Two approaches to Scripting Character Animation

Yasmine Arafa, Kaveh Kamyab
Ebrahim Mamdani
IIS
Imperial College
London SW7 2BT, UK
{y.arafa,k.kamyab,e.mamdani}@ic.ac.uk

Sumedha Kshirsagar
Nadia Magnenat-Thalmann
MIRALab, CUI
24 rue du General Dufour
CH 1211 Geneva, Switzerland
{sumedha,thalmann}@miralab.unige.ch

Anthony Guye-Vuillème
Daniel Thalmann
LIG
EPFL
1015 Lausanne, Switzerland
{aguye,thalmann}@lig.di.epfl.ch

ABSTRACT

Lifelike animated agents present a challenging ongoing agenda for research. Such agent metaphors will only be widely applicable to on-line applications when there is a standardised way to map underlying engines with the visual presentation of the agents. This paper delineates functions and specifications of two mark-up languages for scripting the animation of virtual characters. These languages are: Character Mark-up Language (CML) which is an XML-based embodied agent character attribute definition and animation scripting language designed to aid in the rapid incorporation of lifelike agents into online applications or virtual reality worlds. CML is constructed based jointly on motion and multi-modal capabilities of virtual human figures. The other is Avatar Mark-up Language (AML) which is also an XML-based multi-modal scripting language designed to be easily understandable by human animators and easily generated by a software process such as an agent. We illustrate the constructs of the language and describe real-time execution architectures for using these languages in online applications.

Keywords

Embodied Agents, Mark-up Languages, AML, CML, Lifelike Characters, Animated Expression.

1. INTRODUCTION

The merits of embodied agents for different applications are a matter of current debate. Commercial attempts so far have been disappointing. Microsoft, for example, has been eagerly involved in efforts to drive multimodal interfaces with embodied representations, by developing the Microsoft Active Agent tool [MSAgent, 1998]. Other commercial companies like Semantics have now caught on and are developing similar tools [Semantics]. However, while these tools are a relatively easy way to add some simple agent behaviour to web sites and applications and have proven to be useful for rapid prototyping, they remain very limiting to developers who require adding believability to the agents being used and reflect negatively on the merits of multi-modal interfaces on the whole.

The reasons for these negative evaluations lie both in that the animations are too rigid and that there is no provision for believability properties. They have been evaluated to cause user stress or irritation rather than user support or assistance, mainly because they only exhibit repeated mimics of identical behaviour that is not fully context aware. It is to avoid these pit-falls that context-sensitive affective behaviour is introduced to the embodied agent metaphor.

On the other hand current mechanisms for building cohesive believable characters under various theories are of even more importance. Efforts such as those by Elliott [Elliot, 1997], Velazquez [Velazquez, 1998], Reilly [Reilly, 1994], Sloman [Sloman, 1987], and others aid in the definition and planning of appropriate agent believable behaviour, but they do not aim at creating a "fourth generation language" for building them. Emotion and personality models for agents will only be widely applicable when there is an organised way to map design models to the effected representations. Current research at IIS aims at providing the tools for a 4G language to draw together the properties needed to generate embodied agents.

Although, there are several commercial and proprietary agent animation tools available as well as, several emotion engines or mechanisms that are available to generate and govern believable behaviour of animated agents, there is no common

mechanism or API to link the underlying engines with the animated representations. The Character Mark-up Language (CML) and Avatar Mark-up Language (AML) are developed to bridge the gap between the available underlying engines and agent animation tools.

To set the scene, the paper first sets forth the motivations for the mark-up languages developed at the Intelligent and Interactive Systems section, Imperial College London; describes the key features and capabilities they offer; and discusses the technical issues they raise based on our design and development experience on the ESPRIT project EP28831 MAPPA[1], IST project IST-1999-10192 SoNG[2] and IST project IST-1999-11683 SAFIRA[3]. The paper further sets forth the key functionality that such description and scripting languages will need to succeed in animated agent interaction applications. Finally, the implemented architecture is briefly described.

2. DYNAMIC SCRIPTING OF EXPRESSIVE BEHAVIOUR

In recent years a number of attempts have been made to specify and develop mechanisms for the dynamic scripting of expressive behaviour in embodied characters. Early high-level control mechanisms for character animation include Improv [Perlin, & Goldberg, 1996], a rule based mechanism allowing animators to define how synthetic actors communicate and make decisions. Although Improv was successful in many ways, it's use as a scripting mechanism was specific to the underlying implementation. However, Improv made the first realisation that a scripting mechanism was necessary that allowed authors of animations to define elements of a character's expressed behaviour.

