A SMIL-based Multimedia Presentation Authoring System and Some Remarks on Future Extension of SMIL

Mee Young Sung
Department of Computer Science & Engineering
University of Incheon
82 32 770 8496
mysung@incheon.ac.kr

Seung Jin Rho
Department of Computer Science & Engineering
University of Incheon
82 32 770 8498
sjrho@incheon.ac.kr

Jin Hee Jang
Department of Computer Science & Engineering
University of Incheon
82 32 770 8496
s0951111@incheon.ac.kr

ABSTRACT
We developed a multimedia presentation authoring system supporting a mechanism for conceptually representing the temporal relations of different media. Our authoring system represents media (such as images, videos, sounds, etc.) as objects and provides various editing functionalities for temporal compositions and spatial compositions. It is based on the SMIL (Synchronized Multimedia Integration Language). The system contains a temporal relation editor, a timeline editor, a layout editor, an attribute editor, a tag editor, a text editor, and a SMIL Object Manager. Among the many editors that make up our system, the temporal relation editor provides users with an intuitive mechanism for representing the conceptual flow of a presentation by simple and direct graphical manipulations. The SMIL Object Manager is responsible for the dynamic modification of each editor. The conceptual temporal relation editor and other editors of our system exchange their information in real-time and automatically generate SMIL codes through the SMIL Object Manager. Together they form an easy and efficient multimedia authoring environment.

In this paper, we present the design and implementation of a SMIL-based multimedia authoring system and we also propose some ideas to extend the current SMIL specification concerning the reusability of SMIL code.

Keywords
Multimedia authoring systems, SMIL (Synchronized Multimedia Integration Language), Temporal relation representation, Synchronization

1. INTRODUCTION
A multimedia authoring system must provide an environment where the temporal relations and spatial relations can be edited simultaneously. An interactive multimedia authoring system also needs to support user interactions. Some media (such as video, sound, and animation) require users to specify temporal characteristics, and other media (such as video, animation, images, and text) require users to specify the spatial relationship between objects. The key to authoring a presentation lies in the composition of temporal relationships between objects. Conceptually, the temporal relationships between two media can be classified into one of seven possibilities. They are ‘before’, ‘meets’, ‘overlap’, ‘during’, ‘starts’, ‘finishes’, and ‘equals’[1,2,3,4]. Every temporal relationship can be described using one of these seven relations. Spatial relationships can be described by specifying sub-regions within the total presentation region that correspond to each object.
The goal of this study is to develop an easy and efficient multimedia authoring environment where users can create a multimedia presentation in a simple and intuitive manner. Toward this goal, we provide users with the capability to edit temporal relationships between media objects at the conceptual level: for example, presenting object A before B, presenting object A during B, etc. We also want to allow users to create multimedia content without manually specifying the playing time (e.g., a specific start time and duration) for each media. Instead, our authoring system automatically calculates the playing time and then generates proper start times and durations for each object. In the traditional scaled timeline approach, users can directly view and control the structure of the content; however, the representation is fixed, and the operations are manual. Our goal was to develop a good tool for generating the presentation schedules conceptually without considering the details, and the system can automatically detail the properties of the media. Using our system, users can focus on the creative aspects of their design, and not worry about manual specification of timing details for each object.

We developed a multimedia authoring system based on the SMIL (Synchronized Multimedia Integration Language) [6, 7, 8, 9] 1.0 Recommendation [6] and SMIL 2.0 Recommendation [7]. SMIL is an XML-based markup language for integrated streaming media. SMIL allows users to easily describe the components of a multimedia presentation and their synchronization using simple tags, such as `<seq>` and `<par>`. Each component could be a unique web URL (Uniform Resource Locator). This means that a multimedia presentation can be made up of components from different websites, and the components can be shared and reused in many contents. Another advantage of SMIL is the simplicity of the description. The SMIL 2.0 Recommendation works to improve upon the previous version with the expanded description capability for animation and transition effect.

