MULTIMEDIA WEB FORMS

Kari Pihkala, Mikko Honkala, and Petri Vuorimaa

Telecommunications Software and Multimedia Laboratory,
Helsinki University of Technology,
P.O. Box 5400, FI-02015 HUT, Finland.
Email: kari.pihkala@hut.fi, mikko.honkala@hut.fi, and petri.vuorimaa@hut.fi

ABSTRACT

Synchronized Multimedia Integration Language (SMIL) 2.0 supports user interaction with its time-based hyperlinking and event model. However, often more advanced interaction is desired, such as information exchange with a server. Traditionally, Web forms have offered means to send data to a server. XForms is an effort by W3C to create a host language independent Web form standard. It can also be used in conjunction with SMIL. However, the advanced XForms repeat feature requires a flow layout, which is not available in SMIL. This paper describes the integration of SMIL and XForms, the issues in the integration and possible solutions.

1. INTRODUCTION

One of the corner stones in the World Wide Web (WWW) is the ability to transfer data from a client to a server on Web forms. This has enabled creation of various Web applications from e-mail clients to complex maintenance systems. As the WWW is moving towards multimedia content, continued support for this kind of interaction is desirable.

Synchronized Multimedia Integration Language 2.0 (SMIL) is an effort to advance the use of multimedia in the WWW [1]. In SMIL, user interaction can happen either by hyperlinks or by declarative event timing model. These offer a way to traverse within the presentation, or to other presentations. However, form input mechanisms, such as text entry fields, and capability to send data to a server are missing in the current specification.

XForms is an emerging new standard to describe advanced Web forms. The standard relies on the separation of user interface, data model, and instance data. The user interface includes common text entry fields, buttons, and checkboxes. It can also create more complex representations, such as repeated form fields and multi-page forms. The data model takes care of validation and calculations in the form, while instance data describes data in XML, to be sent and parsed on a server. XForms is completely declarative, avoiding use of a scripting language. One of the goals of XForms is that it can be embedded in any host language, including SMIL. However, some of the advanced features of XForms are not possible with the standard SMIL 2.0. Repeating data fields require a flow layout, where cloned data fields are added one after another to the document. By default, SMIL does not support flow layout [2]. This can be fixed by extending the SMIL layout module.

Cascading Style Sheets (CSS) is a style description language [3]. Originally, it was designed to style HTML, but is now also used to style other languages, such as SVG and XForms. Also, the XHTML+SMIL profile [4] uses CSS as the layout language. CSS can also be used to describe the layout of a SMIL document, as discussed by Hoschka and Lilley [5]. The main reason not using CSS by default in SMIL is the small number of CSS properties required by SMIL, and the absence of some multimedia specific properties in CSS. However, CSS can be used in SMIL to overcome the limitation of the missing flow layout. Also, this makes it easier to integrate languages, which already use CSS, into SMIL.

The implementation of SMIL and XForms integration described in this paper is based on the X-Smiles browser [6]. X-Smiles is an XML-browser written in Java. It has an implementation of SMIL [7], and XForms [8]. The browser is also capable of integrating XML languages [9]. As an open source project, the browser can be downloaded at http://www.x-smiles.org.

The key results of this paper are: 1) the integration of standard SMIL 2.0 and XForms 2) the integration of SMIL 2.0, CSS and XForms, to handle repeated data fields, 3) implementation experience of SMIL and XForms integration.

This paper is organized as follows. The next section describes the related work about user input forms in multimedia documents. Section 3 describes the integration of SMIL with XForms and CSS. Section 4 points out the major implementation issues. Section 5 demonstrates the use of multimedia Web forms. Finally, conclusions with future directions are given.

2. RELATED WORK

There are various multimedia standards describing the use of forms in multimedia documents. This section describes their effort.

MHEG-5 specifies an encoding format for multimedia applications. The standard is aimed at a wide area of applications, e.g., Video On Demand and Interactive Television systems. It includes user interface gadgets for forms, e.g., text entry fields, switch buttons, sliders and push buttons [10]. MHEG-6 extends MHEG-5 with an interface between MHEG-5 engine and a Java Virtual Machine, providing access to external resources, such as servers.