As parallel developments several animation tools and many emotion and behaviour models and their respective implementations have been produced. The appearance of these has highlighted the need for powerful yet generic scripting languages to bridge the gap between behaviour generation and animation tools. Notably Virtual Human Mark-up Language (VHML) [VHML] and Human Mark-up Language (HumanML) [HumanML] are examples of these.

As a number of such scripting languages now exist, there appears to be the need for the research community to look at and agree upon the requirements of and expectations from them.

3. SCRIPTING WITH MARKUP LANGUAGES

3.1 Character Mark-up Language

Animated lifelike character representations of agents will only be widely deployed when there are real-time mechanisms to contextually map character and affect models to effected animated personifications. To approach this the SAFIRA project utilises the design elements of an architecture for including personality and emotional responsiveness in an interface agent; semantic abstraction and annotation of the knowledge being manipulated; and mapping resulting semantic understanding onto appropriate behaviour; which is translated into context-sensitive character traits by varying the planned response using variable emotion indicators and represented in the selected modality(ies).

Currently, there are several commercial and proprietary agent animation tools available (e.g. MS Agent, Jack, etc.), as well as, numerous emotion engines or mechanisms (Affective Reasoner, S&P, etc.) that are available to generate and govern believable behaviour of animated agents. However, there is no common mechanism or API to link the underlying engines with the animated representations until recently.

CML is developed with an aim to bridge the gap between the underlying Affect and process engines, and agent animation tools. CML provides a map between these tools using objects by automating the movement of information from XML Schema definitions into appropriate relational parameters required to generate the intended animated behaviour. This would allow developers to use CML as a glue-like mechanism to tie the various visual and underlying behaviour generation tools together seamlessly, regardless of the platform that they run on and the language they are developed with.

The term Character is used to denote a language that encapsulates the attributes necessary for believable behaviour. The intention is to provision for characters that are lifelike but are not necessarily human-like. Currently the attributes specified are mainly concerned with visual expression, although

[1] MAPPA: Multimedia Access through Persistent Personal Agents
[2] SoNG: portalS of the Next Generation
[3] SAFIRA Supporting Affective Interactions in Real-time Applications

there is a limited set of specification for speech. These attributes include specifications for animated face and body expression and behaviour, personality, role, emotion, and gestures.

3.1.1 Visual Behaviour Definition

Classification of behaviour is governed by the actions an agent needs to perform in a session to achieve given tasks, and is influenced by the agent's personality and current mental state. A third factor that governs character behaviour is the role the agent is given. A profile of both an agent's personality and its role are used to represent the ongoing influences to an agent's behaviour. These profiles are user-specified and are defined using XML annotation. The behaviours are defined as XML tags, which essentially group and annotate sets of action points generally required by the intended behavioural action. The CML processor will interpret these high-level behaviour tags, map them to the appropriate action point parameters and generate an animation script.

The Character Mark-up Language defines the syntactic, semantic and pragmatic character presentation attributes using structured text. CML is based on the definition XML Schema structures. The character mark-up-based language extends the descriptions for facial expressions used in the FACS system. FACS (Facial Action Coding System) defines a set of all the facial movements performed by a human face [Ekman & Rosenberg 1997]. Although FACS is not an SGML-based language in nature, we use their notion of Action Units to manipulate expressions. Character gesture attribute definitions are based on the research and observations by McNeill [McNeill 1992] on human gestures and what they reveal.

Affective expression is achieved by varying the extent and degree values of the low-level parameters to produce the required expression. The CML encoder will provide the high-level script to be used in order to specify the temporal variation of these facial expressions. This script will facilitate designing a variety of time-varying facial expressions using the basic expressions provided by the database.

3.1.2 Classification of Motion

The conceptual architecture upon which the classification of motion is based on is loosely derived from that defined by Blumberg and Russel's research [Blumberg & Russel 1999]. Blumberg and Russel's architecture uses a three-layer structure which includes: *geometry*, *motor* and *behaviour* system.

