IMPROVING MEDIA FRAGMENT INTEGRATION IN EMERGING WEB FORMATS

LLOYD RUTLEDGE AND PATRICK SCHMITZ

CWI, P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands
email: Lloyd.Rutledge@cwi.nl, Patrick.Schmitz@cwi.nl

The media components integrated into multimedia presentations are typically entire files. At times the media component desired for integration, either as a navigation destination or as coordinate presentation, is a *part* of a file, or what we call a *fragment*. Basic media fragment integration has long been implemented in hypermedia systems, but not to the degree envisioned by hypermedia research. The current emergence of several XML-based formats is beginning to extend the possibilities for media fragment integration on a large scale. This paper presents a set of requirements for media fragment integration, describes how standards currently meet some of these requirements and proposes extensions to these standards for meeting remaining requirements.

1 Introduction

The computer system file, as the most basic unit of data storage and distribution, has long provided the initially conceived division between distinct media components in multimedia integration. Most systems most easily, or only, handle the integration of a media object, whether as the destination of a hyperlink navigation or as one component of a coordinated presentation, as a single, entire file. When integrated media objects must be entire files, then they can only be reused in multimedia presentations that can work with the whole file. Consequently, if a portion of an existing media file is to incorporated in a multimedia display, then a separate file must be created whose sole contents are that portion. This results in redundant media data. Preventing this motivates the ability of multimedia systems to integrate portions of media files.

We call a portion of media file that is integrated into a multimedia presentation a *fragment*. If multimedia presentations can handle fragments of media files as whole media objects in their own right, then one media file can have many fragments that are integrated separately into different multimedia presentations. This results in more efficient storage of media data. It also requires that a fragment can be located within a media file, extracted and processed as distinct from the rest of the file.

We call a portion of media file that is not a fragment within it that fragment's *context*. Often times a media file makes up a whole conceptual unit. A multimedia presentation may present a fragment of the file but also present its context as distinct from the fragment but also containing information important to the user for understanding the fragment's significance. The resulting requirement is that the context must not

only be able to be processed distinctly from the fragment, it must also be presentable as distinct from the fragment.

This paper explores the possibilities for expressing and implementing fragmented media integration with existing and emerging standards. First, we present our main functions for fragmented media integration and deduce the requirements for expressing and processing them. Then we describe the ability of current standards and their implementations to meet these requirements in processing and presenting fragments and their contexts. Finally, we discuss some requirements for media fragmenting and integrated presentation that remain unmet and propose future directions in standards for meeting them. These future directions relate in large part to focussing fragment locating in URI attribute values and fragment display in style specifications.

2 Background and Related Work

Multimedia authoring systems such as CMIFed [12][22] have long provided spatial and temporal measurements as a means of fragmenting media files. In such systems, authors can apply numeric coordinates in space, time or both to media files to extract portions of them for placement in their presentations. This enables, for example, the cropping of image files to include smaller images. It also enables the clipping of audio and video files into shorter segments. These cropped image fragments are typically rectangular in shape. Spatial and temporal fragmenting can be combined in some of these systems to crop the image of a played video, though the dimensions of the crop do not change as the video is played.

Digital video editing tools have long provided *edit lists* as a means of reusing clips taken from a single video. Edit lists consist of a sequences of time stamp pairs denoting clips from the source video. They enable source video files to be reused during editing without modifying them or copying segments of them into smaller files. Many different edit lists can exist for one collection of source video, each playing different cuts from the same source video.

Early work on more structure-based approaches to video clip sequencing includes the Video Retrieval and Sequencing System (VRSS) [4]. The VRSS handled all levels of video production. At the beginning of the process, VRSS segments the video into structured and semantically-significant segments to enable retrieving and sequencing them for display of different edit cuts. It then provides retrieval of the segment and then sequences them for edited cut presentations.

The emerging standard MPEG-7 imposes hierarchical semantic structure on digital media such as video and audio, allowing fragments of it to be labelled for external reference [16][18]. The locations of these fragments within their host files can be defined both temporally and spatially. Multimedia applications integrating these fragments can locate them through their MPEG-7 labels, without needing to know their

spatial and temporal coordinates. Furthermore, spatial coordinates for cropping in temporal clips can change during the progression of the clip, defining the classic user case of "following the car as it drives across the screen". The Institut National de l'Audiovisuel (INA) in France has explored using an MPEG-7-based approach for structuring and fragmenting large-scale digital archives for authoring multimedia applications [2]. The VideoMadeus multimedia authoring tool allows intricate hypermedia integration of fragments from structured video [21].

The MacWeb project explored the issues involved when portions of text files are endpoints in hyperlinks [19]. In this system, our term "fragment" corresponds to the term *"anchor"*, which is a text file segment that is navigated to from outside the text file. The MacWeb project discusses extra information necessary for each endpoint of the link beyond that used for the link as a whole. This includes the endpoint's *type*, which is the role it plays in the link. If the endpoint is an anchor, then the *context* of the anchor is useful information for the endpoint within the link. The context is the portion of the text file around the anchor that should be shown to the user with the anchor when the link is traversed so that the anchor itself and its meaning in the link makes sense to the user. MacWeb's "context" is not necessarily all of the file except the fragment. It is a conceptual context rather than a file boundary concept. However, a context in MacWeb cannot extend beyond the file in which its fragment sits.

The terms "anchor" and "context" are used similarly in the Amsterdam Hypermedia Model (AHM) [12][13], an extension of time and multimedia constructs to the Dexter Hypertext model [11]. An AHM *atomic component* integrates all or part of a media file into a hypermedia document. An *anchor* is then a segment of an atomic component that can be linked to from elsewhere. A *context* in AHM is defined for either or both the source and destination endpoints of a navigational hyperlink. The *source context* states what parts of the presentation change when the link is traversed, thus allowing other parts of the presentation to stay the same after link traversal, providing a sense of continuance during navigation.