XHTML+SMIL Profile [4] is based on the XHTML 1.0 language. All features of the XHTML language are supported, including XHTML form controls, such as text input fields, combo boxes, and buttons. The XHTML+SMIL Profile also supports several of the SMIL modules, including timing, animations, and transition effects modules. The combination of XHTML and SMIL modules makes it possible to apply timing, animations and transitions on a typical Web page including Web forms. However, the profile supports legacy Web forms, not the more advanced XForms specification.

Also, several proprietary formats exist to achieve forms embedded in multimedia applications. Data can be sent to the server in the form of a HTTP request, by appending a Get URL action with the data. For instance, Macromedia Flash applications can be embedded in Web pages. Flash does not have build-in form elements, but it is possible to create such with an authoring tool. Also, data can be sent to the server, for instance, in the HTTP request.

3. MULTIMEDIA WEB FORMS

This section describes the integration of SMIL and XForms. First, the standard SMIL 2.0 language and XForms integration is described with possible issues. Then, an integration extended with a CSS layout model is described to use the advanced repeat feature of XForms.

3.1. Integration of standard SMIL and XForms

Since both, SMIL and XForms, are based on the XML standard, it is possible to compose them in the same document. The languages are separated with a namespace [11], with SMIL as *a host language*, and XForms embedded as *a parasite language*.

The most natural place for XForms instance model elements is the head section of the SMIL document, while the XForms controls are placed in the body section. An XForms control will be treated as any other discrete SMIL media element. However, instead of referencing to an external media source, the XForms control will internally create its visual representation, such as a checkbox or a button. In addition, all SMIL media element attributes, including timing attributes, can be placed on the XForms control, allowing defining the spatial position and temporal behavior of the XForms control.

These languages mix well together. As mentioned, SMIL decides the positioning and timing of the XForms elements, while XForms provides the gadgets. Also, both can keep track of the state of each other. When the user interacts with the XForms controls, the *XForms Processor* dispatches *initialization*, *interaction*, *notification*, and *error condition* events to the DOM tree. These DOM events can be listened by SMIL using event timing, allowing changing the presentation according to changes in the form. Vice versa, *XForms Actions* can listen to SMIL events, such as SMIL element state changes and mouse clicks. XForms Actions are declarative XML event handlers that can be used to modify the instance data, dispatch events to the XForms Processor, and to provide feedback to the user. One slightly overlapping feature is the XForms' <switch> element. It is used to create multi-page forms by changing the visibility of its children. The same effect can be achieved with SMIL timing attributes.

XForms is mainly designed to be integrated with XHTML. Thus, it has been designed to take advantage of CSS styling to control the feel and look of its controls, such as the font, colors, and size of the controls. None of these user interface formatting features are available in pure SMIL and XForms integration without CSS. Of course, the styling properties could be added as attributes to the SMIL basic layout. Alternative is to add support for CSS styling in the XForms controls. This enables the use of most of the XForms user interface formatting features. In this case, CSS styling determines the look of the individual XForms control, without affecting the overall SMIL layout.

However, there is still one limitation in the integration mentioned above. XForms defines repeating elements. The repeat feature was designed to work with a host language, which has a flow type of layout, e.g., XHTML. The problem with a language without a flow layout, such as SMIL, is that the repeating elements are exact clones of each others. This means that they will have exactly the same attributes, i.e., the same region and timing attributes. If placed under a <seq> element, each of the repeated elements will show up at the same location one after another. Under a <par> element, the repeated elements will appear on top of each others, at the same time. To overcome this limitation, we introduce a flow layout to SMIL using a CSS layout model.