We assume a motor generation module which is responsible for the basic movements along with correlated transitional movements that may occur between them. Personified animation scripts are generated by blending specification of different poses and gestures. The base motions are further classified by generic controls that are independent of the character itself. For example a generic *move* motion can have different representations which are determined by the character personality attributes defined to represent *walk, skip, leap, run, etc*. Additionally, it is possible that behaviour can be expressed through and affects different parts of the character body, for example the intensity and degree of *arm* movements are varied by an emotion while a *move* is being performed. To realise different part of a body which are to be affected while performing a movement CML divides the character element specifications of into three units: Head, Middle part and Lower part. CML then provide specification of the constructs of each unit with varying granularity.

Action composition script is generated by a CML processor (delineated in *Figure 3.1.1*) which blends actions specified with an input emotion signal to select the appropriate gestures and achieve the expressive behaviour. CML also provisions for the generation of compound animation script by facilitating the definition and parameterisation of sequences of base movements.

The chosen base set of movements allows basic character control (movement and interactions) as well as assures the capability to perform unlimited character specific animations. The interactions can involve other characters and objects that must be referenced by a valid id within the Graphics Engine.

Base Motions

The initial set of the CML base motions are classified by the goal of the motion into: Movement, Pointing, Grasping, Gaze and Gesture as follows:

Movement defines motions that require the rotation or movement of a character from one position to another. Positions are defined by exact coordinates, an object position or a character position. Movement elements are either *move-to or turn-to.*

Pointing defines a pointing gesture towards a coordinate, object or character. Pointing elements are *point-to.*

Grasping defines motions that require the character to hold, throw or come in contact with an object or another character. Grasping elements are *grasp*, *throw* and *touch.*

Gaze defines the movements related to the head and eyes. Gaze elements are *gaze*, *track, blink, look-to* and *look-at.* The *gaze* and *track* elements requires that only the eyes be moved or track an object or character. *look-to* and *look-at* require the movement of both head and eyes.

Gesture includes motions that represent known gestures like hand movements to convey an acknowledgement, a wave, etc. Gesture elements are *gesture* and *gesture-at*

Following is an extract of the CML base movement specifications. It shows a *move-to* motion. The gesture and behaviour in which the movement is made is inherited from the state of emotion, gesture and behaviours specified. Further details on the granularity and specification of CML may be found in public SAFIRA deliverable D-SAFIRA-WP2 D2.2v2 [André *et al*. 2002].

```
<move-to>
  <order {0 to n/before/after} />
  <priority 0 to n />
  <speed {0.n to n.n(unit)/default/slow/fast} />
  <target {x,y,z/object/character} />
  <begin {ss:mmm/before/after} />
  <end {ss:mmm/before/after} />
  <repeat> {0 to n/dur} />
</move-to>
```

Sample CML Base Motion Syntax

Classification of emotion, and behaviour are

3.1.3 CML Specification

CML defines a script like that used for a play. It describes the actions and sequence of actions that will take place in a presentation system. The script is a collection of commands that tell the objects in the world what to do and how to perform actions. The language is used to create and manipulate objects that are held in memory and referenced by unique output-ontology objects. The structure of the language begins with a command keyword, which is usually followed by one or more arguments and tags. An argument to a command usually qualifies a command, i.e. specifies what form of action the command is to take, while a tag is used to denote the position of other necessary information. A character expression mark-up module will add emotion-based mark-up resulting from emotional behaviour generation rules to the CML descriptions.

Animated character behaviour is expressed through the interpretation of XML Schema structures. These structure definitions are stored in a Schema Document Type Definition (DTD) file using XSDL (XML Schema Definition Language). At run-time character behaviour is generated by specifying XML tag/text streams which are then interpreted by the rendering system based on the rules defined in the definition file. Its objective is to achieve a consistent convention for controlling character animation models using a standard scripting language that can be used in online applications.

The language contains low-level tags defining specific character gesture representations defining movements, intensities and explicit expressions. There are also high-level tags that can define commonly used combinations of these low-level tags.