The *destination context* of a link is an ancestor of the destination linkend in the AHM document hierarchy. The entire destination context is displayed as a result of the link traversal to its descendent, the actual destination. The *presentation specification* of the destination endpoint can set a style for the anchor to distinguish it from its context. Presentation specification in AHM refer to *channels* through which components of the document get presented. Channels act as abstract presentation peripherals or devices and contain much specification on how media get processed and displayed for a given circumstance.

An important different between AHM context and MacWeb context is that AHM destination context is structured in the integrating document, not in the integrated file. Thus the destination context is not necessarily entirely from the fragment's host file,

nor does it have to contain the fragment in the fragment's host file. Therefore it is the hypermedia author that determines what the context can be, not the media object creator.

The Look of the Link discussed the visualization of source and destination anchors in hypertext documents [23]. That is, it described how hyperlink starting points can be indicated, as well as the targeted fragments in destination presentation files. Here, a number of historic and current visual approaches are described. Some new visual patterns for distinguishing anchor fragments of hyperlinks are also proposed.

3 Requirements

We define the term *"fragment"* as a portion of a file. This article is concerned in particular with the integration of fragments into interactive multimedia presentations. Thus, these will be fragments will be media files, including not just text but also images, graphics and video. The multimedia author will need to specify what portion of each file the fragment is. The multimedia browser will need to processing this specification, retrieve the fragment and display it appropriately. The nature of these fragments' integration will be as endpoints of navigational link traversals and as media objects whose presentation is coordinated with that of other media objects in a multimedia application.

An important part of integrating a media fragment is determining how to treat its *context*. We define a fragment's context as the portion of the fragment's host media file that is not the fragment itself. Although our use of the term may differ from earlier literature, it is consistent with how W3C fragmenting standards use the term [10]. Often, from the user perspective, the context disappears completely, making the fragment the only part of the media file shown in any form to the user. At other times, having the user understand the context in which a fragment appears is important. In such cases, the user should be able to perceive and navigate through both the context and fragment, but should still be able to distinguish between the two because the author would not have specified access to the fragment if the distinction was not important to the user.

Given these definitions and perspectives of fragments and contexts, we present below a list of requirements for media fragment integration. This list is used to structure our discussion of media fragment integration in the remainder of this paper.

- *spatial fragmenting* — There should be a *spatial* means of specifying what portion of the media object makes up the desired fragment.
- *temporal fragmenting* — There should be a *temporal* means of specifying what portion of the media object makes up the desired fragment.
- *nominal fragmenting* — Fragments that are given a name in should be referable to by integrating formats with this name. This name can either be assigned by the

host media file or by another external data source associating names with location specifiers in the media file.

- *structural fragmenting* — There should be a *structural* means of specifying what portion of the media object makes up the desired fragment. This means of locating will be dependent on the data format representing the media.
- *context removal* — The display of the context should be removable without affecting the fragment's appearance.
- *fragment distinction* — The specification of how the fragment appears should be able to be distinct from that of its context.
- *initial navigation to fragment* — The fragment should be made readily apparent upon initial access.
- *integration control over fragment handling* — The integrating multimedia presentation author should able to specify each fragment's context removal, distinction and initial navigation, rather than having it decided by the browser or the media content creator.

4 Current Standards-based Solutions

The recent development of several XML-based structured media formats empowers further the integration of media files. One impact is that the structure of integrated media can be used as the basis for specifying fragments of it. Such a fragment can then be displayed alone, or differently from, its context. Another impact is that a common style mechanism can be applied to multiple XML-defined media components. This allows details of how a fragment is presented to be specified outside of its context. These features enable current standards to fulfill some of the requirements for fragmented media integration.

The main Web formats for media presentation are HTML, SVG and SMIL. Hypertext Markup Language (HTML) defines the XML encoding of text documents for presentation on the Web [20]. Scalable Vector Graphics (SVG) offers two-dimensional vector graphics as a structured representation of visual content, also using XML [8]. The Synchronized Multimedia Integration Language (SMIL) is an XML encoding for multimedia on the Web. Version 1.0 of SMIL came out in 1998 with the basics for distributed multimedia [14]. SMIL 2.0 is expected to be released soon as a recommendation [6]. It has all SMIL 1.0 features and adds many more, defining state-of-the-art Web-based multimedia.

In terms of presentation and integration, HTML, SVG and SMIL have much in common. All three define presentations that can be integrated, in whole or in part, into other presentations. Furthermore, all three have facilities for integrating the display of other media into their own presentations. All three can integrate other media either as presented directly or as the destination of a navigational hyperlink. Finally, Cascading

Style Sheets (CSS) provides a unified way to specify the style which XML-structured media is to be presented, and has been incorporated into XHTML, SMIL and SVG [3].

HTML is a multimedia language in the sense that it can refer to other media files to integrate their presentations with its text content. The HTML `img` element refers to an external image file for in-line display. The `iframe` element allows external HTML files to be integrated into this HTML file's display. SVG has an `image` element for placing displays from external image files with its graphic objects. SMIL has a more general-purpose constructs for media integration with its *media object elements*.

4.1 Current Measured Fragmenting

CSS style sheets for integrated displays can simulate some spatial cropping in HTML, SVG and SMIL. They can do so by placing the display of entire image files in boxes that are sized only for the fragments. Then they would use the `overflow:hidden` property assignment to remove the display of the image areas beyond the box. The `vertical-align` and `text-align` properties provide the basic vertical and horizontal alignment of the image in the box that controls what sides get cropped off. However, the crops can only be centered or laid along corners and edges.

