Modeling of the Non-Deterministic Synchronization Behaviors in SMIL2.0 Documents

Chun-Chuan Yang, Chih-Wen Tien, and Yung-Chi Wang
Multimedia and Communications Laboratory
Department of Computer Science and Information Engineering
National Chi Nan University, Taiwan, R.O.C.
ccyang@csie.ncnu.edu.tw

ABSTRACT

A novel model namely Extended Real-Time Synchronization Model (E-RTSM) for modeling SMIL2.0 synchronization behaviors is proposed in this paper. E-RTSM deals with schedule-based synchronization as well as event-based synchronization in SMIL2.0. Converting of the temporal relationship of a SMIL2.0 document to E-RTSM is presented. Moreover, design of the E-RTSM-based data-retrieving engine for SMIL2.0 presentations is also proposed in the paper. The data-retrieving engine estimates the worst-case playback time of each object at the parsing stage and applying an error compensation mechanism at run-time to adjust the estimated playback time as well as the schedule of the fetching request for data retrieval.

1. INTRODUCTION

Synchronized Multimedia Integration Language (SMIL) [1-4] was developed by *WWW Consortium* to address the lack of HTML for multimedia over WWW. With the introduction of SMIL, Web multimedia creators have a new tool for building time-based multimedia presentations that combine audio, video, images, animations, text, etc. There are two versions of SMIL specification that have been released. The first version of SMIL (SMIL1.0) [1] is primarily a scheduling model, but with some flexibility to support continuous media with unknown duration.

The second version of SMIL (SMIL2.0) [2-4] enhances SMIL1.0 by providing a strong support for user interaction with a declarative event-based timing. Event-based timing in SMIL2.0 presents the non-deterministic synchronization behavior such that the player cannot get the accurate playback time (and duration) for a media object as well as the total length of a presentation before run-time.

In our previous work [5, 6], we have developed modeling and converting techniques for parsing SMIL1.0 documents. *Real-Time Synchronization Model (RTSM)* [7] was used for modeling the temporal relationship in SMIL1.0 documents. Based on RTSM, an efficient data-retrieving engine was proposed. However, the proposed techniques in our previous work cannot be applied to SMIL2.0 directly. Extensions of RTSM as well as the modification of the converting algorithm are necessary to cope with non-deterministic features in SMIL2.0. In this paper, we propose *Extended RTSM (E-RTSM)* and the converting algorithm to present the temporal relationship in a SMIL2.0 document. Moreover, we apply E-RTSM in data retrieval and propose an E-RTSM-based data-retrieving engine.

It is worth mentioning that not much of the research work in the literature is concerning with modeling of non-deterministic temporal behaviors in multimedia presentations [8, 9], and it seems that none of the existing modeling techniques focuses on SMIL2.0 presentations.

The rest of the paper is structured as follows. First of all, we make a brief survey of original RTSM before *Extended RTSM* is presented in section 2. The converting algorithm from SMIL2.0 to E-RTSM is presented in section 3. The data-retrieving engine designed for SMIL2.0 presentations is presented in section 4. Finally, section 5 concludes this paper.

2. EXTENDED RTSM

2.1. Brief survey of RTSM for SMIL1.0

Real-Time Synchronization Model (RTSM) was proposed to address the lack of Petri-net based models 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 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. For example, Figure 2 shows the RTSM model for the sample SMIL1.0 code snippet in Figure 1.

