineering from National Chi Nan University, Taiwan, in 2002 and 2004. He is currently a Ph.D. student in the Institute of Networking and Multimedia, National Taiwan University. His current research topic is transmission improvement of multimedia traffic over Internet.