Many streaming media players support SMIL, e.g., Real Player G2 of Real Networks [10], QuickTime of Apple [11], GRIINS of Oratrix [12], etc. SMIL authoring tools such as GRIINS Editor [12], RealSlideShow [13], etc. are also available [14]. The existing SMIL authoring tools provide basic user interfaces such as the scaled timeline-based user interfaces (representing media objects as different bars arranged in multiple layers on the scaled timeline) or textual tag editing user interfaces for authoring. What distinguishes our system is that it provides a simple and intuitive editing mechanism for creating conceptual flows of a presentation, in addition to the basic timeline-based interface. In this paper, we present the design and implementation of our system, and we also propose some ideas to extend the standard SMIL to improve the user interaction and the reusability of SMIL content. We will examine our mechanism for representing conceptual temporal relationships in the following chapter. In chapter 3, we will describe two major component of our system: the editing system and the SMIL Object Manager. Remarks on our proposed extensions to SMIL will be addressed in chapter 4. Finally, the last chapter will provide conclusions.

2. REPRESENTATION OF CONCEPTUAL TEMPORAL RELATIONS

A main focus in authoring a multimedia presentation is the design of the temporal behaviors for the components that make up the presentation. Our system is designed to allow users to specify temporal behaviors of media objects at the conceptual level. This chapter describes our model for representing conceptual temporal relations.

2.1 Temporal Representation

Our system’s multimedia representation is based on Allen’s temporal intervals [1]. If we define $Iml$ to be the time interval of a multimedia object $ml$, then the following primitive relations can represent all temporal relations for multimedia authoring [4]:
Before\((m1,m2)\): \(m1\) ends before \(m2\) starts, and there is a delay element of nonzero length between the two.

Meets\((m1,m2)\): \(m2\) starts immediately after \(m1\) ends.

Overlap\((m1,m2)\): \(m1\) starts before \(m2\) starts, and \(m1\) ends before \(m2\) ends, or vice versa.

During\((m1,m2)\): \(m1\) starts before \(m2\) starts, and \(m1\) ends after \(m2\) ends, or vice versa.

Starts\((m1,m2)\): \(m1\) and \(m2\) start at the same time, but end at different times.

Finishes\((m1,m2)\): \(m1\) and \(m2\) end at the same time, but start at different times.

Equals\((m1,m2)\): \(m1\) and \(m2\) start at the same time and end at the same time.

Graphical representations of these seven temporal relations are summarized in Figure 1. The graphical representations shown in Figure 1 correspond exactly with the internal representation of each corresponding temporal relationship. Note that we represent the parallel relationships (such as overlaps, during, starts, and finishes) by adding dummy delay objects to the ‘equal’ relationship as shown in Figure 1.

![Graphical Representation of Temporal Relations](image-url)

**Figure 1. Graphical Representation of Temporal Relations**

### 2.2 Parallel Synchronization Block Representation

As shown in Figure 1, all five parallel relations can be generalized as the ‘equal’ relation by inserting some delay objects when they are needed. As a consequence, any parallel relation can be collapsed into a single object. We simplify a group of networked icons as a synchronization block object. This block object can be opened to show the details or collapsed to a single block icon as shown in Figure 2. This mechanism is proposed to simplify the representation of complicated parallel relationships within SMIL content.
2.3 Generation of SMIL Codes

Figure 3 illustrates an example of graphical representation of a multimedia presentation that is created as a user authors the presentation. After the authoring is finished, a DOM [15] structure (shown in Figure 4) can be generated from the graphical representation. Our system generates SMIL documents through the interaction between the graphical representation and the DOM structure.

Figure 3. An Example of Graphical Representation of a Multimedia Presentation
Figure 3 illustrates the step-by-step results when an author creates a presentation using our authoring system. Note that this example illustrates the full capabilities of our system including all seven temporal relationships.

The author first creates an “overview” of the presentation, as shown in Figure 3 (a). This presentation consists of a video and two parallel blocks. The first object of this presentation is video V1. The temporal relation between video V1 and the first parallel block P1 is ‘meets’. The relationship between the first block P1 and the second parallel block P2 is ‘before’. Every time an object is added in our system, the author is asked to define the temporal relation between the newly inserted object and one of the neighboring objects. The default relationship between two objects is ‘meets’. If the author would like to define the relation ‘before’, the author may choose the relation ‘before’ when the new object is inserted. In this case, a delay object would automatically be inserted by the system between the corresponding objects. Alternatively, he may accomplish the same thing by later inserting a delay object between the corresponding objects. Figure 3 (b) contains the timeline representation of Figure 3 (a).

