Face Modeling and Animation Language for MPEG-4 XMT Framework

Ali Arya, *Senior Member, IEEE*, and Steve DiPaola

***Abstract*—This paper proposes FML, an XML-based face modeling and animation language. FML provides a structured content description method for multimedia presentations based on face animation. The language can be used as direct input to compatible players, or be compiled within MPEG-4 XMT framework to create MPEG-4 presentations. The language allows parallel and sequential action description, decision-making and dynamic event-based scenarios, model configuration, and behavioral template definition. Facial actions include talking, expressions, head movements, and low-level MPEG-4 FAPs. The ShowFace and iFACE animation frameworks are also reviewed as example FML-based animation systems.**

***Index Terms*—Face animation, language, modeling, MPEG, XML, XMT.**

I. Introduction

MODERN multimedia presentations are no longer a one-piece prerecorded stream of audio-visual data but a combination of processed and/or synthesized components. These components include traditional recordings, hand-made animations, computer-generated sound and graphics, and other media objects to be created and put together based on certain spatial and temporal relations. This makes the content description one of the basic tasks in multimedia systems.

Advances in computer graphics techniques and multimedia technologies have allowed the incorporation of computer generated content in multimedia presentations and applications such as online agents and computer games. Many techniques, languages, and programming interfaces are proposed to let developers define their virtual scenes. OpenGL, Virtual Reality Modeling Language (VRML), and Synchronized Multimedia Integration Language (SMIL) are only a few examples that will be briefly reviewed in Section II. Although new streaming technologies allow real-time download/playback of audio/video data, effective content description provides major advantages, mainly

- dynamic creation of new content as opposed to transmission and playback of prerecorded material;
- more efficient use of bandwidth by transmitting only a description to be created on demand (if possible);
- increased capability for content search and retrieval.

The behavior of a virtual online agent or a character in a computer game, for example, would be considerably improved if instead of relying on limited "footage"—the animator could "describe" the desired content needed in response to possible events. Such content could then be created dynamically, provided we had a sufficiently powerful animation engine. In face animation, this means that not only we do not need to create the content off-line; we are also able to apply the same "scenario" to different virtual characters and have new characters perform previously defined actions.

In the area of facial animation, some research has been done to represent and describe certain facial actions with predefined sets of "codes." Facial Action Coding System [1] is probably the first successful attempt in this regard (although not directly a graphics and animation research). More recently, the MPEG-4 standard [2], [3] has defined Face Definition and Animation Parameters (FDP and FAP) to encode low level facial actions like jaw-down, and higher level, more complicated actions like smile. It also provides Extensible MPEG-4 Textual format (XMT) as a framework for incorporating textual descriptions in languages like SMIL and VRML. MPEG-4 face parameters define a low-level mechanism to control facial animation. Although very powerful, this mechanism lacks higher levels of abstraction, timing control and event management. In other words, MPEG-4 face parameters do not provide an animation language whereas XMT languages do not include any face-specific features, yet.

In this paper we propose a language specifically designed for face animation and modeling. Face Modeling Language (FML) is based on Extensible Markup Language (XML, http://www.w3.org/xml) which allows re-use of existing XML tools and products. FML provides a hierarchical structured content description for facial animation; from high level stories to low level face movements, giving maximum power and flexibility to content authors. It is compatible with MPEG-4 FAPs and makes them easier to use by creating a higher level of abstraction, and can be incorporated into MPEG-4 XMT framework, filling the empty space of a face animation language. FML combines the advantages of XML-based multimedia languages and MPEG-4 face parameters into a hierarchical structure dedicated to facial animation. Although the animation engine can use a variety of methods for content creation (such as 3-D models for pure synthetic data or photographs for "modified" imagery), FML provides a data type-independent mechanism for controlling the animation.

Manuscript received April 23, 2003; revised March 15, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Richard B. Reilly.

A. Arya is with the School of Information Technology, Carleton University, Ottawa, ON K1S 5B6 Canada (e-mail: arya@ carleton.ca).

S. DiPaola is with the School of Interactive Arts and Technology, Simon Fraser University, Surrey, BC, Canada (e-mail: sdipaola@sfu.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2007.902862

1520-9210/$25.00 © 2007 IEEE

The authors have developed two facial animation systems which use FML as a possible content description mechanism. *ShowFace* [4]–[6] is the older (and simpler) system that uses image transformations applied to photographs to create animation. *iFACE* [7], [8] is the newer system that isolates low-level graphics data (2-D or 3-D) from the rest of the system so it can be used for either 2-D photographs or 3-D synthesized data with limited replacement of low-level modules.

In Section II, some of the related works in multimedia content description will be briefly reviewed. The basic concepts and structure of FML, some case studies, and our facial animation systems will be discussed in Sections III–V. Although the main focus of this paper is on content description, in Section V we quickly review some approaches to content creation in order to make the study of FML and the discussed facial animation systems self-sufficient. Some conclusions will be the topic of Section VI. FML specification (including details on language constructs, elements, and their attributes) can be found online [9].

II. Related Work

The diverse set of works in multimedia content description involves methods for describing the components of a multimedia presentation and their spatial and temporal relations. Historically, the first technical achievements in this regard were related to video editing where temporal positioning of video elements is necessary. The Society of Motion Picture and Television Engineers (SMPTE) Time Coding [10], [11] that precisely specifies the location of audio/video events down to the frame level is the basis for EDL (Edit Decision List) [10], [11] that relates pieces of recorded audio/video for editing. Electronic Program Guide (EPG) is another example of content description for movies in the form of textual information added to the multimedia stream.