2.2. Extended RTSM

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. (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, three mechanisms are added in E-RTSM.

```
<seq>
    <par endsync=A1>
        <audio id=A1>
        <text id=X1>
    </par>
    <img dur="10s" id=I1>
    <par endsync=A2>
        <audio id=A2>
        <text id=X2>
    </par>
    <img dur="10s" id=I2>
    <par endsync=A3>
        <audio id=A3>
        <text id=X3>
    </par>
</seq>
...
```

Figure 1. A SMIL1.0 code snippet

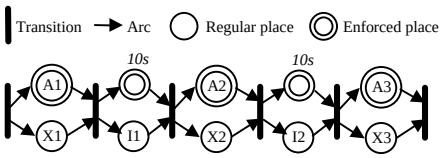


Figure 2. RTSM for the sample in Figure 1

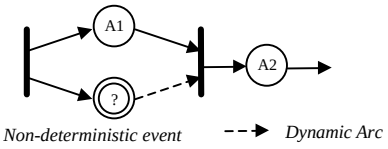


Figure 3. Using Dynamic Arc in E-RTSM

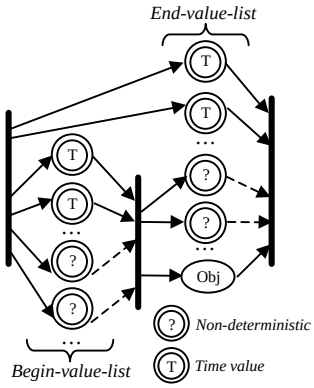


Figure 4. Converting *Begin-value-list* and *End-value-list*

(1) Allowing a place in E-RTSM to be mapped to a non-deterministic event. (2) *Dynamic Arc* is defined for the places of non-deterministic events. (3) *Run-time controllers* for complicated synchronization features are defined.

Dynamic Arc (denoted by dashed arrow in E-RTSM) is used to replace the normal arc following a non-deterministic event place. A dynamic arc is changed to a normal arc when the event occurs at run-time. For example, in Figure 3, the playback time of object A2 depends on A1 and the non-deterministic event. Two run-time cases may happen: (1) A2 is played when the playback of A1 is finished as long as the event does not occur during the playback of A1. In this case, the event does not have any effect on the firing time of the starting transition of A2. (2) If the event occurs while A1 is playing, the dynamic arc is changed to a normal one and that fires the following transition. Thus, A1 is stopped and A2 is played right after the event has occurred.

Run-time controllers are used to model complicated timing features in SMIL2.0 that are difficult to be represented by combination of other basic elements (arc, transition, place). By using run-time controllers in E-RTSM, handling of these complicated timing features is delayed until run-time, instead of the modeling (parsing) phase. Usage of the run-time controllers for SMIL2.0 elements is presented in the next section.

3. CONVERTING SMIL TO E-RTSM

In this section, we focus on converting major differences between SMIL2.0 and SMIL1.0 mentioned in section 2. The rest of the conversion for SMIL2.0 is similar to the conversion of SMIL1.0 in our previous work [5].

3.1. Converting Begin and End Attributes

We can classify the values of <*begin*> and <*end*> attributes to two types of event values: *Time value event* (event with a known occurring time, e.g., *Clock-value*) and *Non-deterministic event* (*Mouse-Click*, *Key-Pressed*, etc). A time value event is converted to an enforced place with duration specified by the event as in SMIL1.0 conversion. On the other hand, a non-deterministic event is converted to an enforced place that maps to that event. Thus, the following arc of a non-deterministic event place should be a dynamic arc.

E-RTSM for an element (media element or time container) with multiple <*begin*> values and <*end*> values is illustrated in Figure 4. Note that in the figure, the starting transition of non-deterministic event values in *End-value-list* is different from that of time value events. The reason is: SMIL2.0 specifies that a non-deterministic event in *End-value-list* does not have any effect on an element until the element has been activated.

3.2. Using the run-time controller

Currently, three run-time controllers (Figure 5) are defined in E-RTSM: *Restart controller*, *Min controller*, and *Repeat controller*. SMIL2.0 allows an element to be restarted multiple times during the element's active duration. The behavior is controlled by the <*restart*> attribute. Restart controller is used when the value of the <*restart*> attribute of an element equals "*always*" or "*whenNotActive*".