After sketching the presentation overview, then the author further details the parallel blocks as shown in Figure 3 (c). The first parallel block is composed of an image I1, a video V2, and text data T1. The second parallel block is composed of an image I2, an audio clip A1, and a nested parallel block P3. Figure 3 (d) corresponds to the timeline representation of Figure 3 (c). The author specifies the relationship between image I1 and video V2 in the first parallel block as ‘overlaps’, and he sets the duration of image M2 to 20 seconds. Because the desired relation is ‘overlaps’, the author needs to specify the delay time for starting video V2 relative to the starting time of image I1. In this example, the delay time was set to 5 seconds. The author then specifies the relation between video V2 and text T1 as ‘finishes’. Because the duration of video M3 can be determined by its native length, the author only needs to specify the duration of text data T1 or the delayed starting time of text data T1 relative to the start time of video V2. In this example, the author chose to specify the duration of M4 as 20 seconds. In the second parallel block, image I2 is in ‘starts’ relationship with audio A1, audio A1 is in ‘meets’ relationship with the nested parallel block P3, and the parallel block P3 is in ‘finishes’ relationship with image I2 as shown in Figure 3 (c). Therefore, the relationship between image I2 and the sequence of audio A1 and the nested block P3 becomes ‘equals’. The author could also define the same relationship among these three objects by simply placing the media objects at the appropriate positions in the second block without explicitly specifying relationships, because the default temporal relationship of a par block is always ‘equals’.
The author then proceeds to specify in detail the nested parallel block. The results are illustrated in Figure 3 (e). The nested block is composed of audio A2, image I3, and video V3. The relationship between audio A2 and image I3 is ‘starts’, and the duration of image I3 is set to 20 seconds. In this case, the native duration of the audio A2 is 15 seconds. The relationship between image I3 and video V3 is ‘during’. In the ‘during’ relationship, the delay time and the duration (or the end time) of the late-starting object need to be manually set. In this example, the delay time of V3 and the duration of V3 are set to 3 seconds and 12 seconds respectively. This means that video V3 is the object with the latest start time. Figure 3 (f) corresponds to the timeline representation of Figure 3 (e).

Our authoring tool calculates implied delays and applies them to the temporal specification of the multimedia contents, and then it automatically generates the correct SMIL code. In our authoring system, the playing time of an object can be determined automatically from the native duration. Therefore, authors do not need to define the start time or the end time of media explicitly. Authors need only to define the type of temporal relation and the delay if necessary. (For example, when the temporal relationship is ‘overlaps’ or ‘during’, the delay time and the duration (or the end time) of the late-starting object need to be specified. When the temporal relationship is ‘before’, the delay between the end of the first object and the start of the second object must be specified.)

Note that an author is able to define a custom duration for a media object which is different than its native length. An author can also specify to play a certain part of a media object using the ‘clip-begin’ and ‘clip-end’ attributes. These attributes can be specified or modified in our attribute editor (presented in section 3.1). In addition, a media object or a parallel block can be repeated by specifying the ‘repeat’ attribute using the attribute editor.