More recent efforts by SMPTE are focused on Metadata Dictionary that targets the definition of a metadata description of the content (see http://www.smpte-ra.org/mdd). These metadata can include items from title to subject as well as components. The concept of metadata description is the basis for other similar researches like Dublin Core (http://dublincore.org), EBU P/Meta (http://www.ebu.ch/pmc_meta.html), and TV Anytime (http://www.tv-anytime.org). Motion Picture Expert Group (MPEG) is also another major player in standards for multimedia content description and delivery. The MPEG-4 standard that came after MPEG-1 and MPEG-2, is one of the first comprehensive attempts to define the multimedia stream in terms of its forming components (objects like audio, foreground figure, and background image). Users of MPEG-4 systems can use Object Content Information (OCI) to send textual information about these objects.

A more promising approach in content description is the MPEG-7 standard [12]. MPEG-7 is mainly motivated by the need for a better, more powerful search mechanism for multimedia content over the Internet but can also be used in a variety of other applications including multimedia authoring. The standard extends OCI and consists of a set of Descriptors for multimedia features (similar to metadata in other works), Schemes that show the structure of the descriptors, and an XML-based Description/Schema Definition Language.

Most of these methods are not aimed at and customized for a certain type of multimedia stream or object. This may result in a wider range of applications but limit the capabilities for some frequently used subjects like human face. To address this issue MPEG-4 includes Face Definition Parameters (FDPs) and Face Animation Parameters (FAPs) [3], [13], [14]. FDPs define a face by giving measures for its major parts and features such as eyes, lips, and their related distances. FAPs on the other hand, encode the movements of these facial features. Together they allow a receiver system to create a face (using any graphics method) and animate that face based on low level commands in FAPs. The concept of FAP can be considered a practical extension of Facial Action Coding System (FACS) used earlier to code different movements of facial features for certain expressions and actions.

It should be noted that FAPs do not need to be used with a synthetic face and geometric models. They are independent of animation method and simply define the desired movements. They can be used to apply pre-learned image transformations (based on detected location of facial features) to a real 2-D picture in order to create a visual effect like talking, facial expression, or any facial movements [4], [5], [15].

MPEG-4 FDPs and FAPs do not provide an animation language but only a set of low-level parameters. Although they are powerful means in facial animation, the content providers and animation engines still need higher levels of abstraction on top of MPEG-4 parameters to provide group actions, timing control, event handling and similar functionality usually provided by a high-level language.

After a series of efforts to model temporal events in multimedia streams [16], important progress was made in multimedia content description with Synchronized Multimedia Integration Language (SMIL) [17], an XML-based language designed to specify temporal relationships of components in a multimedia presentation, specially in web applications. SMIL can coexist quite suitably with MPEG-4 object-based streams. SMIL-Animation is a newer language (http://www.w3.org/TR/smil-animation) based on SMIL which is aimed at describing animation pieces. It establishes a framework for general animation but neither of these two provide any specific means for facial animation. There have also been different languages in the fields of Virtual Reality and computer graphics for modeling computer-generated scenes. Examples are Virtual Reality Modeling Language (VRML, http://www.web3d.org) and programming libraries like OpenGL (http://www.opengl.org).

The MPEG-4 standard includes eXtensible MPEG-4 Textual format (XMT) framework [18] to represent scene description in a textual format providing interoperability with languages such as SMIL and VRML. It consists of two levels of textual formats. XMT-A is a low-level XML-based translation of MPEG-4 contents. XMT-Ω is a high-level abstration of MPEG-4 features, allowing developers to create the scene description in languages like SMIL and VRML. These descriptions can then be compiled to native MPEG-4 format to be played back by MPEG-4 systems. It can also be directly used by compatible players and browsers for each language, as shown in Fig. 1.

None of these languages are customized for face animation, and they do not provide any explicit support for it, either. The absence of a dedicated language for face animation, as

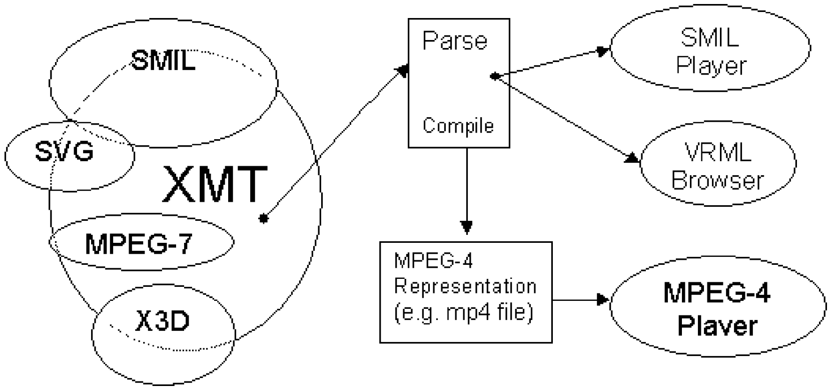

Fig. 1. Interoperability in XMT [18].

an abstraction on top of FACS AUs or MPEG-4 FAPs, has been evident especially within the XMT framework. Recent advances in developing and using Embodied Conversational Agents (ECAs), especially their web-based applications, and growing acceptance of XML as a data representation language, have drawn attention to markup languages for virtual characters [19]–[22]. The basic idea is to define specific XML tags related to agents' actions such as moving and talking. Virtual Human Markup Language (VHML) [21] is an XML-based language for the representation of different aspects of "virtual humans," i.e., avatars, such as speech production, facial and body animation, emotional representation, dialogue management, and hyper and multimedia information (http://www.vhml.org). It comprises a number of special purpose languages, such as EML (Emotion Markup Language), FAML (Facial Animation Markup Language), and BAML (Body Animation Markup Language). In VHML, timing of animation-elements in relation to each other and in relation to the realization of text is achieved via the attributes "duration" and "wait." These take a time value in seconds or milliseconds and are defined for all elements in EML and FAML, i.e., for those parts of VHML concerned with animation. A simple VHML/FAML document looks like this

```
< vhml >
< person disposition = ”angry” >
< p >
First I speak with an angry voice and look very angry,
< surprised intensity = ”50” >
but suddenly I change to look more surprised.
< /surprised >< /p >< /person >< /vhml >.
```

Multimodal Presentation Markup Language (MPML) [22] is another XML-based markup language developed to enable the description of multimodal presentation on the WWW, based on animated characters (http://www.miv.t.u-tokyo.ac.jp/MPML/en). It offers functionalities for synchronizing media presentation (reusing parts of the Synchronized Multimedia Integration Language, SMIL) and new XML elements such as < `listen` > (basic interactivity), < `test` > (decision making), < `speak` > (spoken by a TTS-system), < `move` > (to a certain point at the screen), and < `emotion` > (for standard facial expressions). MPML addresses the interactivity and decision-making not directly covered by VHML/FAML, but both suffer from a lack of explicit compatibility with MPEG-4 (XMT, FAPs, etc).