SMIL2.0 also allows the author to control the lower and upper bound of the element active duration by using the <*min/max*> attributes. Min controller is used when the <*min*> attribute is presented for an element. The effect of the <*max*> attribute is similar to that of a time value event in *End-value-list* as shown in Figure 5. Repeat controller is used when either the <*repeatCount*> attribute or the <*repeatDur*> attribute is presented for an element.

3.3. Example

The sample SMIL2.0 code snippet in Figure 6 is used to illustrate the converting process. Note that the SMIL2.0 sample is similar to the sample in Figure 1 except some non-deterministic events are presented in the SMIL2.0 sample. The final E-RTSM for the SMIL2.0 sample is displayed in Figure 7.

4. DESIGN OF THE DATA-RETRIEVING ENGINE

E-RTSM can be applied in designing the data-retrieving engine for SMIL2.0 documents. The data-retrieving engine is responsible for retrieving proper media data for the playback of a presentation. The concept of *just-in-time* data retrieving is proposed in our previous work for SMIL1.0 presentations [5]. Just-in-time data retrieving expects the retrieval process for an object to be finished right before the playback time for the object so that the player is able to continue the presentation smoothly. Thus, the

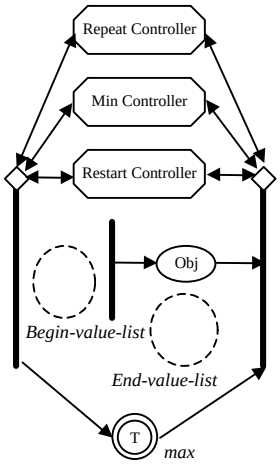


Figure 5. Run-time controllers in E-RTSM

```
<seq>
    <par endsync=A1>
        <audio id=A1>
        <text begin="Btn1.Click" id=X1>
    </par>
    <img end="Btn2.Click dur="10s" id=I1>
    <par endsync=A2>
        <audio id=A2>
        <text begin="Btn1.Click" id=X2>
    </par>
    <img end="Btn2.Click dur="10s" id=I2>
    <par endsync=A3>
        <audio id=A3>
        <text begin="Btn1.Click" id=X3>
    </par>
</seq>
...
```