SMIL has XML constructs that imitate cropping much like these CSS properties, as does SVG. SMIL 1.0 attributes for *regions* in the layout have the CSS equivalent constructs for the same limited spatial fragmenting. SMIL 2.0 introduces the `regPoint` construct, which can position an image anywhere within its region, thus being able simulate cropping any rectangle of the image. However, all of these properties and attributes do not describe true cropping — they refer to an entire image that has its context hidden rather than directly referring to the desired fragment of the image and displaying that in its entirety. Furthermore, these standards provide no means for having link destinations be spatial fragments.

SMIL1.0 has the `clipBegin` and `clipEnd` attributes, which take a temporal media object such as an audio or video file or stream and present only part of it, starting it and ending it at particular times. The attributes can be assigned to both media object and hyperlinking elements, providing both types of temporal media fragment integration. The SMIL 2.0 `mediaRepeat` attribute performs a particular kind of temporal fragmenting: it removes any top-level repeats that happen on an media file, such as one encoded with animated GIF. Thus, the `mediaRepeat` attribute effectively clips the first iteration of a repeat as its temporal fragment.

4.2 Current Nominal Fragmenting

SMIL 2.0 `clipBegin` and `clipEnd` attributes also approach a time-based subset of nominal fragmenting with *media clip markers*. When their values being with

"marker=", what follows is a URI locating a timestamp within the media file. If the URI begins with '#' then what comes after is a name of a timestamp encoded in the media file itself. Below is an example of such media clip markers.

```
<video src="http://www.examples.org/romeo.vidx"
      clipBegin="marker=#act2scene3line2"
      clipEnd="marker=#act2scene3line3"         />
```

If the URI begins with a file on the Internet followed by a '#' then the file provides third party markers for the media file. What comes after the '#' is a string the third party file associates with a point in time in the media file. This could potentially be used to extract timestamps from third party MPEG-7 files that annotate the host media files from which fragment comes.

Although SMIL media clip markers approach it, they do not provide full nominal fragment. First of all, they only clip the media file in terms of time. Furthermore, they do not use one name to locate the fragment itself, they use two names for locating the bounding timestamps of the fragment.

Nominal fragmenting is more directly implemented with current URI *fragment identifiers*. The XML term "fragment" refers primarily to the *fragment identifier* portion of a URI: the part after the '#', which defines a portion of the file referenced. These can be used by the `src` and `href` attributes of XHTML, SMIL and SVG. In current implementations of these formats, the fragment can refer to the element with the named anchor or unique identifier in XML-defined integrated media. This requires that the integrated media have an identified element encapsulate the desired fragment. The resulting behavior is discussed in Section 4.4.

4.3 Current Structural Fragmenting with XPointer

The clear emerging standard for providing more fragmented media fragmenting is *XPointer* [7]. XPointer defines the locations of fragments of XML code in terms of XML structure. It enables any subcomponent of an XML document to be referenced, whereas without XPointer, references can only be made to XML elements with unique identifiers. In principle, any URI attribute in any XML document can have an XPointer reference as its value. This includes, for example, the `src` and `href` attributes of XHTML, SMIL and SVG. However, XPointer is currently not implemented in any of these language's browsers, so XPointer values for these attributes cannot yet be processed. While XPointer can separate a fragment from its context, it explicitly leaves it up to its applications to specify how to process or present a fragment in relation to its context. Since XPointer is not yet released, none of its applications have formalized how fragments and their contexts are to be handled differently.

However, a small subset of XPointer is widely used, and thus provides a precedent. XPointer specifies the *bare name XPointer value* as a shortcut for identifier ref-

erencing that is consistent with the use of fragment identifiers in `href` attribute values in XHTML. That is, if a URI attribute value has a `'#'` character followed by a name, then XPointer accepts that as locating the element in the reference document that is assigned that unique identifier. As such, XHTML `href` attribute named anchor references use valid XPointer values. For example, the following bare name XPointer value

```
http://www.examples.org/index.html#foo
```

is valid XPointer and is equivalent to the full XPointer value

```
http://www.examples.org/index.html#xpointer(id("foo"))
```

In addition to the presentation-oriented and user-perceived distinction between fragment and context, the use of XML-based solutions makes *syntactic context* also important to consider. Even if only the fragment is to be presented, often the XML code for its context needs to be processed for the fragment's presentation to be rendered. For example, CSS code at the beginning of the document could affect the display of the fragment, even if the CSS was not in the fragment code itself. Rendering the fragment would need to account for this CSS code to determine its appearance. Similarly, spatial transform elements in SVG can affect the display of elements elsewhere in the same document. When rendering an SVG fragment, spatial transforms in the context that applies to the fragment would need to be processed.

4.4 Current Initial Navigation to Fragments

HTML, SVG and SMIL all allow fragment identifiers to be used in URIs that address them. Typically, upon initial access they load the entire file into the presentation space but scroll forward along the vertical, horizontal and temporal axes, whichever apply, until the object is shown. Normal access to the entire file is then resumed.

The display of an XPointer-located fragment with its *HTML* context is illustrated in Figure 1. It shows the fragment of text "It is the east, and Juliet is the sun" from the play *Romeo and Juliet* in a current HTML browser. Here, the display is scrolled to where the fragment appears so that the user can see it at first. But after initial access the user if free to scroll through the rest of the play. This would be preferred to initially scrolling to the beginning of *Romeo and Juliet*, and requiring the user scroll to find the fragment. One the other hand, if this passage from Shakespeare appears as a caption in a multimedia display, one typically does not want scrollbars to appear with the passage to allow access to the rest of the play. This would clutter the visual appearance of the display as a whole.