Another important group of related works are behavioral modeling languages and tools for virtual agent. BEAT [23] is an XML-based system, specifically designed for human animation purposes. It is a toolkit for automatically suggesting expressions and gestures, based on a given text to be spoken. BEAT uses a knowledge base and rule set, and provides synchronization data for facial activities, all in XML format. This enables the system to use standard XML parsing and scripting capabilities. Although BEAT is not a general content description tool, it demonstrates some of the advantages of XML-based approaches.

Other scripting and behavioral modeling languages for virtual humans are considered by other researchers as well [13], [24], [25]. These languages are usually simple macros for simplifying the animation, or new languages which are not using existing multimedia technologies. Most of the time, they are not specifically designed for face animation. Lee, *et al.* [13] have proposed the concept of a hierarchical presentation of facial animation but no comprehensive langugae for animation and modeling is proposed.

Table I summarizes major methods and languages that may be used for facial animation, and their supported features. The need for a unifying language specifically designed for facial animation that works as an abstraction layer on top of MPEG-4 parameters is the main motivation in designing FML as described in the next section.

III. Face Modeling Language

A. Design Ideas

Fig. 2 illustrates a series of facial actions. A "wink" (closing eye lid and lowering eyebrow), a "head rotation", and a "smile" (only stretching lip corners, for simplicity). To play back a multimedia presentation of this sequence, we can record a "live action", transfer it if necessary, and finally play the file. This requires

- availability of a character to record the scene;
- storage of multimedia data for each action;
- high bandwidth transfer.

As we mentioned before, a simple description like wink-yaw-smile (with some more details) can achieve the

TABLE I
CONTENT DESCRIPTION METHODS AND FACIAL ANIMATION FEATURES

	FACS	**MPEG-4**	**SMIL**	**VRML**	**FAML**	**MPML**	**BEAT**
Face-specific Parts	Yes	Yes	No	No	Yes	Yes	Yes
MPEG-4 Compatible	No	Yes	No	No	No	No	No
Timing Control	No	No	Yes	Partial	Yes	Yes	Yes
Decision-making	No	No	Yes	Partial	No	Yes	Partial
XML-based	No	No	Yes	Yes	Yes	Yes	Yes
High-level Face Components	No	Partial	No	No	Partial	Partial	Partial
Behavioural Modeling	No	No	No	No	No	No	Yes

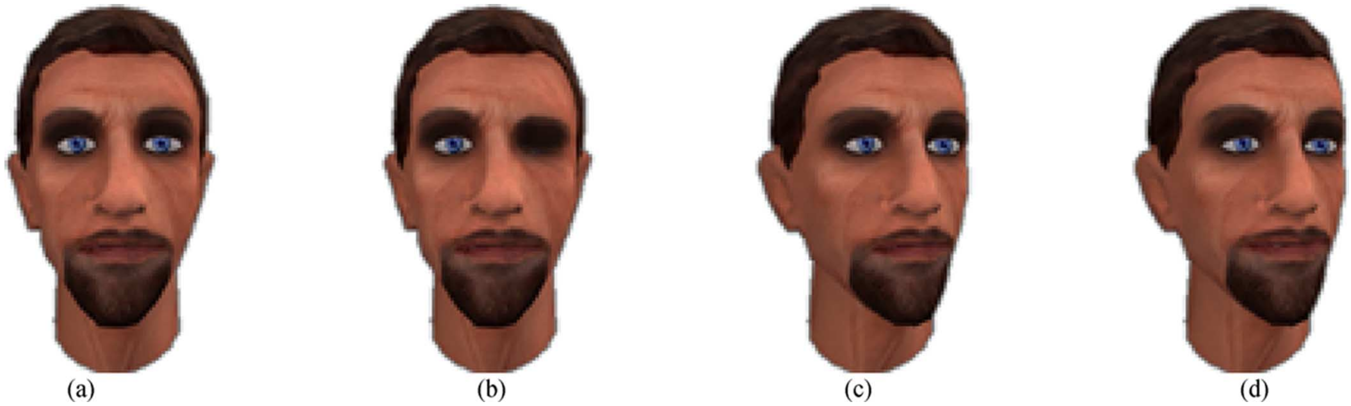

Fig. 2. Series of facial actions: (a) start, (b) wink, (c) head rotation, and (d) smile.

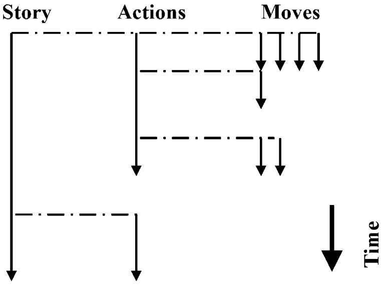

Fig. 3. FML timeline and temporal relation of face activities.

same result with no prerecorded data or massive data transfer (assuming the character is modeled and we have a sufficiently powerful graphics system). For instance, these actions can be done by following MPEG-4 FAPs, as shown in the first action of Fig. 3 (see MPEG-4 documentations [3] for a list of FAPs).

