A Model for Editing Operations on Active Temporal Multimedia Documents

Jack Jansen, Pablo Cesar, Dick C. A. Bulterman
CWI: Centrum Wiskunde & Informatica, Amsterdam
{Jack.Jansen,Pablo.Cesar,Dick.Bulterman}@cwi.nl

ABSTRACT

Inclusion of content with temporal behavior in a structured documents leads to such a document gaining temporal semantics. If we then allow changes to the document during its presentation, this brings with it a number of fundamental issues that are related to those temporal semantics. In this paper we study modifications of active multimedia documents and the implications of those modifications for temporal consistency. Such modifications are becoming increasingly important as multimedia documents move from being primarily a standalone presentation format to being a building block in a larger application.

We present a categorization of modification operations, where each category has distinct consistency and implementation implications for the temporal semantics. We validate the model by applying it to the SMIL language, categorizing all possible editing operations. Finally, we apply the model to the design of a teleconferencing application, where multimedia composition is only a small component of the whole application, and needs to be reactive to the rest of the system.

The primary contribution of this paper is the development of a temporal editing model and a general analysis which we feel can help application designers to structure their applications such that the temporal impact of document modification can be minimized.

Categories and Subject Descriptors

D.3.2 [**Language Classifications**]: Specialized application languages; I.7.2 [**Document and Text Processing**] Document Preparation - Languages and systems.

General Terms

Design, Experimentation, Languages.

Keywords

Declarative languages, Dynamic transformations, Multimedia application design.

1. INTRODUCTION

In the first generation of multimedia documents, the document format served as a simple transport wrapper for a byte stream of content. In the second generation, the document often contained additional temporal, styling and structuring primitive that allowed the transformation for interoperable playback on compliant multimedia players and some user interaction. In both cases, the multimedia content is dynamic, temporally and spatially constrained by the document, but the document itself is static: the relationships described by it do not change. Currently, multimedia documents are entering a new generation: the document is embedded as part of a larger application, where the content of the document may be adapted during rendering time, based on the needs and directives of the embedding application. These adaptations can change the underlying multimedia data, but also the general presentation structure. Examples of such applications include live editing of broadcast content [10] and interactive multimedia applications that make use of Web services [11]. In this paper, we use the term *multimedia transformations* to refer to the process of adapting various aspects of a multimedia document while it is playing.

A representative example of dynamic multimedia applications is a video conferencing system that allows multiparty playing of an electronic board game [23]. The application needs to support multi-party social communication, guided by verbal and physical cues from users, as well as camera actions that are driven by the game logic. Requirements for this type of multimedia applications include: the interoperability between the game logic and the video conferencing system, the ability to switch between cameras for improving communication, and the ability to control cameras - crop video of one camera and track faces - based on verbal and physical cues. This kind of application requires a wide range of multimedia transformations: layout, content selection and adaptation, and presentation enrichment. A key challenge of this type of application is that transformations to the document model occur in real-time and affects active nodes in the presentation.

In most existing work, the interface between the multimedia rendering engine and the processes that steer the document transformation is based on an ad-hoc solution that is targeted to a specific use case. This ad-hoc approach limits reusability and makes long-term maintenance of the application difficult. We believe that there is a need for an in-depth study of the types of multimedia transformation that are available at an abstract level, clustering them into levels of complexity and feasibility. This facilitates the development of structured solutions that address each of the multimedia transformation types.

This paper provides a classification of multimedia transformations into a consistent model that can be applied to a generic set of uses. The main contribution of this paper is a taxonomy of multimedia transformations, together with mechanisms for implementing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DocEng '10, September 21-24, Manchester, United Kingdom.
Copyright 20XX ACM XX-X-XX…$5.00.

clusters of the transformations in a uniform manner. We feel that this work is important for various groups of document and system architects. For *applications developers* the model permits de-coupling application logic from a specific multimedia rendering environment, through the use of a set of well-defined interfaces. For *language designers* our taxonomy allows the partitioning of transformational functionality "in the right place" within the language, making it more efficient and effective. It also allow languages to support "safe" transformations that do not violate the temporal graph associated with the multimedia content.

This paper is structured as follows. Section 2 discusses previous work in the area of multimedia transformation, and shows how our example application relates to that. Section 3 provides a complete taxonomy of multimedia transformations, classifying them based on their inherent complexity. Sections 4 validates the taxonomy from the viewpoints of language development, by applying it to all possible atomic modifications on a SMIL document. Section 5 then applies the taxonomy and the resulting atomic SMIL modifications to the design of an application. We finish with discussion of where the model could go from here, related work and conclusions.

2. MOTIVATION AND RELATED WORK

A number of systems exist that implement adaptive multimedia using multimedia document formats of the previous generation. The challenge here is to ensure multimedia playback in clients with different configurations. This is solved by performing a number of transformations of the multimedia document *prior* to delivery.

Examples of such systems include Cuypers [16], Multimedia for You (MM4U) [21], Web, Adaptation, and Multimedia (WAM) [13], and Media Spatial Temporal and Interactive (MSTI) [17]. Cuypers uses a number of processing steps (e.g., semantic structure and quantitative constraints) to generate the final presentation. MM4U is a layered architecture that assembles different media objects (e.g., temporal course and spatial layout) and, in the end, generates one concrete multimedia format (e.g., Flash) for a specific device and user. WAM proposes an architecture that uses an adaptation and a negotiation module, and a context module embedded in the device. Finally, MSTI performs document adaptation applying the concept of scalability. MSTI uses spatial, temporal, and interactive descriptions for adapting a presentation to the characteristics of a multimedia terminal. Apart from general adaptation systems, as the ones described above, specific solutions for layout adaptation [4], genre adaptation [15], and dynamic multi-device adaptation [9] have been proposed elsewhere.