3.2. Repeat feature with CSS

XForms repeat feature assumes that the host language has a flow type of layout. The standard SMIL 2.0 Languge

```
 1: <layout type="text/css">
 2:   *           { display: inline; }
 3:   body        { background-color: white; color: black; }
 4:   .largeblock { background-color: gray; position: absolute; display:block;
 5:                  top: 140px; left: 20px; width: 580px; height: 380px; }
 6:   .groupblock { position:static; display:block; height:25px; }
 7: </layout>
...

10: <smil:par class="largeblock">
11:   <xfm:repeat nodeset="data/names">
12:      <xfm:group class="groupblock">
13:         <smil:img src="p.jpg" begin="1s"/>
14:         <xfm:input ref="name" class="nm"/>
15:         <xfm:range start="0" end="10"
16:               stepsize="1" ref="units"/>
17:      </xfm:group>
18:   </xfm:repeat>
19: </smil:par>
```

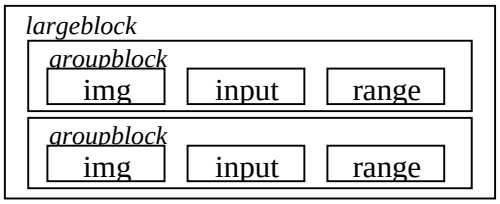

Figure 1: Repeating forms example on the left, the resulting layout on the right.

profile does not support it. By using an alternative layout model in SMIL, it is possible to construct a flow type of layout. A layout model based on CSS was selected, as CSS is also used by XForms.

The CSS layout model replaces the standard SMIL layout model. The type of the layout model is declared with the type attribute of the <layout> element. Instead of using <region> elements, the layout is described in CSS. Regions similar to SMIL can be created in CSS with absolute positioning (i.e., top, left, right, bottom, width, and height properties). Also, z-indexing and background-color are supported. However, some features present in SMIL are missing in CSS (e.g., scaling media to fit the region, audio volume control, and top-layouts). [5]

There are three ways to declare the actual CSS sheet in SMIL. A reference to an external CSS can be given with a processing instruction <?xml-stylesheet type=”text/css” href=”external.css”?>, or it can be written as plain text under a layout element. Third alternative is to add a style attribute to an element in the body section.

There are two alternatives to support the table representation intended in repeat. The first option is to implement the table properties of CSS. The second option is to support inline and block formatting. The latter option is discussed in this paper, as it is a more basic feature of CSS. A repeat element will clone its children, which will be added one after another into a parent element. Figure 1 depicts an example of a repeat element. Lines 1-7 contain the CSS sheet used in the example. By default, all elements behave as inline elements. In line 10, the time container defines the position and dimension of a block. In line 12, the XForms <xfm:group> element groups the underlying elements into rows. Finally, lines 13-16 have the inline elements, which are under the <xfm:group> element. In the example, the <xfm:repeat> element has repeated twice, resulting in the presentation shown in Figure 1.

In the standard SMIL 2.0, the layout and timing are completely separated. Introducing CSS will combine them. The order of the elements under the body section will define the timing order and the layout order. We have defined the layout order of the elements to be the same as the order in the document. Thus, even if the image, as in the example, starts later than the other controls, it will be positioned before them. This way, the timing and the layout of the document are kept consistent.

As both, SMIL and XForms, use CSS to define the layout and style of the presentation, it is possible to define one common CSS for both. Thus, it is possible to create reusable external stylesheets to describe the layout and style of the most commonly used presentations.

4. IMPLEMENTATION DETAILS

This section reveals some of the implementation details in order to integrate SMIL with XForms and CSS. X-Smiles XML-browser was used as the experimental platform.

4.1. XForms integration

When SMIL and XForms are mixed within a single document, the result is called *a hybrid XML document*, which is a document containing elements from different XML languages, usually distinguished by namespaces [11]. XForms was integrated into SMIL using a browser framework, which allows rendering hybrid

XML documents. The framework is a result of an earlier research [9]. The basic idea in the framework is that a single DOM tree is created, but each node within that DOM tree contains a specialized implementation for that node type.