- Wink
  FAP-31 (raise-l-i-eyebrow).
  FAP-33 (raise-l-m-eyebrow).
  FAP-35 (raise-l-o-eyebrow).
  FAP-19 (close-t-l-eyelid).
- Head Rotation
  FAP-49 (head-rotation -yaw, i.e., horizontal).
- Smile
  FAP-6 (stretch-l-lipcorner).
  FAP-6 (stretch-r-lipcorner).

The first problem with this approach is lack of parameters at facial component level (e.g., one eye-wink instead of four FAPs) and proper timing mechanism. The situation will be even more complicated when we want the animation to pause after this action series, wait for some external event and then start a proper response. Grouping, time, and events are out of the scope of MPEG-4 parameters and need to be addressed by a higher-level language that abstracts on top of MPEG-4 parameters. Languages that provide timing and other animation control mechanism are not compatible with MPEG-4 or do not have face-specific features (as discussed in the previous section).

The analysis in the previous section provides us with a set of features (illustrated in the above example) that a face animation language needs to support, collectively. FML is an XML-based language that is designed to do this, filling the gap in XMT framework for a face animation language. So the main ideas behind FML are

- timeline definition of the relation between facial actions and external events;
- defining capabilities and behavior templates;
- compatibility with MPEG-4 XMT and FAPs;

- compatibility with XML and related web technologies and existing tools;
- allowing customized and facial component-level parameters;
- independence from content generation (animation) methods and data types (2-D versus 3-D).

The first action of Fig. 2 can be done by an FML script such as the following lines (elements are discussed later; details omitted):

$$
\begin{aligned}
&< \textbf{param}\ \texttt{type} = \texttt{"eye-wink"}/ >\\
&< \textbf{hdmv}\ \texttt{type} = \texttt{"yaw"}/ >\\
&< \textbf{expr}\ \texttt{type} = \texttt{"smile"}/ >.
\end{aligned}
$$

The choice of XML as the base for FML is based on its capabilities as a markup language, growing acceptance, and available system support in different platforms. FML supports a hierarchical view of face animation, i.e., representing simple individually-meaningless moves to complicated high level stories. This hierarchy consists of the following levels (bottom-up).

- Frame, a single image showing a snapshot of the face (Naturally, may not be accompanied by speech).
- Move, a set of frames usually representing transition between key frames (e.g., making a smile).
- Action, a "meaningful" combination of moves.
- Story, a stand-alone piece of face animation.

FML defines a timeline of events (Fig. 3) including head movements, speech, and facial expressions, and their combinations. Temporal combination of facial actions is done through time containers which are XML tags borrowed from SMIL (other language elements are FML-specific). Since a face animation might be used in an interactive environment, such a timeline may be altered/determined by a user. So another functionality of FML is to allow user interaction and in general event handling (notice that user input can be considered a special case of an external event.). This event handling may be in form of

- Decision Making; choosing to go through one of possible paths in the story.
- Dynamic Generation; creating a new set of actions to follow.

A major concern in designing FML is compatibility with existing standards and languages. Growing acceptance of MPEG-4 standard makes it necessary to design FML in a way it can be translated to/from a set of FAPs. Also due to similarity of concepts, it is desirable to use SMIL syntax and constructs, as much as possible. Satisfying these requirements make FML a good candidate for being a part of MPEG-4 XMT framework.

B. FML Document Structure

Fig. 4 shows the typical structure of FML documents. An FML document consists, at the higher level, of two types of elements: **model** and **story**. A **model** element is used for defining face capabilities, parameters, and initial configuration. This element groups other FML elements (model items) described in next subsection.

A **story** element, on the other hand, represents the timeline of events in face animation in terms of individual Actions (FML **act** elements). The face animation timeline consists of facial activities and their temporal relations. These activities are themselves sets of simple Moves. Sets of these moves are grouped together within Time Containers, i.e., special XML tags that define the temporal relationships of the elements inside them. FML includes three SMIL time containers **excl**, **seq** and **par** representing exclusive, sequential and parallel move-sets. Other XML tags are specifically designed for FML

```
<fml>
  <model> <!-- Model Info -->
    <model-item />
  </model>
  <story> <!— Story TimeLine -->
    <action>
      <time-container>
        <FML-move>
      </time-container>
    </action>
  </story>
</fml>
```

Fig. 4. FML document map; model-item, time-container, and FML-move represent parts to be replaced by actual FML elements.

FML supports three basic face moves: talking, expressions, and 3-D head movements. Combined through time containers, they form an FML **act** which is a logically related set of activities. Details of these moves and other FML elements and constructs will be discussed in the next subsections.

C. Modeling Elements

The **model** element encloses all the face modeling information. As described in FML specification, some important model elements are as follows.

- **character**: The personality being animated. This element has one attribute **name**.
- **img**: An image to be used for animation; This element has two major attribute **src** and **type**. It provides an image and tells the player where to use it. For instance the image can be a frontal or profile pictures used for creating a 3-D geometric model.
- **range**: Acceptable range of head movement in a specific direction; It has two major attributes: **type** and **val** specifying the direction and the related range value.
- **param**: Any player-specific parameter (e.g., MPEG-4 FDP); **param** has two attributes **name** and **val** .
- **event**: external events used in decision-making; described later.
- **template**: defines a set of parameterized activities to be recalled inside **story** using **behavior** element.

Fig. 5 shows a sample model module. FML templates will be extended in later versions to include advanced behavioral modeling.