The previous examples do not consider live transformations of the multimedia document, but only provide pre-presentation modifications. Typically, a final self-contained presentation is created and delivered, ready for consumption in one specific device. But in the last years, there has been an increasing interest on multimedia transformations that happen during rendering time. Examples of such applications include the Ambulant Annotator [8], the Watch-and-Comment framework [7], Live editing [10] and adaptive time-based web applications [11]. The first two provide an infrastructure for enriching television content while watching by adding and removing elements of the multimedia document. Live Editing allows broadcasters to provide extra multimedia presentations to live television programs. The last embeds a multimedia presentation in an interactive web application. Each of these systems tackles some of the concerns addressed in this paper, but provides only an ad-hoc solution for the specific use case of that system.

A more complete example of the current generation of application categories sketched above is the application that triggered the research in this paper: a dynamic multiparty videoconferencing application that adapts the conferencing to an ongoing parallel activity. The application is developed in the context of the TA2 project, which aims to strengthen ties between people who know each other well a priori, but are geographically separated. To this end, a TA2 system provides a platform to connect multiple households through high quality audio, video and communication links. The intention is to increase togetherness by allowing people to partake in a shared activity. Figure 1 shows an artist's impression of the system in operation.

In the project, we are targeting a hardware platform that is representative of the maximum we can expect to deploy in private homes within a three year window, given technical, financial and social constraints. The visual hardware in the home consists of an HD television and audio speakers for output, and three HD cameras and a 3D microphone for input. While video connectivity is HD 720p the central camera is a full 1080p HD camera from which a 720p image is cropped or scaled. This allows a large number of viewpoints to be selected for display at the remote side. Depending on the activity, there may be auxiliary screens or other hardware involved too. Invisible hardware is a 10 Mbps network connection and 2-3 desktop-class machines.

Even though audio and video connectivity are the most visible part of the system, the whole is much more than a simple teleconferencing application. Interaction in TA2 is centered around a shared activity, and the intention is that the platform will eventually support many different activities. At the time of writing we have started experimentation with a first activity, a network-based implementation of the board game *Space Alert*[1]. This is a cooperative strategy game with realtime aspects. As this particular application is a board game, it was decided to have the game itself

Figure 1. Sketch of the Application. The family members are playing a board game and video conferencing with other households.

1 http://czechgames.com/en/space-alert/

run on a touch screen mounted on a table, and to use the main screen only for live video and atmosphere: remote video and game-driven status information against a backdrop of the stars, as most people will be familiar with from traveling on intergalactic spacecraft.

For the main screen, temporal composition is a key requirement. A comprehensive comparison of temporal models can be found elsewhere [19], where the authors describe 29 multimedia interval relationships. They provide an exhaustive comparison with relevant multimedia description formats and conclude that SMIL [5] can represent all 29 multimedia interval relationships sufficiently. Based on this result (and because of its proven characteristics and flexibility), we used SMIL as the basis for the TA2 project. Thus, the contents of the central HD television are controlled by a SMIL presentation, and this presentation needs to interact with the game and with other components, so that the presentation can be changed dynamically. Because the goal of the project is increasing togetherness, the TA2 system contains a number of components that do person detection and speaker detection, and an orchestration component that uses input from those two components and the game to determine what is currently happening in the person-to-person interaction. From this data, it decides what output to show on the screen.

This application is quite distinct from those sketched in [8], [10] and [11]. Where multimedia presentation is the core functionality of those applications, TA2 has person-to-person communication and shared game play as its main focus. Multimedia presentation plays a somewhat more incidental – albeit quite important – role.

3. TAXONOMY OF MULTIMEDIA DOCUMENT TRANSFORMATIONS

Applications such as the one sketched in the previous section require complex multimedia composition. Multimedia composition refers to the provision of mechanisms for integrating multi-source media into one coherent multimedia presentation. This section starts with a survey [2; 5; 12; 18; 20] that summarizes the core aspects of multimedia composition. Based on this, a set of transformations is defined, described as modifications of any one of the core aspects identified. We then cluster these transformations, and classify them based on the level of complexity of behavioral specification and implementation. For this clustering, we specifically look at the temporal semantics, as this area of document transformation has seen the least previous study. The intention of this section is to provide a taxonomy of multimedia document transformations in a manner that relates application functionality to language semantics.

3.1 Aspects of Multimedia Document Formats

We can identify recurrent aspects that describe a multimedia presentation: media items, style, spatial composition, temporal composition, and user interaction. We call these the *core aspects* of multimedia document formats. Here is our breakdown of these aspects:

- *Media Items* (defining *what* to render): corresponds to each of the media assets composing the multimedia presentation. A document model must provide support for rendering a variety of media items and formats including video, images, text, and sometimes 3D objects. Some formats include the media data in their delivery file format, some will use an indirection mechanism such as URLs to reference the actual data. For selectivity purposes, the document model might also provide mechanisms for rendering one of multiple alternative assets.
- *Style* (defining *how* to render media): apart from media item inclusion in the presentation, the document model must provide styling mechanisms (e.g. font for text, image cropping parameters). Within this category we also consider other multimedia styling options and digital effects such as zooming within an image.
- *Spatial composition* (defining *where* to render media): in order to provide a meaningful and aesthetically attractive presentation, the multimedia document model must layout the media items onto an output screen. The position and size of the media items can be defined in different ways such as such as absolute or relative positioning, flow based layout or hierarchical layout. Such positioning does not need to be static across time, so multimedia document models might offer animation support: the possibility of relocating media items.
- *Temporal composition* (defining *when* to render media): the multimedia document must allow authors to when the media items should be rendered. Temporal composition includes specifying start time and duration of media items, and also synchronization constraints between the items.
- *User interaction* (defining *how to influence* the presentation): user interaction refers to the actions exposed to the user for influencing multimedia presentations. User interaction can be considered as an special case of the work presented in this paper, as it provides ways in which the multimedia document can interact with its surroundings. Rewind, forward, and content skipping are examples of how the user can affect a presentation while playing. Nevertheless, multimedia player-based interaction does not necessarily transform the multimedia document, so it will not be considered in detail in this paper.