Synchronisation between the audio and visual modalities is achieved through the use of SMIL (Synchronized Multimedia Integration Language) specification [SMIL]. SMIL defines an XML-based language that allows authors to write interactive multimedia presentations. Basically, CML uses the SMIL <par> and <seq> tags to specify the temporal behaviour of the modalities being presented. The <seq> tag to define the order, start time and duration of execution of a sequence, whereas the <par> tag is used to specify the elements be played in parallel. For further flexibility, CML also provides order and time synchronisation attributes the motion and audio elements defined.

```
<cml>
 <character name="n1" personality="p1"
          role="r1" gender="M"
          disposition="d1"
          transition_Dstate="t1">
  <happy intensity="i1" decay="dc1"
          target="o1" priority="pr1">
    <move-to order="o1" priority="pr2"
           speed="s" object="obj1" begin="s1"
           end="s4"/>
    <point-to order="2" priority="pr3"
           object="obj1" begin="s2" end="s4"/>
    <utterance priority="pr2" begin="s2">
        UtteranceText
    </utterance>
  </happy>
  ……
 </character>
```

Sample CML script

3.1.4 Generating Script

Script generation through to the effected animation process components consist of a set of MPEG-4-

compliant facial and body models; high level XML-based descriptions of compound facial and body features; XML-based descriptions of user-specified personality models; behaviour definitions, a CML processor, and finally a CML decoder. The general function of this component is delineated in Figure 3.1.1.

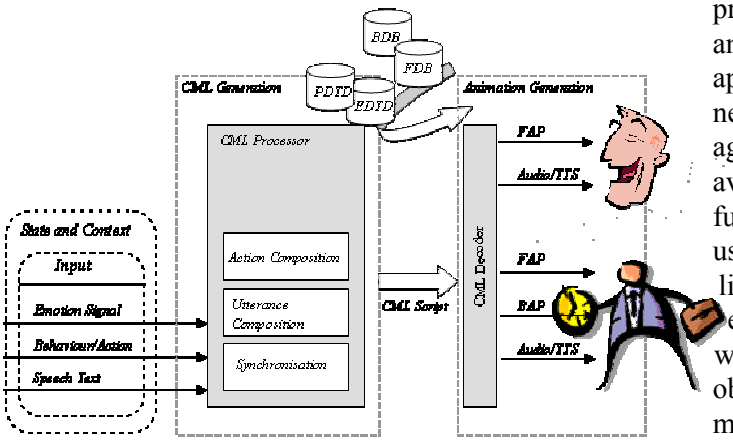

Figure 3.1.1 **Script Generation Function Abstract**

On the same basis as the AML architecture, described in the following section, the architecture of an implementation generating and using CML is divided into three conceptual components of the supporting models and database for face and body animation, CML scripting and an animation rendering tool.

The script generation component assumes state and contextual input resulting for the underlying affective processing, planning, and domain knowledge-base engines. Based on these inputs and a defined character personality, the CML processor then generates the consequent synchronised behavioural action and utterance CML script. The script is then passed onto the CML decoder which parses the CML and maps its elements onto view-specific commands for final animation rendering.

3.2 Avatar Mark-up Language

The Avatar Mark-up Language (AML) was developed in the context of the IST project SoNG in collaboration between IIS, Miralab and LIG. The objective of the project was to design and develop a full end-to-end MPEG 4 multimedia framework to support, amongst other features, 3D avatar based multi-user chat rooms and autonomous synthetic characters. The first of these was to be facilitated via the development of an interface tool that allowed users to define animation sequences by selecting and merging predefined and proprietary animation units. Likewise, synthetic characters were to be controlled in a similar manner to fill roles such as sales assistants in virtual shops.

The SoNG design philosophy was to concentrate on providing the tools and infrastructure necessary to anybody who would like to develop such applications. Hence, a common mechanism was needed to allow both human users and autonomous agent based systems to define full face and body avatar animation. However, it was important to allow future users or developers to animate their avatars using non-procedural commands whilst trying not to limit their creativity by imposing predefined facial expressions or gestures on them. Also, the focus was on providing a means of generating externally observable behaviour and not on specifying a mapping between internal reasoning and behaviour, as may be the case with synthetic characters.

The design of such a mechanism saw the animation process conceptually divided into three components (see section 3.2.2). Firstly, a database of basic facial and body animation units, which could be extended or modified by a third party interested in generating avatar animations. Secondly, a rendering system capable of merging multiple face and body animation units and text to speech input in real-time. Finally, a high-level scripting language designed to allow animators – both human and non - to specify which animations to use together with timing, priority and decay information. The resulting scripting language - AML – is the only one of the three components that we specify.