A single XML language in the framework is implemented by a *Markup Language Functional Component* (MLFC). MLFC is responsible for providing the specialized node implementations and a rendering capability. There are two types of MLFCs; hosts and parasites. A host MLFC, such as SMIL MLFC, is determined by the root element of the document and is responsible for the main layout of the document. A parasite MLFC, such as XForms MLFC defines elements, which need a host language to live in. [9]

Usually there needs to be communication between parasite and host nodes in the DOM tree. The framework provides interfaces for this communication and thus enables direct communication between the nodes in the DOM tree. In the SMIL+XForms integration discussed in this paper, the XForms nodes send their visual component to the parent SMIL element, which decides the temporal and spatial placement of the component.

4.2. The CSS Layout Model

In order to be able to use a single CSS stylesheet across the different language elements in the document, a common CSS module was created. Its responsibilities include *Stylesheet parsing*, *sorting rules by specificality*, *selectors* and *inheritence* [3]. The CSS module goes thru the DOM and attaches style information for each element. It is then up to the MLFCs to use that information, while rendering.

The SMIL MLFC was extended with a support for a CSS layout model. Because the CSS module does not support *visual formatting*, the existing functionality in the standard SMIL layout module was reused. The following gives an overview of the process.

Each element in the body section will have a sub-region created for it. This is similar to the HierarchicalLayout Module in SMIL. However, the regions are nested according to the hierarchy of the body section instead of the head section. The CSS properties for an element are mapped to the region's attributes. This means that we are only supporting the CSS properties available in SMIL regions, extended with the display and position properties.

To create inline or block regions, the region element was extended with an ability to display the child element's regions in a flow, one after another. In case of absolute positioning, left and top properties determine the position of the region.

It is possible to insert and delete items in the repeat feature. In these cases, the actual element is inserted or deleted in the DOM tree. The implementation must cope with such a dynamic DOM tree. In previous research, we have shown how to extend SMIL with a scripting language [7]. It also required a dynamic DOM tree, and the functionality is reused here.

Currently, the CSS implementation is not dynamic, i.e., it is not possible to animate the CSS values. This means that, currently, the animation module is not supported with the CSS layout model. However, this is only a limitation of the implementation.

5. APPLICATIONS

This section demonstrates the use of multimedia forms. First, the use of XForms events to launch SMIL media is demonstrated. Then, a more comprehensive example animating XForms controls is described. Finally, the XForms' repeat functionality in SMIL is demonstrated.

5.1. Audio enhanced forms

This example demonstrates the use of audio to guide the user to fill in a Web form. This is useful in devices with small displays, such as mobile phones, as there is not much space for help texts. Also, an audible error message is given in case of mistakes.

This demonstration, shown in Figure 2, was created as a pure SMIL+XForms document. No scripting was used in the document. It makes use of the XForms events to launch the help messages as audio. For instance, when a user writes an invalid value to the field "Retype" the "xforms-invalid" – event is dispatched. Then a <smil:audio> element listens to that event and launches an audio message describing the error as shown in Figure 3. Similarly, focus changes in XForms dispatch the "DOMFocusIn" events, which again are used to launch audio.

Figure 2: Audio enhanced Web form.

```
<xforms:secret id="retype"
 ref="retypepasswd" region="rtp">
   <xforms:caption>Retype
   </xforms:caption>
</xforms:secret>
<audio src="screwed.wav"
 begin="retype.xforms-invalid"
 region="sound"/>
```

Figure 3: Snippet showing the use of event timing.

5.2. Phone Order

This example demonstrates the use of animation in Web forms, depicted in Figure 4. The standard SMIL 2.0 layout module defines the position for all media and form controls. To demonstrate interaction between SMIL and XForms, an advanced select one control is created with SMIL. Instead of displaying the standard XForms select one gadget, images of phones are shown. Clicking on the phone image will select it and show specific options for it. The options will not just appear and disappear, but rather they are animated using SMIL's animation feature. The phone selection will also change the instance data in XForms, using the <xforms:setValue> element, and data will be sent to the server, when the "Order" button is pressed.

Figure 4: Animated Web form.

5.3. Repeat in XForms

This example demonstrates the use of XForms repeat in SMIL. The form has several data rows with five answer columns, totaled in the last column. The CSS layout model was used to create the table format shown in Figure 5. Rows are created with a <par> element styled as a block element. The columns are inline elements added to the block element one by one. The document fragment of this repeat is very similar to Figure 1. However, due to implementation limitations, we have replaced the <xfm:group> element with a <smil:par> element, which does not have any affect in the end result.