In order to better illustrate these core aspects we can investigate and compare a few distinct multimedia document formats: QuickTime File format, HTML5+CSS, and SMIL and NCL. These three formats were selected because they are sufficiently different that together their designs cover the space of all multimedia document formats reasonably well. A comparison between the formats is shown in Table 1.

Table 1 - Comparison of Document Models

	QuickTime	HTML5+CSS	SMIL
Media Items	Embedded	By reference	By reference
Style	Limited	Flexible using CSS	Style, cropping, digital effects
Spatial	Flexible model	Flexible using CSS	Hierarchical using regions
Temporal	Playlists, parallel	Not provided. Needs scripts	Complete temporal model
Interaction	Limited	Links, events	Links, events

The *QuickTime File format* is a large and rich format usable not only for multimedia distribution but also for archiving and during authoring. For the purpose of this paper we will investigate only the features that are applicable to its use as a distribution format. QuickTime is a container format that allows significant flexibility in the pixel-based spatial and temporal assembly of embedded media items ('atoms') associated with 'tracks' which are maintained in a data structure. QuickTime file format provides rich spatial and temporal composition constructs. However, it allows only limited external interaction during playback, since the entire composition is embedded inside the QuickTime file, which must be loaded completely before execution.

HTML5 provides rich structured constructs for declarative webpage description. Unlike QuickTime file format, most media items are referenced by a URL, and not embedded in the presentation. Layout is primarily flow-based, but in addition *CSS* provides styling capabilities and spatial composition. Interaction, transformations on the media items, styling, and spatial composition are possible through scripting, by updating values in the associated DOM tree. It does not provide declarative mechanisms for temporal composition, this needs to be handled in Javascript.

SMIL and NCL inherit the benefits of HTML as a declarative and structured document format. It supports distributed multimedia presentations, where media items are refer to as URLs. It provides styling and spatial composition, and a rich temporal model [19]. The biggest difference between the two languages is that while SMIL provides high-level constructs defining a restricted set of temporal relationships, NCL allows an author to create a set of custom relationships from a toolkit of language primitives as objects.

3.2 Multimedia Document Transformations

Based on the core aspects of multimedia document models presented above, we can identify a number of *simple transformations*: potential modification of a single instance of one of those core aspects. For example, changing the position of a media item would be considered as a simple transformation, while modifying the complete layout of the presentation would be considered as an aggregation of simple transformations. As we are interested in multimedia documents as components in a larger application we should also look at the use cases for these transformations. The transformation classes include:

- *Media item change*: given a media item, the runtime environment can decide to activate it or not, or replace it by another media item. This kind of transformations are useful for selection purposes, for example by selecting the German language subtitle track, instead of the English one. Another example is to decide to start playing a media item with a given offset.
- *Style change*: given a media item, its characteristics can be modified. This kind of transformation is useful for adaptation purposes. Examples include increasing the fonts for elderly users, digital effects such as zooming and color changes or highlighting for attracting attention.
- *Spatial composition change*: given a media item, its position and size can be modified. This kind of transformation is useful for adaptation purposes, for example by changing the layout when the presentation is moved to a different device (or orientation of the same device). These transformations are also useful to change relative sizes and position, if the users mode of operation or focus of attention changes during the application run.
- *Temporal composition change*: given a media item, its start time or duration can be modified. Moreover, changes to the synchronization between media items can occur with the addition or deletion of other media items. This kind of transformation is useful for enriching a multimedia presentation while viewing [8] and for providing related media items to a viewer at viewing time [10].

3.3 Taxonomy

After identifying the simple multimedia transformations, the next step is to classify the operations based on the level of complexity. The level of complexity is inferred based on the implications imposed on the underlying rendering model. The major difference between multimedia rendering (SMIL, NCL) and a-temporal rendering (HTML) lays in the scheduler. The scheduler is in charge of constructing a time graph of the presentation, based on the duration of the media items and on the temporal synchronization between them. Based on the time graph, media items composing the presentation become active or inactive at specific moments in time. If we want to evaluate the complexity that a certain transformation imposes at rendering time, there are two key issues to be addressed: Does the modification affect the time graph? Does the modification affect the current playing item or an item to be activated in the future?

We realize that this is an over-simplification of the problem, but we do this on purpose. Modification of a-temporal aspects of multimedia documents is similar to modifying formats like HTML. Its implications for recomputing layout and flow has received a lot of study in the past, and is much better understood than modifying the temporal aspects of a document.

Table 2 provides the clustering, based on complexity of the identified multimedia transformations.

The first cluster of transformations, the *selection cluster*, correspond to selectivity mechanisms. They do not affect the temporal graph. These transformations can normally be specified at authoring time and allow the user to select among a number of

Table 2 - Taxonomy of Document Transformations

Selection cluster	Selection among predefined media items. It does not affect the time graph of the presentation.
Adaptation cluster	Modification of the style or layout composition. They do not affect the time graph of the presentation.
MediaItem cluster	Modification of the content of a media item. They might affect the time graph of the presentation.
Structural cluster	Modification of the temporal composition (add/remove item). They must recompute the temporal graph.

media items. A typical example is the selection of a language track for the subtitles. Since selection only enables or disables items that already exist in the document it does not give rise to changes in the schedule.

The second cluster of transformations, the *adaptation cluster,* correspond to adaptation transformations (how and where to render). They include things such as changing the characteristics of media items, changing the position and size of a media item, and applying effects to a media item. Some examples include layout changes and zooming. In the first case, the layout of the presentation can be adapted to different screens, while in the second case special effects can be used for aesthetic purposes. They do not affect the temporal graph.