When an *SVG* document is accessed with an XPointer fragment, the closest ancestor `svg` element is displayed according to its view specifications. The SVG specification does not state what happens to its context `svg` elements, and there are no current implementations of these features to set a precedent with. SVG also defines an *SVG view specification* syntax that can be used as a URI fragment. It provides the co-

ordinates to which the initial view should be scrolled, and specifies other initial user interface activities such as zooming and panning. SVG view specifications also provide *transforms*, which can alter this presentation of the graphics from its default.

SMIL has adopted the HTML constructs for hyperlinking to XML-located fragments, keeping equivalent semantics as they apply to timed multimedia. The `href` attribute of SMIL `<a>` element hyperlinks can refer to a portion of the same or an external SMIL presentation by having as its value the unique identifier of an element in that presentation. Triggering such an access causes the presentation to be loaded and forwarded, or "seeked" to the point in time when that element is scheduled to start. As with HTML, such references are valid XPointer.

The behavior that XHTML and SMIL browsers apply to the types of XPointer attribute values described above is that the referenced document be loaded for presentation in its entirety, but the presentation scrolls so that the referenced fragment appears. This behavior is not specified in XHTML. However, when a link is triggered that is defined with a fragment value of the `href` attribute, current browsers typically display the whole document, but scroll it so that the beginning of the fragment is at the top of the window. This is the only distinction made between fragment and syntax: at initial access, the user can see where the fragment starts. However, the user cannot see

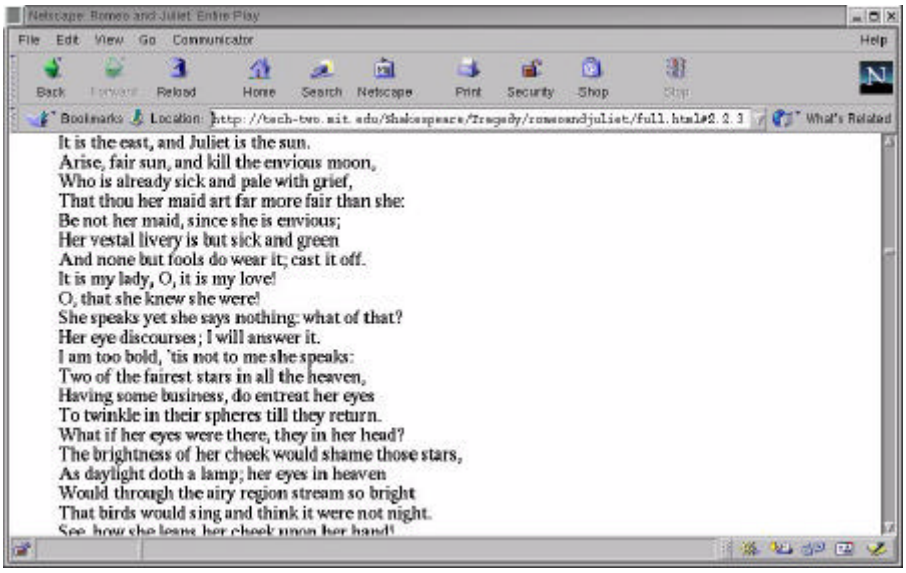

Figure 1. Fragment "It is the east, and Juliet is the sun" of *Romeo and Juliet* shown in a current HTML browser.

where it ends. After this initial display, the user can scroll to anywhere in the document in a manner that does not distinguish fragment from context.

The corresponding forwarding in SMIL linking is similar, but applies to time rather than space. Forwarding in SMIL does not mean scrolling spatially through the flow of text but "seeking" forward along the timeline. When a SMIL hyperlink that has an XPointer bare name value to within a SMIL presentation fires, that presentation is loaded and played starting at the time that the referenced element would start playing. The default behavior is that the linked-to presentation continues playing until the end of the presentation as a whole, which is not necessarily the ending time of the fragment. Once the fragment starts playing, its begin is not longer indicated. Furthermore, subsequent navigation within the SMIL document makes no distinction between fragment and context. However, the SMIL `src` attribute of media object elements specifies neither this behavior nor any other for the use of bare name XPointer values or other XPointer values.

4.5 Current Fragment Distinction with Target

The primary means on the Web of varying the appearance of a piece of a document is through style specifications. The primary example of style specification on the Web is the use of CSS to define the style for presenting XML documents in general and HTML documents in particular. CSS is currently evolving to be able to apply different styles for fragments and their context.

For the most part, a CSS style sheet is a list of *rules*, where each rule consists of a *selector*, specifying some components of the document, and a *declaration block*, which states how those components are to appear. If a selector could specify either a fragment or its context, then a distinct style could be applied to it. One type of CSS selector is the *pseudo-class*, which matches a document portion by a manner other than its location in the XML structure. While the current version of the CSS recommendation, CSS2 [3], does not distinguish the fragment, the version of CSS currently under development, CSS3 [17], does. Among CSS3's proposed new selectors [9] is the pseudo-class `target`, which applies a particular style to those portions of a document that are targeted by a fragment identifier URI.

For example, significant pieces of text are often *highlighted*, typically by being given a yellow background, to distinguish them. The piece of CSS code below uses the `target` pseudo-class to define such a yellow background as distinguishing fragments from their context. This code can be included in the CSS style sheet for an HTML document to cause fragments of it to be highlighted in yellow when linked to or when integrated into other presentations.

```
:target { background-color: yellow }
```

The display of a fragment as visually distinct is illustrated in Figure 2.