D. Story-Related Language Constructs

The FML timeline, presented in Stories, consists primarily of Actions which are a purposeful set of Moves. The Actions are performed sequentially but may contain parallel Moves in themselves. Time Containers are FML elements that represent the temporal relation between moves. The basic Time Containers

```
  <model>
  <img src="me.jpg" type="front" />
    <range type="left" val="60" />
    <template name="hi" >
      <seq begin="0">
        <talk>Hello</talk>
      <hdmv type=”0” begin="0"
         end=”3s” val="30" />
      </seq>
    </template>
</model>
<story>
    <behavior name="hi" />
  </story>
```

Fig. 5. FML model and templates.

are **seq** and **par** corresponding to sequential and parallel activities. The former contains moves that start one after another and the latter contains moves that begin at the same time. The Time Containers include primitive moves and also other Time Containers in a nested way. The **repeat** attribute of the Time Container elements allows iteration in FML documents.

Similar to SMIL, FML also has a third type of Time Containers, **excl**, used for implementing exclusive activities and decision-making as discussed later.

FML supports three types of primitive moves plus MPEG-4 FAPs and some other move elements.

- **talk** is a nonempty XML element and its content is the text to be spoken if a Text-To-Speech module is available. It can also act as an empty element where attribute **file** specifies speech data.
- **expr** specifies facial expressions with attributes **type** and **val**. Every FML-compatible animation system has to support standard MPEG-4 expression types, i.e., smile, anger, surprise, sadness, fear, disgust, and normal. Other expression types can be used depending on the underlying animation system.
- **hdmv** handles 3-D head rotations. Similar to **expr**, this move is an empty element and has the same attributes.
- **fap** inserts an MPEG-4 FAP into the document. It is also an empty element with attributes **type** and **val**.
- **param** inserts a custom parameter depending on the underlying animation system. It is also an empty element with attributes **type** and **val**.
- **wait** pauses the animation. It is also an empty element with only timing attributes.
- Other controls (Play, Capture, Save, etc.).

All story elements have four timing attributes **repeat**, **begin**, **duration**, and **end**. In a sequential time container, begin is relative to start time of the previous move, and in a parallel container it is relative to the start time of the container. In case of a conflict, duration of moves is set according to their own settings rather than the container. The repeat attribute is considered for defining definite (when having an explicit value) or indefinite loops (associated with events). Fig. 6 illustrates the use of time containers and primitive moves. It should be noted that the current version of FML does not include any mechanism for enforcing the timing (such as audio-video synchronization). It is the responsibility of underlying animation system to handle these issues.

```
<action>
  <seq begin=”0”>
    <talk>Hello</talk>
    <hdmv end=”5s” type=”0”
      val=”30” />
  </seq>
  <par begin=”0”>
    <talk>Hello</talk>
    <expr end=”3s” type=”3”
      val=”50” />
  </par>
</action>
```

Fig. 6. FML time containers and primitive moves.

```
<!-- in model part -->
  <event name=”user” val=”-1” />
  <!-- in story part -->
  <excl ev_name=”user”>
    <talk ev_val=”0”>Hello</talk>
    <talk ev_val=”1”>Bye</talk>
  </excl>
```

Fig. 7. FML decision making and event handling.

E. Event Handling and Decision Making

Dynamic interactive applications require the FML document to be able to make decisions, i.e., to follow different paths based on certain events. To accomplish this **excl** time container and **event** element are added. An event represents any external data, e.g., the value of a user selection. The new time container associates with an event and allows waiting until the event has one of the given values, then it continues with exclusive execution of the action corresponding to that value, as illustrated in Fig. 7.

The FML Processor exposes proper interface function to allow event values to be set in run time. **event** is the FML counterpart of familiar if-else constructs in normal programming languages.

F. Compatibility

The XML-based nature of this language allows the FML documents to be embedded in web pages. Normal XML parsers can extract data and use them as input to an FML-enabled player, through simple scripting. Such a script can also use XML Document Object Model (DOM) to modify the FML document, e.g., adding certain activities based on user input. This compatibility with web browsing environments, gives another level of interactivity and dynamic operation to FML-based system, as illustrated in Section IV.

Another major aspect of FML is its compatibility with MPEG-4 XMT framework and face definition/animation parameters. This has been achieved by using XML as the base for FML and also sharing language concepts with SMIL. As the result, FML fits properly within the XMT framework. FML documents can work as an XMT-Ω code and be compiled to MPEG-4 native features, i.e., FDPs and FAPs. FML is a high level abstraction on top of MPEG-4 Face Animation Parameters. FAPs are grouped into visemes, expressions, and low-level facial movements. In FML, visemes are handled implicitly through the **talk** element. The FML processor translates the input text to a set of phonemes and visemes compatible with those defined in MPEG-4 standard. A typical Text-To-Speech

```
<fml>
  <model>
    <img src=”a.jpg”type=”frontal”/>
  </model>
  <story>
    <act>
    <seq repeat=”2”>
      <talk begin=”0”>
        Hello</talk>
      <expr begin=”0” end=”2s”
        type=”smile” val=”80” />
      <expr begin=”0” end=”1s”
        type=”normal” />
    </seq>
    </act>
  </story>
</fml>
```

Fig. 8. Static repeated FML document.

engine can find the phonemes and using the tables defined in MPEG-4 standard these phonemes will be mapped to visemes. FML facial expressions are defined in direct correspondence to those in MPEG FAPs. For other face animation parameters, the **fap** element is used. This element works like other FML moves, and its **type** and **val** attribute are compatible with FAP numbers and values. As a result, FML processor can (and will) translate an FML document to a set of MPEG-4 compatible movements (and also 3-D head rotations) to be animated by the player components.

IV. Case Studies

FML can be used in a variety of ways and applications. It can be used as a high-level authoring tool within XMT framework to create MPEG-4 streams (after translation), or be used directly by compatible players for static or interactive face animation scenarios. Here we discuss three sample cases to illustrate the use of FML documents.

A. Static Document

The first case is a simple FML document that does not need any user interaction. There is only one unique path the animation follows. The interesting point in this basic example is the use of iterations, using **repeat** attribute.