The third cluster of transformations, the *MediaItem cluster*, corresponds to operations that change the media item. Similarly to the selection cluster, they influence what to render. But in this case, the source media item is not known at authoring time. These kind of transformations might affect the temporal graph, based on the implicit duration of the media item. The situation becomes more complex if the change is made upon an active media item.

Finally, the fourth and most complex cluster of transformations, the *structural cluster,* correspond to structural changes: modifications to the temporal composition that force the whole time graph to be recomputed. Examples include the addition and removal of media items and media containers. Live Editing [10], the Ambulant annotator [8] and the Watch-and-Comment paradigm [7] are some examples. This cluster could be further subdivided, depending on whether the affected media items are currently active, have been active in the past or will only become active in the future. This types of modifications have further implications, since they can introduce fundamental errors in terms of correctness and validity of the presentation.

QuickTime file format, HTML5+CSS, and SMIL support different types of modification. QuickTime file format can easily handle *selection* and *adaptation* transformations, but does not support the rest because the entire composition is embedded into one file. HTML5+CSS offers mechanisms for *selection* and *adaptation*, transformations, while *MediaItem* transformations can be supported using JavaScript programming. Because it lacks a temporal model, *structural cluster* transformations are not supported. Finally, as shown in section 4, languages like SMIL can support the different clusters of multimedia transformations.

4. ANALYSIS OF SMIL 3.0

In order to validate the model, we have applied it to the SMIL 3.0 language. The intention is to determine whether we can create a breakdown of all possible editing operations on a running SMIL presentation into the categories outlined in section 3.

The first problem is that SMIL is a rather big language, and the specification itself [6] contains no formal definition. In stead, a mix of English and pseudocode is used to define the semantics.

Given the fact that no complete formal model for SMIL exists, and the fact that our editing model is also not formally defined, we opted for an informal but complete analysis. To this end, we examined all possible *atomic operations* on a SMIL presentation and classified them, where atomic operations are insertion, deletion or substitution of attributes or elements. For each of these, we determined the worst-case category.

out of scope	Unknown: the semantics aren't specified in SMIL
Invalid	
trivial	Has no effect on current state
easy	Has only renderer-local effect, without timing implications
seek	Has timing implications, but no timegraph implications
recompute	Has timegraph implications in the future
recompall	Has timegraph implications in the future and past
Side effect	Has second-order implications

	Notes
1	Changing xml:base is equivalent to instantly changing all URLs in a document
2	Depends on whether accessibility is implemented in the target player
3	Semantic choice for seek: local or global timing consistency
4	depends on implementation: if times are propagated early this is recompute, if they are propagated late it is seek. Check (with
5	
6	SMIL is vague on semantics here...
8	Seems like an oversight by the SMIL designers...
9	Need to make sure this assessment is correct

	B	C	D	E	F	G	H	I	J	
1			Runtime implementation effort							
2	**Type**	**Name**	**Past-active**	**Active**	**Pre-active**	**Notes**	**Animation?**	**Category**	**Module**	**Vari**
3	attribute	abstract	out of scope	out of scope	out of scope			Level Y: out of scope	SMIL 3.0 MediaDescription Module	
4	attribute	accelerate	out of scope	out of scope	out of scope			Level Y: out of scope	SMIL 3.0 Time Manipulations	
5	attribute	accesskey	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Linking	
6	attribute	accumulate	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Animation	
7	attribute	action		Side effect				Level Z: Side effect	SMIL 3.0 State	
8	attribute	actuate	Side effect	Side effect	trivial			Level Z: Side effect	SMIL 3.0 Linking	
9	attribute	actuate	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Animation	
10	attribute	additive	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Animation	
11	attribute	allowReorder	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Animation	
12	attribute	alt	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Linking	
13	attribute	alt	trivial	easy	trivial	2		Level B1: Adaptation	SMIL 3.0 MediaAccessibility Module	
14	element	anchor	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Linking	
15	element	animate	recompall	recompall	recompute			Level D2: Timegraph	SMIL 3.0 Animation	
16	element	animateColor	recompall	recompall	recompute			Level D2: Timegraph	SMIL 3.0 Animation	
17	element	animateMotion	recompall	recompall	recompute			Level D2: Timegraph	SMIL 3.0 Animation	
18	element	animation	recompall	recompall	recompute			Level D2: Timegraph	SMIL 3.0 Media Object	
19	element	area	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Linking	
20	attribute	attributeName	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Animation	
21	attribute	attributeType	trivial	easy	trivial			Level B1: Adaptation	SMIL 3.0 Animation	
22	element	audio	recompall	recompall	recompute			Level D2: Timegraph	SMIL 3.0 Media Object	
23	attribute	author	out of scope	out of scope	out of scope			Level Y: out of scope	SMIL 3.0 MediaDescription Module	
24	attribute	autoReverse	out of scope	out of scope	out of scope			Level Y: out of scope	SMIL 3.0 Time Manipulations	

Figure 2 - Sample Table Data. Note that the full table is available for download.

4.1 Methodology

We have classified all operations on a SMIL document. The full classification contains about 300 entries, one for each SMIL element and attribute, and is too large to include here. Figure 2 gives an impression of the data, the full table is available separately[2]. In the next section we will outline a condensed version of the classification. We have only considered *valid* transformations: operations on a valid SMIL document that results in a new valid SMIL document.