Figure 6. A SMIL2.0 code snippet

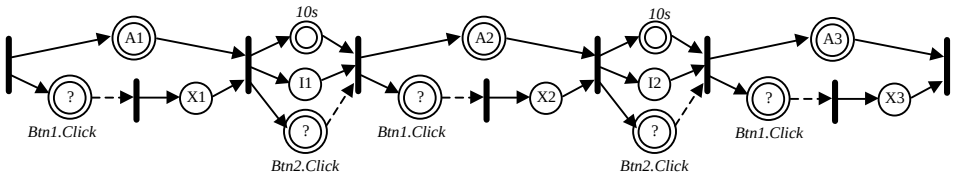


Figure 7. Final E-RTSM for the sample SMIL2.0 code snippet

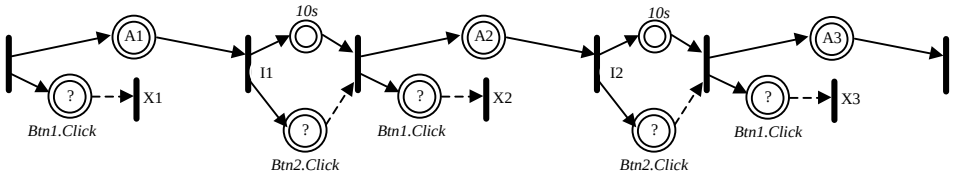


Figure 9. Reduced E-RTSM for the sample SMIL2.0 code snippet

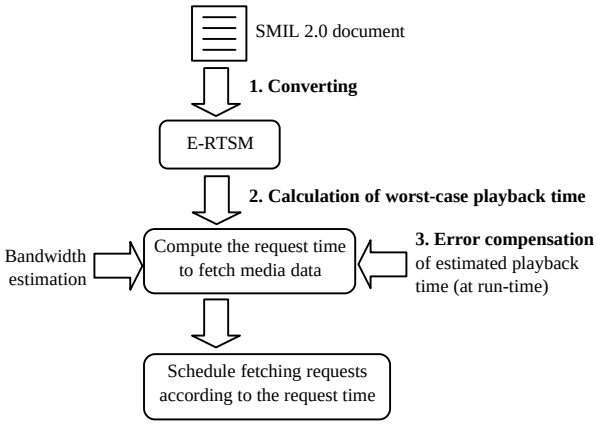


Figure 8. Data retrieving for SMIL2.0 documents

data-retrieving engine needs to compute the playback time for each object. The request time for an object is then calculated according to its playback time and bandwidth estimation.

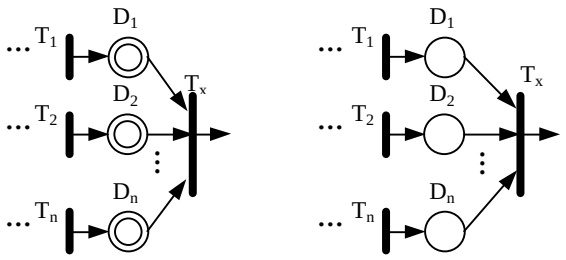
For SMIL2.0 presentations, the accurate playback time for an object cannot be obtained before run-time because of the non-deterministic events. Instead, the data-retrieving engine calculates the worst-case (earliest) playback time for each object at the parsing stage, and at run-time applying an error compensation mechanism for adjusting the estimated playback time as well as the request time. Overview of the proposed data retrieving process is illustrated in Figure 8.

4.1. Calculation of the worst-case playback time

At the parsing stage, the document is first converted to E-RTSM. The data-retrieving engine reduces E-RTSM by removing the regular places that have no effect on the firing time of a transition. As an example, Figure 9 is the reduced E-RTSM for Figure 7. The data-retrieving engine calculates the worst-case playback time for each object by assigning the duration of all non-deterministic events to zero and traversing the reduced E-RTSM. Dynamic arcs in the reduced E-RTSM are treated as normal arcs in the worst-case calculation of the playback time.

The playback time for an object is the firing time of its starting transition. There are only two possibilities for a transition in the reduced E-RTSM: (1) places feeding to the transition are all enforced places, or (2) places feeding 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*" + "*the duration of the following place*", which is illustrated in Figure 10-(a). The firing time of the transition for case (2) is instead the maximum value of its predecessors as illustrated in Figure 10-(b). The duration of each place depends on the type of the media object. For static media objects such as <*img*> and <*text*>, the duration of the place is zero. For continuous media object like <*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. As mentioned in the last paragraph, the duration of places that map to a non-deterministic event is set zero in the worst-case calculation.

During the traversal process of calculating the firing time of each transition, we also need to deal with the run-time controllers to get more accurate values for the worst-case playback time. The Restart controller deals with events that could restart an element during the active duration of the element, so the worst case would be no restart at all. Thus, the restart controller is ignored in the



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

Figure 10. Determine the firing time for transition T_x

worst-case calculation. The Min controller specifies a lower bound of the duration between the beginning transition and the ending transition of an element, so the firing time of the ending transition should be updated if the estimated duration is smaller than the <*min*> value. The Repeat controller deals with the <*repeatDur*> attribute as well as the <*repeatCount*> attribute. The <*repeatDur*> attribute sets the duration of repeating an element, so the firing time of the ending transition should be the end of the repeat duration. The <*repeatCount*> attribute specifies times of repeating for an element, thus the final firing time of the ending transition is extended as many times as specified by the <*repeatCount*> value. Update of the firing time of a transition with the run-time controllers is illustrated in Figure 11.