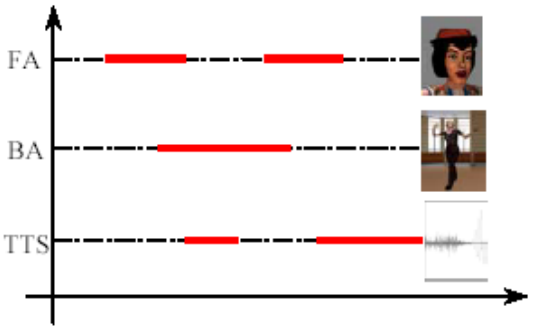

Figure 3.2.1 **Animation Scenario**

To illustrate the functionality of AML, an example of its usage may be helpful. Imagine when a shop assistant welcomes a virtual customer. It may simultaneously smile, wave, and say, “May I help

you?". In order to facilitate such multimodality in its interaction, the agent is required to trigger the appropriate face animation, body animation and TTS modules in a time-synchronized and easy manner. This may involve mixing of multiple gestures and expressions into a single animation, as shown in Figure 3.2.1.

Although no basic animations were specified within AML, it became obvious that some parameterized behaviours would have to be provided. Examples of such behaviours include pointing, facing and walking. Each of the behaviours is generated by the implemented rendering system by calculating the movement of the avatar as a function of its initial position and the target coordinates supplied by the animator.

3.2.1 AML Specification

Having given a brief overview of the requirements and purpose of AML we will now have a look at the syntax of the language. AML was developed as an XML based scripting language as it is understandable by human animators, and it is easily generated by a software process such as an agent. Figures 3.2.2 to 3.2.4 give an outline of the AML syntax. We will not describe the syntax in too much detail here. Interested readers are advised to refer to [Kshirsagar et al. 2002].

```
<AML face id="x" body id="y" root path= "p"
name = "name of animation">
 <FA start_time="t1" input_file= "f1">
   <TTS mode = "m" start_time = "t3"
   output_fap = "f3" output_wav = "f4">
     <Text>TextToBeSpoken<\Text>
   <\TTS>
   <AFML>…<\AFML>
 <\FA>
 <BA start_time = "t2" input_file = "f2">
   <ABML>…<\ABML>
 <\BA>
<\AML>
```

Figure 3.2.2 **AML Structure**

Each individual AML script is encapsulated by the AML root node and consists of either a Facial Animation (FA) node or a Body Animation (BA) node or both. FA nodes may contain a combination of TTS nodes and Avatar Face Markup Language (AFML) nodes, the syntax of which is illustrated in Figure 3.2.3. Here we highlight the flexibility that is given to an animator to define as many Expression Tracks as required, each containing as many Expressions as required. Expressions are stored in a database of facial animation units. A start time and an envelope specifying decay, duration and intensity accompany each one. In addition, Speech Tracks may be specified when a TTS engine is not available or suitable.

```
<AFML>
  <Settings>
    <Fps>FramesPerSecond</Fps>
    <Duration>mm:ss:mmm</Duration>
    <FAPDBPath>"path for expression (.ex)
    files"</FAPDBPath>
    <SpeechPath>"path for speech
     animation (.vis) files"</SpeechPath>
  </Settings>
  <ExpressionsFiles>
    <File>"expression file name" </File>
  </ExpressionsFiles>
  <ExpressionsTrack name="Track name">
    <Expression>
      <StartTime>mm:ss:mmm</StartTime>
      <ExName>"file name"</ExName>
      <Envelope>
        <Point>
          <Shape>{log,exp,linear}</Shape>
          <Duration>InSeconds</Duration>
          <Int>NormalizedIntensity</Int>
        </Point>
      </Envelope>
    </Expression>
  </ExpressionsTrack>
  <SpeechTrack name="name_of_track">
    <StartTime>mm:ss:mmm</StartTime>
    <FileName>"viseme or fap file name"
    </FileName>
    <AudioFile>"FileName"</AudioFile>
  </SpeechTrack>
</AFML>
```

Figure 3.2.3 **AFML Syntax**

Similarly, BA nodes contain Avatar Body Markup Language (ABML) nodes. The syntax can be seen in Figure 3.2.4. Here we draw your attention to a set of parameterized behaviour nodes, namely FacingAction, PointingAction, WalkingAction and ResettingAction. Each behaviour node specifies a start time, speed and priority. A subset also specifies target coordinates for the behaviour. For walking a number of control points can be specified to define the route taken by an avatar in 3D space. In addition, an arbitrary number of PredefinedAnimation nodes can be specified drawing from a database of body animation units. An intensity node is provided for predefined animations.