SVG's `viewTarget` attribute of the `view` element and the `viewTargetSpec` parameter of SVG view specifications locate a target in the SVG document being presented that is to be highlighted. The nature of this highlighting is not defined by the SVG specification. Perhaps with the release of CSS3, SVG's target could be linked with CSS's target so that CSS target style would define SVG target highlighting.

4.6 Current Context Removal with CSS3

CSS can also remove the fragment's context from the presentation. First, CSS3 can select the context by combining its `target` pseudo-class with another pseudo-class it introduces: the negation, or `not`, pseudo-class. Then CSS2 can apply to the selected context a style that effectively removes it from display: a `display` property with a `none` value. This property assignment causes the objects to not be displayed. It also makes the objects' implicit height and width zero, thus affecting the spatial layout of other objects that are displayed in the presentation. These CSS constructs are combined for removing the context in the following CSS3 rule:

```
:not(:target) { display: none }
```

4.7 Current Integration Control over Fragment Handling

The emerging format *XML Fragment Interchange* addresses *syntactic* fragmenting and integration [10]. This format enables fragments of XML code to be parsed when

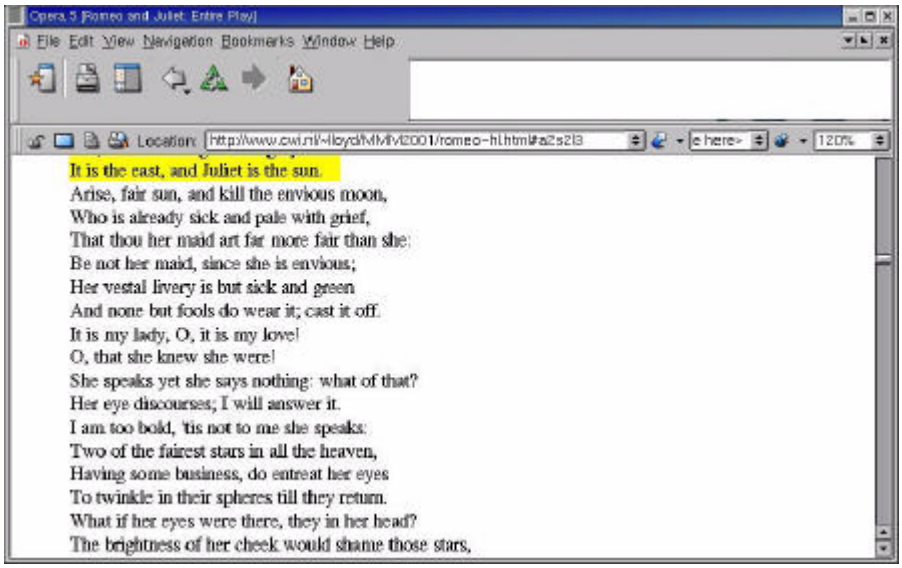

Figure 2. Fragment from Figure 1 shown as highlighted.

included as entities in other XML code. It includes accounting for code in the fragment's context when the fragment is parsed where it is included. However, XML Fragment Interchange address fragments and their context primarily for *parsing*, whereas this paper is concerned with processing the semantics of fragments and their context that relate to rendering presentations. It does not formalize how the issues are handled for processing outside parsing.

5 Open Issues and Potential Future Directions

Current and emerging Web standards provide some constructs for media fragment integration, as discussed in Section 4. However, significant aspects of media fragment integration remain unaddressed by these standards. Furthermore, the solutions that do exist are often not unified or consistent. They are often scattered across different aspects of XML document structure. Also, different standards sometimes provide different solutions for the same aspects of media fragment integration. This section addresses these shortcomings by suggesting new tools for missing solutions and by discussing the unification existing solutions into one approach that is consistent across standards.

5.1 Potential Future Directions for Fragment Locating

Whether the basis for fragmenting is spatial, temporal or structural, all fragmenting locates a particular portion of a file. In the emerging Web architecture, locating is typically specified in URI attributes. Putting all fragment specification, be it spatial, temporal or structure, in the fragment part of a URI (after the '#') would introduce more unification, consistency and simplicity to Web fragmenting. XPointer introduces *schemes* as a place to put new means of locating fragments on the Web [7], making itself is one such scheme. XPointer specifies that the long form of the fragment portion of a URI begins with the name of the scheme followed by the address using the scheme in parentheses. Thus, to keep all fragment addressing consistently located in URI fragments, each type of fragment addressing that currently don't have XPointer schemes can be given one.

Another important characteristic of the URI is that it is the only data in an integrating document that is communicated over the network to the server. There is no stated requirement that the server process the fragment data. In fact, in current systems, the server receiving a fragment request will typically still transfer the whole file, leaving it to the client to perform the fragment extraction. However, future systems may instruct the server to perform the extraction, when possible. This results in only the fragment being transferred over the network, reducing network traffic. This would be the most efficient use of bandwidth if the context of the fragment is not displayed and not otherwise needed for processing the fragment. If fragment locating information is not put in the URI but instead put elsewhere in the XML structure, then there

is not possibility of performing the fragment extraction at the server and lowering bandwidth requirements.

There are thus three benefits to putting all fragment locating information in the URI. One is that authors would know where to find all information that locates a fragment. This place for fragment locating would be unified over all constructs and standards. The authors also benefit the means for media fragmenting for all Web standards could itself be standardized as one format. Unifying the syntax and location of fragment locating code also makes it easier to standardize and incorporate, syntactically and mechanically, into multiple document formats and their browsers. Network efficiency is the final beneficiary of UIR-centered fragmenting, since it enables servers to perform fragment extraction when the context is not needed for processing.