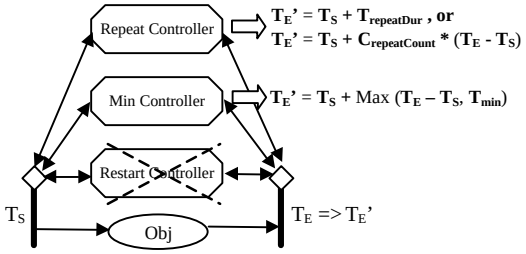
For example, assuming the intrinsic durations for A1, A2, and A3 are all 10s, the worst-case playback time for each object in Figure 9 is: A1= 0s, X1=0s, I1=10s, A2=10s, X2=10s, I2=20s, A3=20s, X3=20s. (Note that these values are relative to the starting time of the presentation)

4.2. Error compensation at run-time

Apparently, the actual playback time for an object at run-time is no earlier than the worst-case estimation. The difference (error of estimation) between the actual playback time and the worst-case estimation can be used to adjust the estimated playback time of the following objects that are not played yet. For example, if the event *Btn2.Click* in Figure 9 has occurred when I1 has been played 8s, we can then update the estimated playback time of the following objects: A2=18s, X2=18s, I2=28s, A3=28s, X3=28s. New estimation of the playback time is then used to re-calculate the new request time. The new request time for an object may not change the schedule of the fetching request since the request probably had already issued to retrieve the data. But the error compensation mechanism does make the estimated playback time of later objects more close to reality and it also makes data retrieval more intelligent.

5. CONCLUSION AND FUTURE WORK

In this paper, modeling of the non-deterministic synchronization behaviors in SMIL2.0 presentations has been presented. The proposed model namely Extended Real-Time Synchronization Model (E-RTSM) is the extension of our previous work for SMIL1.0 modeling. Based on E-RTSM, we propose a



T_E is the firing time w/o considering run-time controllers
T_E' is the updated firing time
$T_{repeatDur}$ is the value of <*repeatDur*> attribute
$C_{repeatCount}$ is the value of <*repeatCount*> attribute
T_{min} is the value of <*min*> attribute

Figure 11. Impact of the run-time controllers on the worst-case firing time

data-retrieving engine for SMIL2.0 presentations. The experimental platform previously developed for SMIL1.0 is currently being extended for supporting SMIL2.0. The future work of this paper includes modeling and handling of other complicated timing features in SMIL2.0 such as the <*excl*> element, the <*priorityClass*> element, etc.

REFERENCES

[1] *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*, W3C Recommendation, June 1998, http://www.w3c.org/TR/REC-smil.

[2] *Synchronized Multimedia Integration Language (SMIL) 2.0 Specification*, W3C Recommendation, 2001, http://www.w3.org/TR/smil20

[3] D.C.A. Bulterman, "SMIL 2.0 part 1: overview, concepts, and structure," IEEE Multimedia, Volume: 8 Issue: 4, Oct.-Dec. 2001, Page(s): 82 –88

[4] D.C.A. Bulterman, "SMIL 2.0. 2. Examples and comparisons," IEEE Multimedia, Volume: 9 Issue: 1, Jan.-March 2002, Page(s): 74 –84

[5] C. C. Yang, "Design of the Data-Retrieving Engine for Distributed Multimedia Presentations," Proceedings, IEEE International Conference on Communications, 2001 (ICC2001), pp. 3237-3243.

[6] C. C. Yang, "User-Interaction Supported Data- Retrieving Engine for Distributed Multimedia Presentations," Proceedings, IEEE International Conference on Communications, 2001 (ICC2001), pp. 3244-3250.

[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, No. 1, pp. 212-225, Jan. 1996.

[8] P. Diaz, I. Aedo, and F. Panetsos, "Modeling the Dynamic Behavior of Hypermedia Applications," IEEE trans. on Software Engineering, vol. 27, no. 6, June 2001, pp. 550-572.

[9] J. B. D. Joshi, Z. K. Li, H. Fahmi, B. Shafiq, and A. Ghafoor, "A Model for Secure Multimedia Document Database System in a Distributed Environment," IEEE trans. on Multimedia, vol. 4, no. 2, June 2002, pp. 215-234.