An example of this case can be animating the image of a person who is not available for real recording. The **img** element specifies the frontal (base) view of the character and the **story** is a simple one: saying hello then smiling (Fig. 8). To add a limited dynamic behavior, the image, text to be spoken, and the iteration count can be set by an interactive user and then a simple program (e.g., a script on a web page) can create the FML document. This document will then be sent to an FML-compatible player to generate and show the animation.

B. Event Handling

The second case shows how to define an external event, wait for a change in its value, and perform certain activities based on that value (i.e., event handling and decision making). An external event corresponding to an interactive user selection is defined first. It is initialized to -1 that specifies an invalid value. Then, an **excl** time container, including required activities for possible user selections, is associated with the event. The **excl** element will wait for a valid value of the event. This is equivalent to a pause in face animation until a user selection is done.

```
< ... >
<action repeat=”quit”>
  <excl ev_name=”select”>
    <seq ev_val=”0”>
      <talk>Text One</talk>
      <expr type=”smile”
        val=”100” end=”2s” />
    </seq>
    <seq ev_val=”1”>
      <talk> Text Two</talk>
      <expr type=”smile”
        val=”100” end=”1s” />
    </seq>
  </excl>
</action>
```

Fig. 9. Events and decision making in FML.

A good example of this case can be a virtual agent answering users' questions online. Depending on the selected question (assuming a fixed set of questions), a value is set for the external event and the agent speaks the corresponding answer. The FML document in Fig. 9 uses two events: one governs the indefinite loop to process the user inputs, and the second selects the proper action (replying to user question in the mentioned example).

The FML-compatible player reads the input, initializes the animation (by showing the character in initial state) and when it reaches this action, waits for user input because the **select** event does not match any of the values inside **excl**. After the event is set through the proper API (see Section V), the related action is performed. This will continue until the **quit** event, used by **repeat**, is set to a nonnegative value. If the value is zero, it stops, otherwise continues for the defined number of times.

C. Dynamic Content Generation

The last FML case to be presented illustrates the use of XML Document Object Model (DOM) to dynamically modify the FML document and generate new animation activities. For instance, a new choice of action can be added to the previous example, on-the-fly. Fig. 10 shows a sample JavaScript code that accesses an XML document, finds a particular node, and adds a new child to it. Since this case uses standard XML DOM features, we do not discuss that in more details. It only shows how the use of XML as the base language can be helpful in FML documents.

The same DOM methods can be accessed from within the FML Player to modify the document while/before playing it. The FML Player can expose proper interface functions to provide these capabilities to users/applications in an easier way.

V. Animation Systems

A. Background

Video streaming systems [26] usually do not have specific facilities for face animation. Traditionally, animating human faces is done through synthetic computer generated images with 3-D head models [13], [27], [28]. Although powerful in creating a variety of facial states, these methods require relatively complex and time-consuming computation and the output may not be very realistic. Another approach to facial animation uses image

```
function onAdd()
{
  //fmldoc is FML (XML) document
  //loaded at startup
  //get the root (fml) element
  var fml =
    fmldoc.documentElement;
  //find the proper element by
  var fmlnode;
  . . . //details not shown
  //create/add a new element
  var new =
  fmldoc.createElement(“hdmv”);
  new.setAttribute(“type”,”0”);
  new.setAttribute(“val”,”30”);
  fmlnode.appendChild(new);
}
```

Fig. 10. JavaScript code for FML document modification.

processing techniques applied to 2-D data (normal photographs) [5], [15], [29], [30]. This is usually a simple morphing between a library of key frames which means animation requires a large database of photographs [15], but the computational complexity is lower and the results seem more realistic.

The authors have been involved in two face animation research projects that used FML as a content description method. Arya and Hamidzadeh [4]–[6] developed *ShowFace* as a simple streaming system that could create facial animation streams. *ShowFace* uses a 2-D method but reduces the database requirement by learning image transformations from a set of training images. More recently, Arya and DiPaola [7], [8] have proposed *Interactive Face Animation – Comprehensive Environment (iFACE)* as a more general approach to facial animation, based on the concept of Face Multimedia Object [7], discussed later. *iFACE* uses a hierarchical geometry where data type (2-D or 3-D) is considered only at the lowest level. This is combined with a behavioral model consisting of Knowledge, Personality, and Mood. *iFACE* can be used as a "face engine" for a variety of face-based applications. Both *iFACE* and *ShowFace* systems are MPEG-4 compatible.

B. Face Multimedia Object (FMO)

For a large group of applications, facial presentations can be considered a means of communication. A "communicative face" relies and focuses on those aspects of facial actions and features that help to effectively communicate a message. We believe that the communicative behavior of a face can be considered to be determined by the following groups (spaces) of parameters (Fig. 11).

- *Geometry*: This forms the underlying physical appearance of the face. Creating and animating different faces and face-types are done by manipulating the geometry that can be defined using 2-D and/or 3-D data (i.e., pixels and vertices).
- *Knowledge*: Behavioral rules, stimulus-response association, and required actions are encapsulated into Knowledge. In the simplest case, this can be the sequence of actions that a face animation character has to follow. In more complicated cases, knowledge can be all the behavioral rules that an interactive character learns and uses.

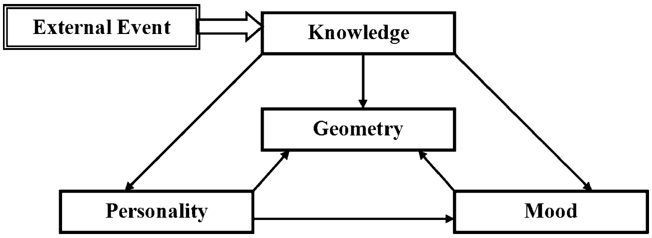

Fig. 11. Parameter spaces for face multimedia object.