5.1.1 Potential Future Directions for Measured Fragmenting

Current spatial fragmenting solutions, as presented in Section 4.1, typically use four measurements out of the following six: left, right and width and top, bottom and height. The units for each measurement are typically pixel units of the rendered image or percentages. These measurements could be used as parameters is a new number-based locator scheme. Their values, for consistency, could be the same as now used uniformly in CSS, HTML, SVG and SMIL. Below is an example URI using such a hypothetical scheme called “coords”.

```
http://www.examples.org/juliet.jpg#
coords(left="50px",right="30px",top="60px",height="100px")
```

Current temporal fragmenting solutions, as presented in Section 4.1, typically specify the begin time and end time of the fragment clip in terms of the timing of the file as a whole. These timestamps could be used as parameters is a number-based locator scheme. These parameters can be used in the same scheme use for spatial fragmenting, since the coordinates won’t conflict with each other. Furthermore, using one scheme for both spatial and temporal fragmenting allows them to be used together more easily, such as for locating a spatial crop of a video clip. Below is an example URI using such the hypothetical coords scheme.

```
http://www.examples.org/romeo.mpg#
coords(left="50px",right="30px",top="60px",height="100px",
begin="0:30:15",end="0:31:20")
```

The equivalent of SMIL’s media clip markers can be used in the coords scheme to add some nominal fragmenting, such as in the following example.

```
http://www.examples.org/romeo.mpg#
coords(left="50px",right="30px",top="60px",height="100px",
begin="marker=#act2scene2line3",
end="marker=#act2scene2line4")
```

5.1.2 Potential Future Directions for Nominal Fragmenting

Truer nominal fragmenting comes from using one name directly representing the fragment itself to extract it. Fragments of HTML, SVG and SMIL files can be nominally located with bare name XPointer values. It would be consistent to introduce using the same URI syntax for non-XML media with named fragments, as in the following code.

```
http://www.examples.org/romeo.vids#act2scene2line3
```

Third party nominal fragmenting could be introduced into the URI fragment with a locator scheme for separate media annotation files such as those encoded in MPEG-7. Below is an example of how an MPEG-7-based nominal fragment URI may appear.

```
http://www.examples.org/romeo.mpg#
mpeg7(annotFile="http://www.examples.org/romeo.mpg7",
clip="act2scene2line3")
```

5.1.3 Potential Future Directions for Structured Fragmenting

Structured fragment addressing in XML is already completely contained in the URI. Annotation formats such as MPEG-7, however, define non-XML structure imposed on a media file. Location specifications using MPEG-7 structure could be added to the hypothetical MPEG-7 locator scheme described above. Below is an example of how an MPEG-7-based structural fragment URI may appear.

```
http://www.examples.org/romeo.mpg#
mpeg7(annotFile="http://www.examples.org/romeo.mpg7",
act="2", scene="2", line="3")
```

5.2 Potential Future Directions for Context Removal

In Section 4.6, we discussed how the CSS property assignment `display:none` makes context removal possible for current XML-encoded displays. To make this property assignment function in multimedia, we propose extending the semantics of `none` to apply in the same way to time as well as space and visual display. The premise of this semantic extension is that the object whose style is assigned the `none` property should have no affect on the timing of the rest of the presentation. Whatever influence the object timing itself would have had on the presentation when its display wasn't assigned `none` would be removed. One implication is that the duration of such a non-displayed object would be treated as if it were zero. Thus, if it appears in a SMIL sequence, the object after it in the sequence starts when the object before it ends. Another implication is that an undisplayed element never triggers any events, so any timing dependent on undisplayed object events never happens.

5.3 Potential Future Directions for Initial Navigation to Fragment

There are many ways to navigate to fragments in such a wide variety of presentation formats. Therefore, there are many potential ways to specify initial navigations that would make a fragment best perceived in its context. This results also in many issues and complications to address in developing the means of specifying and processing such initial navigations. We leave most of these to future work. What we present here is two alternatives. One is that initial navigation be put in the URI fragment, as is done in SVG with SVG view specifications, which are essentially a distinct XPointer-enabled locator scheme. In order to apply it more broadly, such a locator scheme could be applied to any media that can be scrolled in one or both spatial dimensions. Navigation schemes could be appended to fragment schemes in URI fragments, causing both types of processing to occur sequentially.

The other alternative is to put initial navigation in style sheets with a simple yet useful potential extension to what CSS currently provides. CSS has constructs for aligning objects within the boxes they are presented in: `vertical-align` and `text-align`. These constructs specify, respectively, the vertical and horizontal positioning of an object within its box. Similar vertical and horizontal alignment could position a fragment within its presentation window, effectively automatically performing the scrolling of the entire file's display that is necessary to do so. New CSS properties could be made for each direction of alignment that apply to window scrolling instead of box positioning. These new fragment-oriented window-scrolling properties would take the same values with the same semantics as their box positioning counterparts. The CSS code below, with such extensions, defines the de facto standard behavior of current HTML browsers, that of scrolling so that the fragment's top is at the top of the window, as shown in Figure 1 and Figure 2.

```
* { fraginit-vertical-align: text-top }
```

This de facto browser fragment scrolling does not provide good initial perception of context if information important to appreciating the fragment is directly above it. For many fragments, useful information is in the context close to the fragment both above and below it. Similarly, for fragments on wide lines, useful context display may be both to the left and right of the fragment. In both these cases, scrolling the display so that the fragment is centered in the window along both axes would provide the best initial navigation. Such a display is illustrated in Figure 3. Potential code for defining the alignment of the display is below.

```
* { fraginit-vertical-align: middle;
    fraginit-text-align:      center  }
```

Further initial navigation could be specified in CSS by introducing CSS properties that are the equivalents of what are in SVG view specifications. The SVG `viewBox` construct, which provides the coordinates for initial scrolling, would be particu-

larly valuable for initial access. However, it must use coordinates and cannot simply refer to the fragment.