AML scripts offer a number of advantages to animators – human or non. Firstly they give explicit control over the mutual synchronization of facial expressions, gestures and speech by allowing start times and durations to be specified for each. This means animators are free to have even partial overlap of animation tracks starting before, together with or after any other track. Secondly, AML is independent of the basic animation units defining facial expressions and gestures. This allows animators the freedom to be creative by modifying or extending the

```
<ABML>
  <Settings>…<Settings>
  <BodyAnimationTrack name="Track name">
    <Mask>BAP indices = 0/1</Mask>
    <PredefinedAnimation>
      <StartTime>
        {mm:ss:mmm/autosynch/autoafter}
      </Start Time>
      <FileName>"filename.bap"</FileName>
      <Speed>{normal/slow/fast}</Speed>
      <Intensity>0 to n</Intensity>
      <Priority>0 to n</Priority>
    </PredefinedAnimation>
    <FacingAction>
      <StartTime>
        {mm:ss:mmm/autosynch/autoafter}
      </StartTime>
      <XCoor>target's X coordinate
      </XCoor>
      <YCoor>target's Y coordinate
      </YCoor>
      <ZCoor>target's Z coordinate
      </ZCoor>
      <Speed>{normal/slow/fast}</Speed>
      <Priority>0 to n</Priority>
    </FacingAction>
    <PointingAction>
      <StartTime>
        {mm:ss:mmm/autosynch/autoafter}
      </StartTime>
      <XCoor>target's X coordinate
      </XCoor>
      <YCoor>target's Y coordinate
      </YCoor>
      <ZCoor>target's Z coordinate
      </ZCoor>
      <Speed>{normal/slow/fast}</Speed>
      <Priority>0 to n</Priority>
    </PointingAction>
    <WalkingAction>
      <StartTime>
        {mm:ss:mmm/autosynch/autoafter}
      </StartTime>
      <ControlPoint>
      <XCoor> target's X coordinate
      </XCoor>
      <ZCoor> target's Z coordinate
      </ZCoor>
      </ControlPoint>
      <Speed>{normal/slow/fast}</Speed>
      <Priority>0 to n</Priority>
    </WalkingAction>
    <ResettingAction>
      <StartTime>
        {mm:ss:mmm/autosynch/autoafter}
      </StartTime>
      <Speed>{normal/slow/fast}</Speed>
      <Priority>0 to n</Priority>
    </ResettingAction>
  </BodyAnimationTrack>
</ABML>
```

Figure 3.2.4 **ABML Syntax**

animations used by AML. This feature also makes AML independent of the underlying implementation making it suitable for the animation of any character-based system.

3.2.2 AML Architecture

Although only the syntax of AML is specified, a reference implementation of an AML rendering mechanism – the AML Processor – has been developed in SoNG. Figure 3.2.5 gives an overview of the AML architecture. As mentioned previously, the AML architecture is divided into three conceptual components: databases of face and body animation units, a rendering mechanism and the AML script itself. The dotted arrows between the face and body databases indicate that the AML script makes reference to their content. The databases can be easily extended to contain any basic animation definitions. Expressions such as smile and surprise as well as head movements and body gestures can be stored. Indeed variations of a single animation may be stored to reflect, for example, changes in mood and personality.

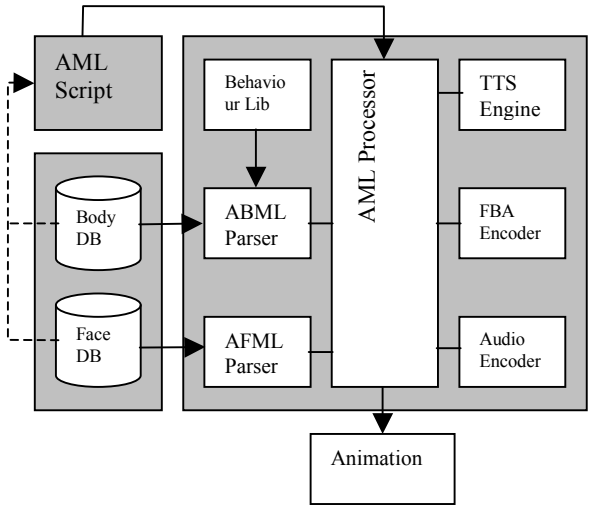

Figure 3.2.5 **AML Architecture**

Once a script has been generated, it can be passed to the AML Processor where appropriate sub-modules are called in order to decipher the script and generate animations. Notice that a behaviour library has been implemented in order to deal with parameterised behaviours such as pointing and walking.