Besides the transformation categories from section 3 we have identified four pseudo-classes of transformations:

- *Invalid* operations, which do not result in a valid document. Most operations on items from the SMIL Identity Module and SMIL Structure Module fall in this category. Examples are changes to the baseProfile attribute, or the <body> element.
- *Trivial* operations, which have no direct effect. Operations on declarative elements in the <head> section fall into this category.
- *Out of scope* operations. These are operations on items for which the SMIL specification leaves the semantics implementation-defined, therefore we cannot determine the effect of changes. Operations on items from the SMIL Media Description and Metainformation Modules and on the <param> element fall in this category.
- *Side effect* operations cannot be reasoned about because they may have an indirect effect. An example is removal of a <setvalue> that has already been executed. Determining the effect of this operation on the timegraph is impossible without complete dataflow analysis. Operations on items from the SMIL State Module fall in this category.

4.2 Observations

For each element and each attribute, we have determined the effect of a runtime DOM-like atomic operation on that element or attribute. The operations we have considered are addition, deletion or replacement, and we have not distinguished between the three. We have always considered the worst case effect for each operation.

If we now examine whether there is any correspondence between SMIL modules and transformation categories it turns out there is a reasonable match, with two exceptions: the SMIL Timing and Synchronization and Media Object modules, operations on items from these modules fall in all categories. As these two modules can be considered the core of SMIL this is not unexpected.

Let us start with atomic operations on elements. These fall into two categories:

- the *Adaptation cluster* is the category for operations on elements from the Linking and Layout modules, and on styling elements from the SmilText module.
- *Structural cluster* is the category for operations on elements from the Timing and Synchronization module and on timed elements from the other modules.

Finally, we have to categorize operations on about 150 attributes:

- *Selection cluster* mainly for modifying Content Control attributes and the priorityClass attributes from Timing and Synchronization.
- *Adaptation cluster* for operations on attributes from the Linking, Media Accessibility, Media Object, Animation, Transition Effects, Layout and SmilText modules. From Timing and Synchronization, the fill-related and synchronization-related attributes fall in this category.
- *MediaItem cluster* for modifying some attributes from the Media Object module (src, clipBegin, clipEnd, mediaRepeat) and some from Timing and Synchronization (repeat, repeatDur, repeatCount).
- *Structural cluster* for a few attributes from Timing and Synchronization: begin, end, dur, min, max, endsync, restart and restartDefault.

Two more interesting observations can be made from this breakdown. First, there are only very few items that fall in the *structural cluster*: elements from Media Object and Timing and Synchronization plus 8 attributes. Apparently, the whole timegraph of a SMIL presentation hinges only on these items. As the timegraph is often seen as the concept that makes SMIL hard to fathom it might be interesting to see whether things become easier to understand if this set of items was treated as its own separate module.

The second observation is that it seems to be somewhat random whether an attribute is modifiable through SMIL Animation or SMIL State attribute value templates. Most animatable attributes fall into our *Adaptation cluster*, which is to be expected. But: there are many other attributes in that category that are not animatable, such as mediaOpacity or fit, and it is unclear why the SMIL specification does not allow animating these.

5. EXAMPLE APPLICATION: SHARED GAME PLAYING

In this section we will examine the sample TA2 videoconferencing application from section 2, and show how the model from the previous sections has been applied to the design of this application.

5.1 Global Architecture

Aside from audiovisual and game connectivity, the TA2 system contains a number of components that do person detection and speaker detection, and an orchestration component that uses input from those two components and the game to decide what output to show on the screen. The items that can be shown on the main screen are:

- prerecorded media such as video clips and static images, used for static backgrounds, but also to enhance game immersion,
- one of three live HD video streams (720p) and a mirrored video self-view,
- activity output, for activities that lend themselves to being projected on the TV.

Figure 3 shows the main components and their connections while a session is ongoing:

- the *activity* (the game),
- the *Visual Composition Engine* (VCE), the videoconferencing and multimedia composition engine,
- the *Audio Composition Engine* (ACE), which digitizes, transmits and renders 3D audio,

2 http://homepages.cwi.nl/~jack/doceng-live-editing/

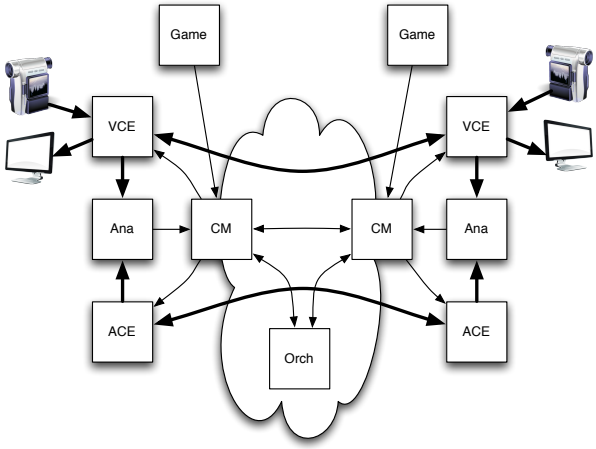

Figure 3. Interconnection architecture

- *Analysis*, which gets a video and 3D audio feed and does person identification and speaker detection,
- *Orchestration*, which gets cues from analysis and the activity and determines which video shots to present to the participants, and
- the *Connection Manager* (CM), which handles routing messages (local and remote) and overall control.

5.2 VCE Requirements

The multimedia composition component, the VCE, is responsible for the following functionality:

- Digitizing, cropping, encoding and transmitting video from the central 1080p camera to the peer VCE and to the analysis engine, possibly also displaying it on screen locally;
- Receiving and decoding live video, and showing the currently selected video feed(s) while maintaining lip-sync with the audio feed handled by the ACE;
- Showing graphics rendered by the activity;
- Showing pre-recorder media; and
- Handling the spatial and temporal composition of all these items in a way envisioned by the activity designer.

Each of these tasks can be modified by the orchestration engine, which can modify the video crop to transmit, which video feed to display and which prerecorded media to display at what time.

All this has to be done with predictable, and preferably low, latency. Moreover, because TA2 is intended for deployment in the home there is a limit on the amount of hardware we can throw at the problem.