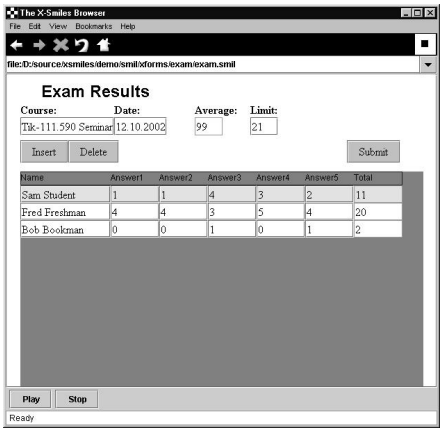

Figure 5: Repeat in SMIL.

6. CONCLUSIONS

The integration of SMIL and XForms was described in this paper. Generally, the languages mix well together. The XForms controls can be treated as media elements and the languages can communicate via DOM events. However, formatting and repeat feature in XForms are problematic.

XForms has been designed to take advantage of CSS properties to format its controls. The standard SMIL 2.0 layout does not support this. To overcome this, a style attribute and CSS formatting was added for XForms controls. Also, the SMIL layout cannot handle the XForms' repeat feature. To overcome this limitation, CSS layout model was used in SMIL. With the CSS layout, it is possible to display also the repeat feature.

The CSS layout has its drawbacks, as the layout and timing will be mixed in the body section. However, this is unavoidable. Keeping the layout order always the same as the document order will keep the layout and timing as separate as possible.

In the future, introduction of flow type of layout to SMIL would allow taking advantage of the full XForms specification.

6. ACKNOWLEDGEMENT

The authors Kari Pihkala and Mikko Honkala would like to thank Nokia Oyj Foundation for providing the scholarship and support during the research work.

Also, thanks go to Alessandro Cogliati for the implementation of the CSS Engine used in X-Smiles.

7. REFERENCES

[1] L. Rutledge, “SMIL 2.0: XML for Web multimedia,” IEEE Internet Computing, Vol. 5, iss. 5, pp. 78-84.

[2] M. Honkala and P. Vuorimaa, "Advanced UI features in XForms", a workshop in the 8th International Conference on Distributed Multimedia Systems, September 25 - 28, 2002.

[3] B. Bos, et al., “CSS2: Cascading Style Sheets, level 2,” W3C Recommendation, May 12, 1998.

[4] D. Newman, et al., "XHTML+SMIL Profile," W3C Note, January 31, 2002.

[5] P. Hoschka and C. Lilley, “Displaying SMIL Basic Layout with a CSS2 Rendering Engine,” W3C Note, July 20, 1998, http://www.w3.org/TR/NOTE-CSS-smil.html.

[6] P. Vuorimaa, et al., “A Java based XML browser for consumer devices,” the 17th ACM Symposium on Applied Computing, Madrid, Spain, March 10-13, 2002, pp. 1094-1099.

[7] K. Pihkala and P. Vuorimaa, "Design of a Dynamic SMIL Player," the IEEE International Conference on Multimedia and Expo, ICME’2002, Lausanne, Switzerland, August 26-29, 2002.

[8] M. Honkala and P. Vuorimaa, “XForms in X-Smiles,” WWW Journal, Vol. 4, no. 3, 2001, pp. 151-166.

[9] K. Pihkala, M. Honkala, and P. Vuorimaa "A Browser Framework for Hybrid XML Documents," Proc. of the 6th IASTED International Conference, Internet and Multimedia Systems, and Applications, (IMSA 2002), August 12-14, 2002, Kauai, Hawaii, USA.

[10] M. Echiffre, et al., “MHEG-5 – aims, concepts, and implementation issues,” IEEE Multimedia, Vol. 5, Iss. 1, Jan-March, 1998, pp. 84-91.

[11] T. Bray, et al., "Namespaces in XML," W3C Recommendation, January 14, 1999.