Our system generates the following SMIL codes from the information of the document in Figure 3 and Figure 4. Note that in SMIL, we can specify attributes for each object, such as ‘duration’, ‘begin’, ‘end’, etc. The ‘duration’ attribute defines the total playing time. The ‘begin’ attribute defines the delay interval from the starting time of the objects in a parallel relationship. The ‘end’ attribute defines the ending time of an object relative to the starting time of the objects in a parallel relationship. If we know the temporal relationships among objects, it is sufficient to specify only one of the attributes ‘duration’, ‘begin’, and ‘end’, because the DOM can use known temporal information to automatically calculate values for the unspecified attributes.

```xml
<body>
  <seq>
    <video id="M1" region="video_region" src="intro.mpg" dur="20s"/>
    <par>
      <img id="M2" region="img_region" src="choice.gif" dur="20s"/>
      <video id="M3" region="video_region" src="move.mpg" begin="5s" dur="25s"/>
      <text id="M4" region="text_region" src="testword.txt" begin="10s" end="30s"/>
    </par>
    <par begin="5s">
      <img id="M5" region="img_region" src="wind.gif" dur="40s"/>
      <seq>
        <audio id="M6" src="wind.wav" dur="20s"/>
        <par>
          <audio id="M7" src="explain.wav" dur="15s"/>
          <img id="M8" region="img2_region" src="moon.gif" dur="20s"/>
          <video id="M9" region="video.region" begin="3s" dur="12s"/>
        </par>
      </seq>
    </par>
  </seq>
</body>
```
The SMIL code excerpt above shows only the <body> segment of the complete SMIL document. It specifies that video M1 will be played alone. After video M1 finishes, image M2 will be shown together with video M3 and text data M4 playing in parallel as specified with the <par> tag. Video M1 will be played for its native duration of 10 seconds. Then image M2 will be played for 20 seconds as the author custom defined. Video M3 will begin 5 seconds after image M2 begins as specified by the author. Because the relation between video M3 and text data M4 is ‘finishes’, and video M3’s native duration is 25 seconds, text data M4’s end time is automatically set to 30 seconds. (As explained before, the author had chosen to specify the duration of text data M4 as 20 seconds.) Text data M4’s start time is automatically calculated as 10 seconds and stated explicitly in the code. Then there will be a 5 second delay, after which image M5 will be presented together with audio M6. After audio M6 plays for 20 seconds, audio M7 and image M8 will be played simultaneously. 3 seconds after the start of these two objects, video M9 will be played for 12 seconds. Audio M7 will be played for 15 seconds and image M8 will persist for 5 additional seconds after audio M7 stops. Image M5 continues to be shown and disappears with image M8 at the end of the presentation.

3. EDITING SYSTEM AND SMIL OBJECT MANAGER

Our system represents images, videos, sounds, texts, text streams, and animations as media objects, and provides various editing functionalities for temporal compositions and spatial compositions. Our system is composed of the following two main sections:

- Editing System
- SMIL Object Manager

We summarize the global system architecture of our authoring system in Figure 6.
3.1 Editing System

The editing system consists of a temporal relation editor, a timeline editor, a layout editor, a tag editor, an attribute editor, and a text editor. The conceptual temporal relation editor allows users to edit the temporal relations between media by simple and direct graphical manipulations. Moreover, it provides users a mechanism for representing the conceptual flow of the presentation intuitively. The timeline based temporal relation editor permits us to define the detailed temporal relations in an absolute timeline, such as delay time, playing time (duration), etc. The layout editor provides the facility to edit the spatial relations in a drag & drop manner. These editors exchange their information through the SOM (SMIL Object Manager) and together form an easy and efficient editing environment.

Figure 7 shows the user interfaces of our editing system. The role of each editor is as follows:

- **Temporal Relation Editor**: an editor that allows us to represent the conceptual flow of the presentation.
- **Timeline Editor**: an editor based on a timeline where the start time and the duration of the presentation of each media are represented. Note that the Timeline Editor is not visible in the screenshot in Figure 7. The author can choose to view either the Timeline Editor or the graphical Temporal Relation Editor using the tabs at the bottom of the Temporal Relation Editor panel shown in the figure.
- **Layout Editor**: an editor for defining the spatial information for media that will be presented in a space.
- **Tag Editor**: an editor that can be used to insert and to delete the SMIL tags.
- **Attribute Editor**: an editor where the attributes of SMIL objects can be specified in detail.
- **Text Editor**: an editor where we can directly edit SMIL files in text form.