5.4 Potential Future Directions for Integration Control over Fragment Handling with Integration Style

This paper has presented several ways in which CSS style sheets could be used to specify how fragments and their contexts are displayed. A shortcoming of this approach in the current Web architecture is that style sheets that apply to external XML code integrated into a presentation can only be written by the author of that external XML code or by the user it is presented to. There is currently no mechanism in HTML, SVG or SMIL with which the author of an integrating multimedia presentation can write CSS style sheets that apply to the presentation of the media being integrated.

However, HTML, SVG and SMIL each have constructs for applying CSS to constructs *within* the same XML file. HTML and SVG both have a `style` element and attribute, which allows CSS style sheets to be applied to text and color objects con-

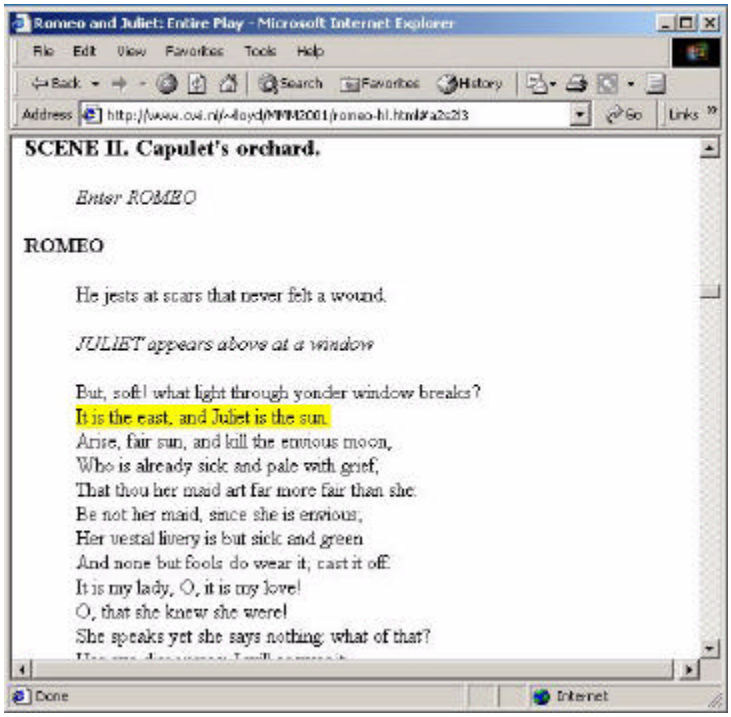

Figure 3. Fragment from Figure 1 shown as highlighted and centered.

tained in the same HTML or SVG file. SMIL does not directly contain directly-presentable media to which style could be applied, and thus has no `style` construct. However, SMIL's layout constructs do use direct CSS equivalents to position media that is integrated. Furthermore, SMIL allows an alternative layout to be defined for a presentation that uses CSS directly, although this is currently not implemented.

These constructs for internal styling could be extended and modified to apply to *external* styling: the style of presentation of external media and fragments that are integrated into a presentation. This styling would need to apply to media that is both inked to and integrated into a coordinated display. If HTML, SVG and SMIL all used the same styling constructs, and if these constructs applied to both internal and integrated media, then it would make style authoring easier to learn. It would also make these constructs *modularizable*, and thus more easily incorporated into future XML formats for integrated presentation [1].

One possibility is a new `channel` element that replaces the `style` constructs of HTML and SVG and the `region` constructs of SMIL. This `channel` elements would correspond with AHM's channel, acting as a conduit through which all media gets processed and presented, and which specifies all aspects of how this media is to be processed and presented. It would contain or refer to CSS code that defines the style of internal media, integrated media, and media fragments and their context.

The `channel` element would be incorporated into the evolving Web architecture more cleanly if all aspects of style were put in CSS. This would include the extensions already suggested. It would also include introducing new CSS properties to replace the XML constructs for fragment integration described in Section 4. There are many more details for `channel` element and CSS property unification than would fit in this paper, so we leave these to future work.

6 Conclusion

Integrating media objects into multimedia presentations is limited if these objects are treated as atomic at the file level. Structured document data, as provided by XML, enables media file fragments to be accessed and integrated in a unified manner. Several recent and emerging standards, such as SMIL, XPointer, SVG and CSS contribute to this ability. We described here the issues and possibilities for integrating media fragments, provided an overview of current solutions, and proposed further directions. How current standards and our proposed extensions meet our proposed requirements is illustrated in Table 1.

Table 1. Fragment integration requirements with their current and proposed solutions

Requirement	Current Solution	Potential Future Direction
spatial fragmenting	**region** element **fit** and measurement attributes	**coords** locator scheme
temporal fragmenting	temporal clipping **mediaRepeat** attribute time markers	
nominal fragmenting	ID, SMIL media clip markers	make applicable to non-XML
structural fragmenting	XPointer	**mpeg7** locator scheme
context removal	XML Fragment Interchange **:not(:target)** **{ display: none }**	applying **display:none** to time
fragment distinction	:**target** {}	*none needed*
initial navigation to fragment	start forwarded to with **href="#"** in HTML, SVG and SMIL SVG view specifications	add **fraginit-vertical-align** and **fraginit-text-align** properties to CSS add SVG view specification as CSS properties
Integration Control over Fragment Handling	HTML and SVG **style** SMIL **region** SVG view specifications	unified **channel** construct related current XML constructs introduced as CSS properties

Our first primary conclusion is that all locating of media file fragment should be put in fragment portion of URI attribute values. This facilitates authoring and enables improved network efficiency. The second primary conclusion is that style specification and processing is a good place for handling fragment and context presentation issues. Thus, it is beneficial to extend how style sheets can be used in integrating media and media file fragments. These two conclusions together indicate that most aspects of fragment integration should be taken out of other XML structure such as local attribute values and put in either URI fragments or style sheets. This results in necessary extensions to current URI fragments and style sheets, for which we propose specific possibilities.