- *Personality*: Different characters can learn and have the same knowledge, but their actions, and the way they are performed, can still be different depending on individual interests, priorities, and characteristics. Personality encapsulates all the long-term modes of behavior and characteristics of an individual. Facial personality is parameterized based on typical head movements, blinking, raising eye-brows and similar facial actions.
- *Mood*: Certain individual characteristics are transient results of external events and physical situation and needs. These emotions (e.g., happiness and sadness) and sensations (e.g., fatigue) may not last for a long time, but will have considerable effect on the behavior.

Face Multimedia Object (FMO) is a high-level multimedia data type encapsulating all the functionality of a communicative face. It exposes proper interfaces to be used by client objects and applications, and can be used as a "face engine" for face-based applications such as computer games and online agents. *iFACE* system provides an implementation of FMO in addition to other required tools.

C. iFACE System

iFACE [7] is a parameterized face animation system, i.e., animation is defined and created through activation of groups of parameters interacting with each other as illustrated in Fig. 11. Geometry is the foundation of facial animation. *iFACE* uses a hierarchical model that provides different layers of abstraction (such as Features, Components) on top of head data, each with their own interfaces exposing functionality related to that layer. The system will translate higher-level functions and commands to lower-level ones, and eventually to point-level manipulation. This means that animators and programmers do not need to be involved in details unless they want to override the default behavior of the system.

Knowledge, Personality, and Mood are designed as components around the Geometry, exposing their own interfaces for access by the application programs. Knowledge receives the input script and external events, and holds the rules of interaction. All of these are applied to Geometry in the form of parameters at the appropriate layer of abstraction. Personality (which can be configured through input scripts or interactively) suggests facial gestures and states based on the explicit actions requested by Knowledge. For example, if the script requires a piece of speech, Knowledge translates this to a set of phonemes and visemes and their timing, so the Geometry can animate the face. Meanwhile, Personality suggests certain head movements, facial gestures, visemes and expressions that are attributes of the chosen character's personality, based on the content of speech and energy level. Mood applies a base facial state to all the facial actions.

iFACE includes an off-line design environment, *iFaceStudio*, for creating animations and configuring the head objects, and a wrapper control, *FacePlayer*, that can be easily used in web forms and similar GUI applications. A normal scripting language can control the animation by accessing the object methods and properties. More about *iFACE* and sample animations and FML scripts can be found online [31].

D. ShowFace System

ShowFace [4]–[6] is a simple modular system for streaming facial animation. It includes components for receiving input stream (MPEG-4 descriptions or FML scripts), parsing them in order to provide a list of video and audio actions, objects for generating the multimedia content, and finally a mixer that creates the final output stream. Ituses Feature-based Image Transformation (FIX) [5]. In a training phase, a set of image-based transformations is learned by the system, which can map between these face states (e.g., transform a neutral face to a smiling one). Transformations are found by tracking facial features when the model is performing the related transitions, and then they are applied to a given image, in order to find the new location of feature points and lines. The facial regions are then mapped according to their closest feature points. FIX allows the ShowFace system to perform a 2-D version of MPEG-4 FAP operations.

VI. Conclusions

XML-based Face Modeling Language (FML) is proposed to describe the desired sequence of actions in facial animation. FML allows event handling, and also sequential or simultaneous combination of supported face states, and can be converted to a set of MPEG-4 Face Animation Parameters. It uses a similar structure to SMIL which makes the language a good candidate for being part of MPEG-4 XMT framework.

The main contributions of FML are its hierarchical structure, animation configuration and modeling, flexibility for static and dynamic scenarios, and dedication to face animation. FML fully supports MPEG-4 FAPs using high-level constructs which can be translated to FAPS and also direct FAP embedding. Compatibility with MPEG-4 XMT and use of XML as a base are also among the important features in the language. Future extensions to FML can include more complicated behavior modeling and better coupling with MPEG-4 streams.

FML covers a wide range of applications including but not limited to authoring tools for MPEG-4 streams, input to standalone animation programs, and animation components for web pages. Such applications can be used in video conferencing, games, visual effects, and online services with animated agents. FML can be used in a compatible player or in combination with MPEG-4 presentations. *ShowFace* and *iFACE* are briefly discussed, as examples of FML-compatible facial animation systems. The FML-based *iFACE* system has been successfully used by animators to create animation pieces as illustrated on the web site [31]. It provides the animators with the ease of use and full control at the same time, and allows for authoring control mixed with programming control which is highly useful in high-end games and other interactive applications.

References

[1] P. Ekman and W. V. Friesen, *Facial Action Coding System*. Mountain View , CA: Consulting Psychologists Press, 1978.

[2] S. Battista, F. Casalino, and C. Lande, "MPEG-4: A multimedia standard for the third millennium," *IEEE Multimedia*, vol. 6, 7, no. 4, 1, pt. 1 and 2, pp. 74–83, Oct.–Jan. 1999–2000.

[3] I. S. Pandzic and R. Forchheimer, Eds., *MPEG-4 Facial Animation: The Standard, Implementation and Applications* New York: Wiley, 2002.

[4] A. Arya and B. Hamidzadeh, "Personalized face animation in ShowFace system," *Int. J. Image Graph., Special Issue on Virtual Reality and Virtual Environments*, vol. 3, no. 2, pp. 345–363, 2003.

[5] A. Arya and B. Hamidzadeh, "FIX: Feature-based image transformations for face animation," in *Proc. IEEE Conf IT in Research and Technology (ITRE)*, Newark, NJ, Aug. 12–14, 2003.

[6] A. Arya and B. Hamidzadeh, "ShowFace MPEG-4 compatible face animation framework," in *IASTED Int. Conf Computer Graphics and Imaging (CGIM)*, Aug. 12–14, 2002.

[7] A. Arya, S. DiPaola, L. Jefferies, and J. T. Enns, "Socially communicative characters for interactive applications," in *Proc. 14th Int. Conf. Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG-2006)*, Plzen, Czech Republic, Jan.–Feb. 2006.