![Figure 7. Editing Systems](image-url)
3.2 SOM (SMIL Object Manager)

Figure 8 presents the internal structure of SOM, an essential part of our system. SOM is responsible for maintaining the consistency of the presentation information and distributing all modification information from each editor to other editors. SOM consists of the following three parties:

- **SMIL Object Controller (SOC):** exchanges information between each editor and dynamically updates the status of all editors to ensure consistency. SOC also creates and maintains the internal structure, automatically inserting delay objects when necessary. SOC also communicates with the DOM generator to read/store presentation information.

- **SMIL Parser:** examines and parses SMIL files. SMIL parser also verifies that the syntax of SMIL files that are automatically generated by the SOC and the DOM Generator. It also displays the SMIL code on the text editor.

- **DOM (Document Object Model) generator:** works with information provided by the SOC to generate a standardized DOM document corresponding to the SMIL code.

4. SOME REMARKS ON FUTURE EXTENSION OF SMIL

Our experience in the development of a SMIL-based multimedia authoring system leads us to propose two ideas to extend and improve the existing SMIL. One is to include a more general mechanism for diverse user interactions in SMIL. The other is a method to reuse previously created SMIL code (in whole or in part).

Note that XML provides a mechanism to refer to a specific block of documents using the construct Xpointer(Xpath). Consider the example of referencing a child of the chapter node in a document structure. Assuming the child has its ‘title’ attribute set to ‘Introduction’, a reference could be made by specifying “xpointer(/child::chapter[attribute::title=’Introduction’])”. However, even though SMIL is based on XML, no mechanism is currently available for referencing and reusing already prepared blocks of SMIL code.
when authoring SMIL content. The current SMIL specification allows for references to other SMIL document (.smi file) or another web document (.htm or .html file) via the <a> tag or the <anchor> tag. We propose that SMIL should include a mechanism to import a block of SMIL code from the same SMIL file or from some other SMIL files.

It would be possible to include a referencing mechanism by incorporating XML’s XPointer(Xpath) construct into SMIL. However, we propose a simpler mechanism which makes use of the ‘id’ attribute within the existing <seq> and <par> tags to facilitate access and reuse of SMIL blocks. The following SMIL code shows an example for reusing existing SMIL codes via the second approach. This example illustrates that a block whose id is ‘show’ in a presentation named ‘PresentationA’ is to be reused in another presentation named ‘PresentationB’. This reference could be coded as:

```
<par ref="PresentationA.smil//id(show)" />
```

PresentationB becomes a new presentation composed of a predefined ‘show’ block and a text data object. If this mechanism of reusing SMIL blocks is incorporated by SMIL systems, then new SMIL content can be generated in a very efficient way.

```
//PresentationA.smil
<smil>
 ...... 
 <body>
 <par id="show">
   <audio src="show.wav"/>
   <img region="I_region" src="showimage.gif" dur="20s"/>
 </par>
 </body>
</smil>

//PresentationB.smil
<smil>
 ...... 
 <body>
 <par>
   <par ref="PresentationA.smil//id(show)" />
   <text region="text_region" src="showtext.txt" dur="20s"/>
 </par>
 </body>
</smil>
```

5. CONCLUSION

We developed a SMIL-based authoring system which allows users to edit the temporal relations among media conceptually by simple and intuitive graphical manipulations. The system editors exchange information through the SOM (SMIL Object Manager) and together form an easy and efficient editing environment. One advantage of our system is that multimedia authors need not specify all of the painstaking details about start times and duration information when creating a presentation. This frees multimedia authors to focus instead on the creative aspects of the presentation. Another advantage is the use of collapsible synchronization blocks to efficiently represent and specify simultaneous presentation of parallel media. The collapsible synchronization blocks used by our system also provide a means for portable and reusable SMIL code blocks. Concerning SMIL extension, we proposed a mechanism to include a method to define and to reuse existing blocks of SMIL code.
Note that in addition to their use in SMIL authoring systems, our system editor modules can be used separately in various kinds of multimedia presentation authoring systems, such as XMT (eXtensible MPEG-4 Textual format) [16]-based authoring tool [17]. Because our system makes use of the standard DOM structure, its components can easily be applied by any multimedia system which also uses the DOM structure for its internal document representation.

6. ACKNOWLEDGEMENTS

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Multimedia Research Center at the University of Incheon.

7. REFERENCES