5.3 Applying the Model

Initially, before we developed the model outlined in section 3, we considered a composition engine that would be highly dynamic, driven by the orchestration engine. This design put a significant portion of the burden of temporal composition on the orchestration engine, with the VCE only doing the composition "in the small". This design had the problem of putting the orchestration engine in the critical path with respect to timing, and would also require the application designer to split the design over multiple components.

Applying the model led to a number of observations:

- *Selection cluster* operations are required, to play optional media items in response to gameplay changes.
- *Adaptation cluster* operations are required, to implement changing the transmitted video crop,
- *Mediaitem cluster* operations are required to implement switching video feeds,
- *Structural cluster* operations are not needed.

Structural changes to the running multimedia presentation are not needed because the overall structure of the presentation can be known in advance. While we have only implemented the Space Alert game so far, we have done preliminary requirement analysis for a number of other TA2 activities, and the observations seem to hold for a large class of activities. Board games all seem to fit this usage pattern. Similar for shared media consumption: while the movie being watched may need to be changed the overall structure of the presentation will remain the same.

We have identified two future application areas where this analysis may not hold:

- If the number of participants in the activity can change an activity designer may want to dynamically alter the number of video feeds shown.
- Shared editing of media will probably require structural changes to the presentation.

Pursuing these remains as future work.

5.4 VCE Design and Implementation

To ensure that the multimedia presentation is only modified in a well-defined manner we have created a special-purpose API for the Orchestration component to use. The API also contains a call that allows the analysis engine to communicate position and identity of faces detected in the video feed to the VCE. The relevant portion of the API is depicted in table 3.

This API isolates orchestration (and the rest of the system) from multimedia composition implementation details, but also from needing any knowledge of pixel coordinates. The orchestration

Table 3 - VCE API

PlayOptionalMedia(*mediaid*)	Inserts a predefined item into the presentation
CropVideo (*viewid*)	Crop/scale transmitted video to a predefined view rectangle
AnalysisFacePosition(*personid, x, y, w, h*)	Communicates identity and coordinates of a person detected in the video stream
TrackFace(*personid*)	Crop/scale the transmitted video to track the given person
CutToCamera (*feedid*)	Start displaying the given video stream

engine is only concerned with person IDs, and the corresponding coordinates are communicated directly from analysis to the VCE. Similarly, orchestration uses high level view identities when selecting the video crop to transmit.

This separation of functionality also means that hardware parameters such as camera resolution, transmitted video size and output display size are known only to the VCE, and can be adapted to the local available hardware easily. Moreover, the visual designer of the activity need only concern itself with the VCE.

The complete VCE design is shown in figure 4. The core of the VCE is a complete SMIL playback engine. We have used the open source Ambulant SMIL player as the basis for this, extended with the special-purpose renderers we need (video grabbing, live video display, etc).

The API from table 3 has been implemented in an XMLRPC server module. The SMIL documents played back by the VCE have a prescribed layout of the SMIL State element, and the XMLRPC server implements the various methods by changing values in the state document. These values can then be propagated to the renderers (for *adaptation* and *MediaItem* cluster operations) or used as triggers for a SMIL StateChanged event for *selection cluster* operations.

The whole implementation of the API (XMLRPC server, method implementations, glue) is about 800 lines of Python code.

6. DISCUSSION

This paper represent a first step of a long journey, where the final goal is to streamline the interaction between multimedia documents and their environment. We acknowledge that our work has many limitations and it is not necessarily based on strong foundations, but it serves as a starting point. A possible contentious issue is whether it is useful to model transformations to multimedia documents if there is no underlying formal model of the document itself. Another question that might arise is how much trust one can put in the analysis of SMIL using our model. After all, there is no way to prove that the analysis is complete or even correct.

As all known formal models for SMIL are incomplete (and this is even more the case for other multimedia languages), our model is based on observation and on our previous knowledge on the topic. Undoubtedly, some of the details are incomplete and will need modifications in the future, but that should not invalidate our conclusions. We view our contribution as being a framework for future work - a first foundation brick. Furthermore, we hope that this work will open new path for researchers interested in the role of document modification in other languages.

The model presented in this paper is validated in section 4 by an exhaustive study of the SMIL 3.0 language. This validation is based on only a subset of the possible transformations that could conceivably be done: only atomic DOM operations are considered. Future work will have to consider other document transformations, such as addition or removal of subtrees, splitting of nodes, etc. Despite the limitation is scope the validation has provided some interesting insights. There seems to be a