[8] S. DiPaola and A. Arya, "Socially expressive communication agents: A face-centric approach," in *Proc. Eur. Conf. Electronic Imaging and the Visual Arts, EVA-2005*, Florence, Italy, Mar. 17–18, 2005.

[9] [Online]. Available: http://img.csit.carleton.ca/iface/fml.html

[10] J. Ankeney, "Non-linear editing comes of age," *TV Technol.*, May 1995.

[11] T. D. C. Little, "Time-based media representation and delivery," in *Multimedia Systems*, J. F. K. Buford, Ed. New York: ACM Press, 1994.

[12] F. Nack and A. T. Lindsay, "Everything you wanted to know about MPEG-7," *IEEE Multimedia*, vol. 6, no. 3, pp. 65–77, Jul. 1999.

[13] W. S. Lee, M. Escher, G. Sannier, and N. Magnenat-Thalmann, "MPEG-4 compatible faces from orthogonal photos," in *Proc. IEEE Conf Computer Animation*, 1999.

[14] I. S. Pandzic, "A web-based MPEG-4 facial animation system," in *Proc. Int. Conf. Augmented Virtual Reality and 3D Imaging*, 2001.

[15] T. Ezzat and T. Poggio, "MikeTalk: A talking facial display based on morphing visemes," in *Proc. IEEE Conf. Computer Animation*, 1998.

[16] N. Hirzalla, B. Falchuk, and A. Karmouch, "A temporal model for interactive multimedia scenarios," *IEEE Multimedia*, vol. 2, no. 3, pp. 24–31, 1995.

[17] D. Bulterman, "SMIL-2," *IEEE Multimedia*, vol. 8, no. 4, pp. 82–88, Oct. 2001.

[18] M. Kim, S. Wood, and L. T. Cheok, "Extensible MPEG-4 textual format (XMT)," in *Proc. ACM Conf Multimedia*, 2000.

[19] Y. Arafa, K. Kamyab, E. Mamdani, S. Kshirsagar, N. Magnenat-Thalmann, A. Guye-Vuillème, and D. Thalmann, "Two approaches to scripting character animation," in *Proc. 1st Int. Conf. Autonomous Agents and Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, Jul. 2002.

[20] B. DeCarolis, M. Bilvi, and C. Pelachaud, "APML, a markup language for believable behaviour generation," in *Proc. 1st Int. Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, Jul. 2002.

[21] A. Marriott and J. Stallo, "VHML: Uncertainties and problems. A discussion," in *Proc. 1st Int. Conf Autonomous Agents & Multi-Agent Systems, Workshop on Embodied Conversational Agents*, Bologna, Italy, Jul. 2002.

[22] H. Prendinger, S. Descamps, and M. Ishizuka, "Scripting affective communication with life-like characters in web-based interaction systems," *Appl. Artif. Intell.*, vol. 16, no. 7–8, 2002.

[23] J. Cassell, H. H. Vilhjálmsson, and T. Bickmore, "BEAT: The behavior expression animation toolkit," in *Proc. ACM SIGGRAPH*, 2001.

[24] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive modeling: Knowledge, reasoning, and planning for intelligent characters," in *Proc. ACM SIGGRAPH*, 1999.

[25] M. Kallmann and D. Thalmann, "A behavioral interface to simulate agent-object interactions in real time," in *Proc. IEEE Conf Computer Animation*, 1999.

[26] G. Lawton, "Video streaming," *IEEE Comput.*, vol. 33, no. 7, pp. 120–122, Jul. 2000.

[27] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proc. ACM SIGGRAPH*, 1999.

[28] F. I. Parke and K. Waters*, Computer Facial Animation*. New York: A. K. Peters, 2000.

[29] C. Bregler, M. Covell, and M. Slaney, "Video rewrite: Driving visual speech with audio," in *ACM Comput. Graph.*, 1997.

[30] H. P. Graf, E. Cosatto, and T. Ezzat, "Face analysis for the synthesis of photo-realistic talking heads," in *Proc. IEEE Conf Automatic Face and Gesture Recognition*, 2000.

[31] [Online]. Available: http://img.csit.carleton.ca/iface

Ali Arya (S'95–M'98–SM'06) received the B.Sc. degree in electrical engineering from Tehran Polytechnic, Tehran, Iran, in 1990, and the Ph.D. degree in computer engineering from the Department of Electrical and Computer Engineering, University of British Columbia, Canada, in 2004.

He has worked as Research Engineer, System Analyst, and Project Manager in different research centers and leading companies, including Tehran Cybernetic Arm Project, Iran, and Honeywell, Canada, and also as Instructor and Postdoctoral Fellow at the University of British Columbia and Simon Fraser University, Vancouver, BC, Canada. Since August 2006, he has been an Assistant Professor at the School of Information Technology, Carleton University, Ottawa, ON, Canada. His research interests include social user interfaces, interactive multimedia systems, computer graphics and animation, real-time systems, and Web-based applications.

Steve DiPaola received the B.Sc. degree in computer science from the State University of New York at Stony Brook in 1981 and the M.A. degree in computer graphics from New York Institute of Technology, New York, in 1991.

He is currently an Associate Professor with the School of Interactive Arts and Technology, Simon Fraser University, Surrey, BC, Canada, which actively combines technology, science, and the arts using online and collaborative methods. There, he directs of the iVizLab (http://ivizlab.sfu.ca) which conducts research on "socially-based interactive visualization." He has research interests in human centered design, computer graphics and animation, and interactive systems. He had published (papers and book chapters) extensively in the area of character and avatar based 3-D virtual communication technologies and has given presentations worldwide. He has been a Teacher and Researcher at such institutions as Stanford University, Stanford, CA, and the Computer Graphics Lab, New York Institute of Technology.