As can be seen from Figure 3.2.5, the emphasis in the AML implementation is on the generation of animation in a seamless and synchronized manner. No support is provided for the explicit representation of emotions or personality. This made the implementation of the AML Processor relatively simple. However, it does not limit the type and variety of animations that can be generated.

4. SUMMARY AND CONCLUSIONS

The paper presents an account of two approaches to specifying scripting languages for character animation which are currently being developed and evaluated at Imperial College London. Each approach evolved through the context of the projects

they were developed within. CML took a top-down approach by defining high-level attributes for character personality, emotion and behaviour that are integrated to form the specification of synchronised animation script. New or unspecified behaviours are formed by blending together base elements and attributes thereby providing animators with the flexibility to generate animation script as required. Where as AML took a bottom-up approach in which the language provides a generic mechanism for the selection and synchronised merging of animations. In addition, AML provides the flexibility for animators (human or non) to define higher-level specifications based on the key elements provided plus any others that may be defined. The generic nature of AML implies that any software implementation supporting it will be fairly simple.

Forthcoming evaluation of each approach will further corroborate their merits. Our experience on the projects involved presented us with a possibility of a need to merge the specifications of both languages in a two-phase sequence for scripting character animation. Recent emergence of mark-up languages serving similar purposes have also exposed a profound need to agree on taxonomy for the affective and motion elements used, the granularity level of such definitions, what taxonomy for the mark-up tag definitions should be adopted, are current languages sufficient for the requirements of believable affective behaviour animation delivery?, whether Mpeg 4 FAPs and BAPs are sufficient or too granulated to provide the taxonomy required for real-time animation?, what affective and personality theories should be adopted to define the tags for affective expression?, what granularity of affective description is required?, as well as many more open issues.

5. ACKNOWLEDGMENTS

The requirement for and specification of CML was originally a contribution of the European ESPRIT project EP28831 MAPPA. CML as described in this paper is developed within the European IST project IST-1999-11683 SAFIRA and AML is developed in the European IST project IST-1999-10192 SoNG.

6. REFERENCES

[André et al. 2002] André E., Arafa Y., Botelho L., Figueiredo P., Gebhard P., Höök K., Paiva A., Petta P., Ramos P., Sengers P., and Vala M., D-SAFIRA-WP2-D2.2v2 Framework Specification for Developing Affective Agents. SAFIRA Project IST-1999-11683 (2002)

[Blumberg & Russel 1999] Blumberg B. and Russell K.: Behavior-Friendly Graphics, 1999

[Ekman & Rosenberg 1997] Ekman, P. and Rosenberg, E. L..What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System. Oxford University Press, 1997.

[Elliot, 1997] Elliott, C. Using the Affective Reasoner to Support Social Simulation, IJCAI'93, p194-200

[HumanML] -http://www.humanmarkup.org/work/humanmlSchema.zip

[Kshirsagar et al. 2002] Kshirsagar, S., Guye-Vuilleme, A., Kamyab, K., Magnenat-Thalmann, N., Thalmann, D. and Mamdani, E., "Avatar Markup Language", to appear in S. Müller and W. Stürzlinger (Editors), Proceedings of the Eigth Eurographics Workshop on Virtual Environments 2002

[McNeill, 1992] McNeill, D. Hands and Mind. The University of Chicago Press, 1992.

[MSAgent, 1998] Microsoft Corporation: Microsoft Agent Programming Interface Overview, 1998. http://www.microsoft.com

[Perlin, & Goldberg, 1996] K. Perlin, and A. Goldberg, "Improv: A System for Scripting Interactive Characters in Virtual Worlds", Proceedings of SIGGRAPH 96, ACM Press, pp. 205-216.

[Reilly, 1994] Reilly, W.S., & Bates, J. Emotions as part of a broad Agent Architecture, Working notes of the Workshop on Architectures Underlying Motivation and Emotion, 1993.

[Sloman, 1994] Sloman, A. "Motives, mechanisms and emotions", Cognition and Emotion, 1:217-234, 1987.

[SMIL] http://www.w3.org/AudioVideo/

[Semantics] http://www.semantics.com

[Velazquez, 1998] Velásquez, J. A Computational Framework for Emotion-based Control. In proc. of SAB'98 Workshop on Grounding Emotions in Adaptive Systems, 1998.

[VHML] http://www.vhml.org/downloads/VHML