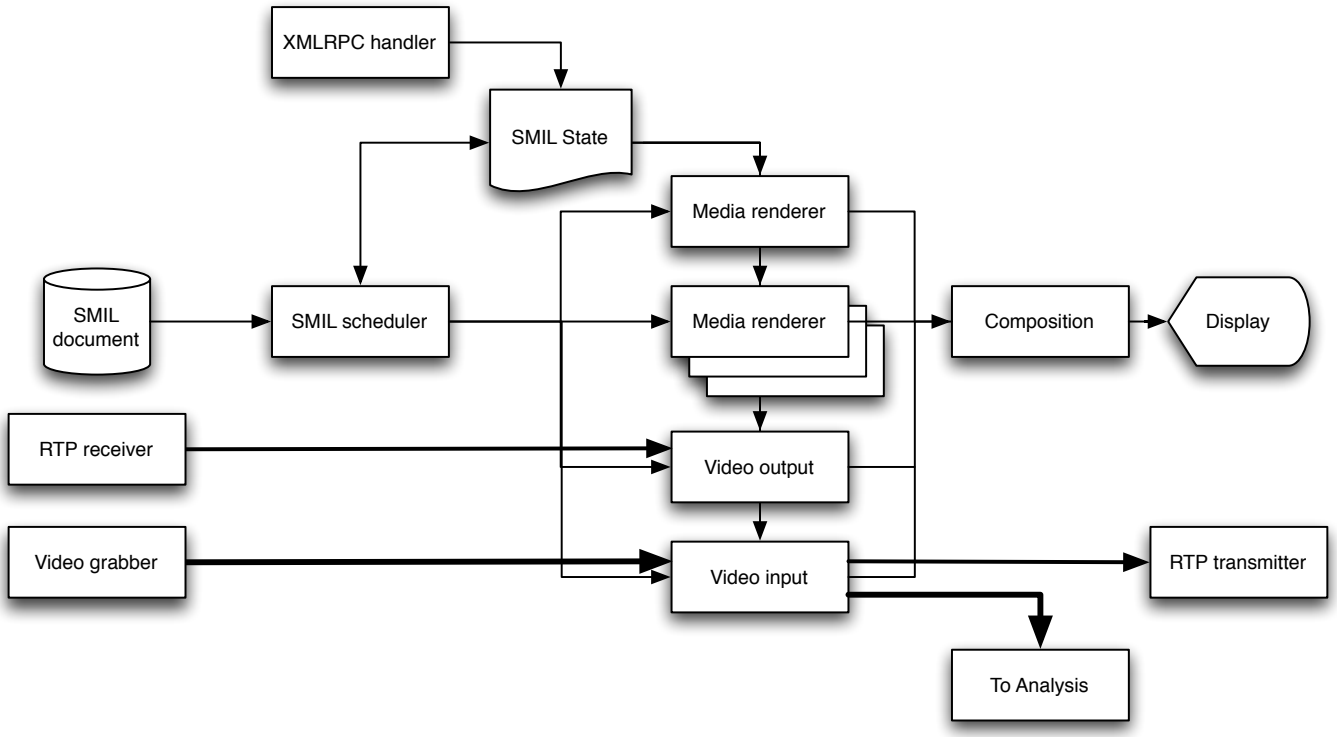

Figure 4. VCE architecture

reasonable match between the different SMIL modules and our document transformation classification, with the exception of operations on the Timing and Synchronization module and the Media Object model. This may indicate that those modules have too much functionality and would benefit from splitting into multiple modules. Another interesting observation is that very few items fall in the structural cluster. This means that the whole structural timing of SMIL is built upon a small number of elements and attributes. This is good news for future work: this paper has explicitly not considered structural modifications, because of their timegraph implications. If we want to study these modifications in the future there is only a very limited subset of SMIL that we need to consider. It may even be possible to create a full formal model for just this functionality of SMIL.

Section 5 discusses how the model has been used for designing an innovative multimedia application, as part of the TA2 project. In the design and development process, the model presented in this paper has been useful. While not a full formal model, even having something more like a *mental model* available has clearly helped here. The biggest benefit of using the model has been in the design of the APIs from/to the renderer and in the usage of SMIL State for assuring consistency and response time. Once we were aware that structural modifications were inherently more difficult to implement than other modifications we were able to adapt our global design to steer clear of these difficulties. All in all, we feel the proof of the pudding is in the eating: we have successfully applied the model to the design of an application with positive results.

In order to better understand what is our intention and where our contribution lays, let us digress for a couple of paragraphs. In the last years the Web has become a truly interactive platform. CSS has been an essential Web development for enabling Web page transformations to different devices, people, and situations. Based on templates, one can for example layout a webpage that fits mobile devices. Since the re-computation needed for text processing and reflow is significant, a number of optimization algorithms has been proposed lately [14]. The introduction of the DOM made it possible not only to transverse the contents of a Webpage, but to add, modify, and delete elements. By making Webpages modifiable, a number of new applications started to be a reality. Two current examples include fluid annotations [3] and 3D DOM integration [1]. Recently, the introduction of AJAX technology has made it more efficient to dynamically change the current view of a page [22], and it is the basis of many Google products and Flickr. In all the previous cases, the transformations - or no transformation when using CSS - on the base document affect the document view. But there are is no dependency with an scheduler, which makes things more complicated.

SMIL Animation is a good example of *simple transformations* in the temporal domain. SMIL allows for controlled modifications of core aspects in a declarative manner. The author of the document can specify the animation function and the duration of the animation for, for example, change a color in the presentation, move and resize media assets, and change the style of a text element. But a running animation does not actually change the attribute values in the DOM, it uses the so-called “sandwich model” [6] where animation-triggered modifications are layered on top of the underlying DOM values. Ideally, the animation runtime maintain a presentation value for any target attribute separated from the DOM, CSS, or other object model in which the target attribute is defined.

A-temporal Web documents have become dynamic, unfortunately we cannot say the same yes for multimedia documents. One can easily add annotations or aggregate Web contents using standardized methods and have reasonable confidence this will work, also across implementations. The same cannot be done yet for multimedia applications, where each application needs ad-hoc methods to modify the underlying document. From the previous examples we can see that there has been quite a lot of work on a-temporal document transformations and that it does exist work on temporal, but not structural, changes in the temporal domain. This paper tries to fill the gap: to study structural transformations for multimedia documents. The final intention is to propose an starting model that can be subsequently being extended by other researchers. As indicated before this is not an end point, but the beginning of an exciting journey.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have investigated the feasibility of a classification of editing operations on live multimedia documents and the applicability of such a classification.

We have created a classification, based on perceived implementation complexity of the individual operations, and applied this classification to two distinct problem areas: application design and multimedia language design and implementation. For the application design, the model has helped us rethink how the multimedia delivery component should interact with the rest of the system, resulting in a simpler and more elegant implementation. We believe this will also hold for other application designers. The analysis of SMIL using the classification has resulted in insights that should be valuable to future multimedia language designers.

In sections 3 and 5 we have already hinted at the fact that our structural modification category is rather large, and has operations of varying complexity. We plan to investigate a further classification in this area in the future.