7 Acknowledgements

The research for this paper was funded in part by the Multimedia Information Analysis (MIA) project and by the RTIPA project. Essential feedback was provided by Steven Pemberton, Jacco van Ossenbruggen and Lynda Hardman.

8 References

[1] M. Altheim, F. Boumphrey, S. Dooley, S. McCarron, S. Schnitzenbaumer and T. Wugofski. *Modularization of XHTML*. W3C Recommendation. April 10, 2001. http://www.w3.org/TR/.

[2] G. Auffret, J. Carrive, O. Chevet and T. Dechilly. "Audiovisual-based Hypermedia Authoring: using structured representations for efficient access to AV documents". *Proceedings of the Tenth ACM Conference on Hypertext (Hypertext '99)*, Darmstadt, Germany. February 1999, pp. 169-178.

[3] B. Bos, H. W. Lie, C. Lilley, and I. Jacobs (eds). *Cascading Style Sheets, level 2 — CSS2 Specification*. W3C Recommendation. May 12, 1998. http://www.w3.org/TR/.

[4] T.S. Chua and L.Q. Ruan. "A Video Retrieval and Sequencing System". *ACM Transactions on Information Systems*. vol. 3, no. 4, October 1995. pp. 373-407.

[5] T. Boutell. *PNG (Portable Network Graphics) Specification*. W3C Recommendation. October 1, 1996. http://www.w3.org/TR/.

[6] A. Cohen, et. al. (eds). *Synchronized Multimedia Integration Language (SMIL) 2.0 Specification*. W3C Recommendation. August 7, 2001. http://www.w3.org/TR/.

[7] S. DeRose, E. Maler, and J. Ron Daniel. (eds). *XML Pointer Language (XPointer) Version 1.0*. W3C Last Call Working Draft. January 8, 2001. http://www.w3.org/TR/. (work in progress)

[8] J. Ferraiolo. (ed). *Scalable Vector Graphics (SVG) 1.0 Specification*. W3C Recommendation. September 4, 2001. http://www.w3.org/TR/.

[9] D. Glazman, T. Çelik, I. Hickson, P. Linss and J. Williams (eds). *CSS3 module: W3C selectors*. W3C Working Draft. January 26, 2001. http://www.w3.org/TR/. (work in progress)

[10] P. Grosso and D. Veillard (eds). *XML Fragment Interchange*. W3C Candidate Recommendation. February 12, 2001. http://www.w3.org/TR/. (work in progress)

[11] F. Halasz and M. Schwarz. "The Dexter Hypertext Reference Model". *Communications of the ACM*, vol. 37, no. 2. February 1994, pp. 30-39.

[12] L. Hardman. "Modelling and Authoring Hypermedia Documents". *Ph.D. Thesis*, University of Amsterdam, March 8, 1998.

[13] L. Hardman, D.C.A. Bulterman, and G. van Rossum. "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model". *Communications of the ACM*, vol. 37, no. 2, February 1994, pp. 50-62.

[14] P. Hoschka (ed). *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*. W3C Recommendation. June 15, 1998. http://www.w3.org/TR/.

[15] International Organization for Standardization/International Electrotechnical Commission. *MHEG-5: Coding of multimedia and hypermedia information -- Part 5: Support for base-level interactive applications*. 1997. International Standard ISO/IEC 13522-5:1997 (MHEG-5).

[16] International Organization for Standardization/International Electrotechnical Commission. *MPEG-7: Context and Objectives*, 1998. (work in progress)

[17] E. A. Meyer, and B. Bos (eds). *Introduction to CSS3*. W3C Working Draft. May 23, 2001. http://www.w3.org/TR/. (work in progress)

[18] F. Nack and A.T. Lindsay. "Everything You Wanted to Know About MPEG-7: Part 1". *IEEE Multimedia*, vol. 6, no. 2. July-September 1999, pp. 65-77.

[19] J. Nanard and M. Nanard. "Should Anchors be Typed Too? An Experiment with MacWeb". *Proceedings of the Fifth ACM Conference on Hypertext (Hypertext '93)*, Seattle, USA. March 1993, pp. 51-62.

[20] D. Raggett, A. L. Hors, and I. Jacobs. *HTML 4.01 Specification*. W3C Recommendation. December 24, 1999. http://www.w3.org/TR/.

[21] C. Roisin, T. Tran_Thuong and L. Villard. "A Proposal for a Video Modeling for Composing Multimedia Document". *Proceedings of the 7th International Conference on Multimedia Modeling (MMM 2000)*, November 2000, Nagano, Japan.

[22] G. van Rossum, J. Jansen, K. S. Mullender and D.C.A Bulterman. "CMIFed: A Presentation Environment for Portable Hypermedia Documents". *Proceedings of the First ACM International Conference on Multimedia (Multimedia '93)*, Anaheim, USA, August 1993, pp. 183-188.

[23] H. Weinreich, H. Obendorf and W. Lamersdorf, "The Look of the Link — Concepts for the User Interface of Extended Hyperlinks". *Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01)*, Aarhus, Denmark, August 2001, pp. 19-28.