8. ACKNOWLEDGEMENTS

The work presented in this paper was funded in part by the FP7 project TA2: Together Anywhere, Together Anytime. Work on implementing and evaluating the VCE was performed together with our partners at Alcatel Lucent Bell Labs in Antwerp and British Telecom in Ipswich. The work has benefited from its application across a number of project work areas, most notably the *Space Explorers* game discussed in this paper.

9. REFERENCES

[1] Behr, J., Eschler, P., Jung, Y., and Zöllner, M. 2009. X3DOM: a DOM-based HTML5/X3D integration model. In Proceedings of the Iinternational Conference on 3D Web Technology, pp. 127-135. DOI= http://doi.acm.org/10.1145/1559764.1559784

[2] Boll S., and Klas W. 2001. ZYX-A Multimedia Document Model for Reuse and Adaptation of Multimedia Content, IEEE Transactions On Knowledge and Data Engineering, 13 (3): 361-382.

[3] Bouvin, N. O., Zellweger, P. T., Grønbæk, K., and Mackinlay, J. D. 2002. Fluid annotations through open hypermedia: using and extending emerging web standards. In Proceedings of the International Conference on World Wide Web, pp. 160-171. DOI= http://doi.acm.org/10.1145/511446.511468

[4] Buchanan, M. C. and Zellweger, P. T. 2005. Automatic temporal layout mechanisms revisited. ACM Transactions on Multimedia Computing, Communications, and Applications, 1(1): 60-88.

[5] Bulterman, D. and Rutledge, L. 2008. Interactive Multimedia for the Web, Mobile Devices and Daisy Talking Books. Springer-Verlag, Heidelberg, Germany, ISBN: 3-540-20234-X.

[6] Bulterman, D. et al. 2008. Synchronized Multimedia Integration Language (SMIL 3.0). W3C. URL=http://www.w3.org/TR/SMIL/

[7] Cattelan, R.G., Teixeira, C., Goluarte, R., and Pimentel, M.G.C. 2008. Watch-and-Comment as a Paradigm toward Ubiquitous Interactive Video Editing. ACM Transactions on Multimedia Computing, Communications, and Applications, 4(4)

[8] Cesar, P., Bulterman, D.C.A., and Jansen, J. 2006. The Ambulant Annotator: Empowering Viewer-Side Enrichment of Multimedia Content. Proceedings of the ACM Symposium on Document Engineering, pp. 186-187.

[9] Cesar, P., et al. 2008. Multimedia adaptation in ubiquitous environments: benefits of structured multimedia documents. Proceedings of the ACM Symposium on Document Engineering, pp. 275-284.

[10] Costa, R.M.R., Moreno, M.F., Rodrigues, R.F., Soares, L.F. 2006. Live editing of hypermedia documents. Proceedings of the ACM Symposium on Document Engineering, pp. 165—172.

[11] Jansen, J. and Bulterman, D. C. 2008. Enabling adaptive time-based web applications with SMIL state. In Proceeding of the ACM Symposium on Document Engineering, pp. 18-27. DOI= http://doi.acm.org/10.1145/1410140.1410146

[12] Jourdan, M., Layaida, N., Roisin, C., Sabry-Ismaïl, L., and Tardif, L. 1998. Madeus, and authoring environment for interactive multimedia documents. Proceedings of the ACM International Conference on Multimedia, pp. 267-272.

[13] Lemlouma, T., and Layaida, N. 2003. Adapted Content Delivery for Different Contexts. 2003. Proceedings of the International Symposium of Applications and the Internet.

[14] Meyerovich, L. A. and Bodik, R. 2010. Fast and parallel webpage layout. In Proceedings of the International Conference on World Wide Web, pp. 711-720. DOI= http://doi.acm.org/10.1145/1772690.1772763

[15] Nanard, M., Nanard, J., King, P.R. and Gaillard, L. 2007. Genre Driven Multimedia Document Production by Means of Incremental Transformation. Proceedings of the ACM Symposium on Document Engineering, pp. 111-120.

[16] Ossenbruggen, J.V., Hardman, L., Geurts, J., and Rutledge, L. 2003. Towards a Multimedia Formatting Vocabulary. In Proceedings of the World Wide Web Conference, pp. 384-393.

[17] Pellan, B. and Concolato, C. 2008. Adaptation of scalable multimedia documents. In Proceeding of the ACM Symposium on Document Engineering, pp. 32-41. DOI= http://doi.acm.org/10.1145/1410140.1410148

[18] Pihkala, K., and Vuorimaa, P. 2006. Nine methods to extend SMIL for Multimedia Applications. Multimedia Tools and Applications, 28(1): 51-67.

[19] Rogge, B., and Bekaert, J., and Walle, R.V. 2004. Timing Issues in Multimedia Formats: Review of the Principles and Comparison of Existing Formats. IEEE Transactions on Multimedia, 6(6): 910 - 924.

[20] Silva, H., Rodrigues, R.F., Soares, L.F.G., and Muchaluat-Saade, D.C. 2004. NCL 2.0: integrating new concepts to XML modular languages. Proceedings of the ACM Symposium on Document Engineering, pp. 188-197.

[21] Scherp, A., and Boll, S. 2005. MM4U: A Framework for Creating Personalized Multimedia Content, Managing Multimedia Semantics, IRM Press, pp. 246-287.

[22] Thiessen, P. and Chen, C. 2007. Ajax live regions: chat as a case example. Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, pp. 7-14. DOI= http://doi.acm.org/10.1145/1243441.1243450

[23] Williams, D., Ursu, M. F., Cesar, P., Bergström, K., Kegel, I., and Meenowa, J. 2009. An emergent role for TV in social communication. *Proceedings of the European Conference on Interactive Television*,19-28. DOI= http://doi.acm.org/10.1145/1542084.154208