MetaFace and VHML: A First Implementation of the Virtual Human Markup Language

Simon Beard, Donald Reid
Curtin University of Technology, Perth, Western Australia
{beardsw, donald }@cs.curtin.edu.au

Abstract

VHML is an evolving standard that provides an XML based language to control the onscreen presence of virtual humans. The MetaFace framework is a combination of many technologies designed to bring anthropomorphic (human-like) interaction to websites. This paper discusses the implementation of VHML within the MetaFace framework as well as evaluating the standard and suggesting solutions to any problems encountered.

1.0 Introduction

This paper discusses the issues arising from the implementation of the MetaFace framework as a testing platform for VHML (Virtual Human Markup Language) [1]. MetaFace [2] is a talking head interface for websites and will address the need for a well-defined Internet metaphor. Figure 1 shows a snapshot of one particular implementation of the MetaFace framework in action: a two-dimensional animated talking head embedded in a Perth tourism website. A user can interact with the animated face by typing natural language. The virtual face is cast in a supportive and helpful role, and is able to respond to the user with facial animation (based on the MPEG-4 standard [3]) and synthesized speech. MetaFace is essentially a website host, and as such, is dependent on a large and ever-changing collection of website data.

VHML is a text markup language that can be used to control the overall action of a virtual character. The purpose of VHML is to facilitate the realistic and natural interaction of a Talking Head/Virtual Human [4] with a user. VHML is XML/XSL [5] based, and has six major sub languages important to the functionality of Virtual Humans. The MetaFace system is only concerned with the facial and voice aspects of VHML (version 0.4), so this paper will focus on the following sub languages of VHML: FAML (Facial Animation Markup Language), SML (Speech Markup Language), EML (Emotion Markup Language), and GML (Gesture Markup Language).

Section 2.0 discusses the MetaFace framework and the role of VHML in the implementation. Many issues arose from this first implementation of VHML and are presented in this paper. Section 3.0 discusses the semantic issues, section 4.0 discusses the functional issues, and section 5.0 discusses the implementation-based issues. Following this discussion will be proposed future work and evaluation.

2.0 Role of VHML in MetaFace Framework Implementation

Before discussing this first implementation of VHML it is important to understand how VHML is integrated with the MetaFace framework.

The starting point for user interaction in this particular application occurs through a text box (below the animated character, figure 1) in which the user can enter natural language. Interaction can be in the form of a question, response to a computer posed question, or even a command. This user interaction stimulus is then sent to the server whereby the artificial intelligence subsystem interprets the stimulus, using word graphs [6], and produces appropriate output in the VHML format. The VHML document is parsed, on the server, to extract animation information for expressions, gestures, and facial manipulation. At this stage, any embedded data (HTML files in this case) can be extracted, ready to be sent to the client. Plain text from the VHML document is then extracted and translated to the particular speech markup format used by the Text to Speech TTS [7] (TTS) engine. The text is synthesised into phoneme information (phonemes, durations, pitch etc.) using an Natural Language Processor (NLP), Festival for this application [8]. The phoneme information can then be used to construct MPEG-4 visemes (the visual equivalent of phonemes). The personality module then constructs MPEG-4 Facial Animation Parameters (FAP's) from the gesture tags, emotional tags, facial animation tags and visemes. The phoneme information, FAP's, and embedded data/metadata can now be sent to the client. Phoneme information is sent to the client instead of a compressed waveform because it is less bandwidth intensive [9]. The client machine's TTS Digital Signal Processor (DSP) (Mbrola in this application [10]) synthesises the phoneme information to produce a speech waveform. The embedded data is

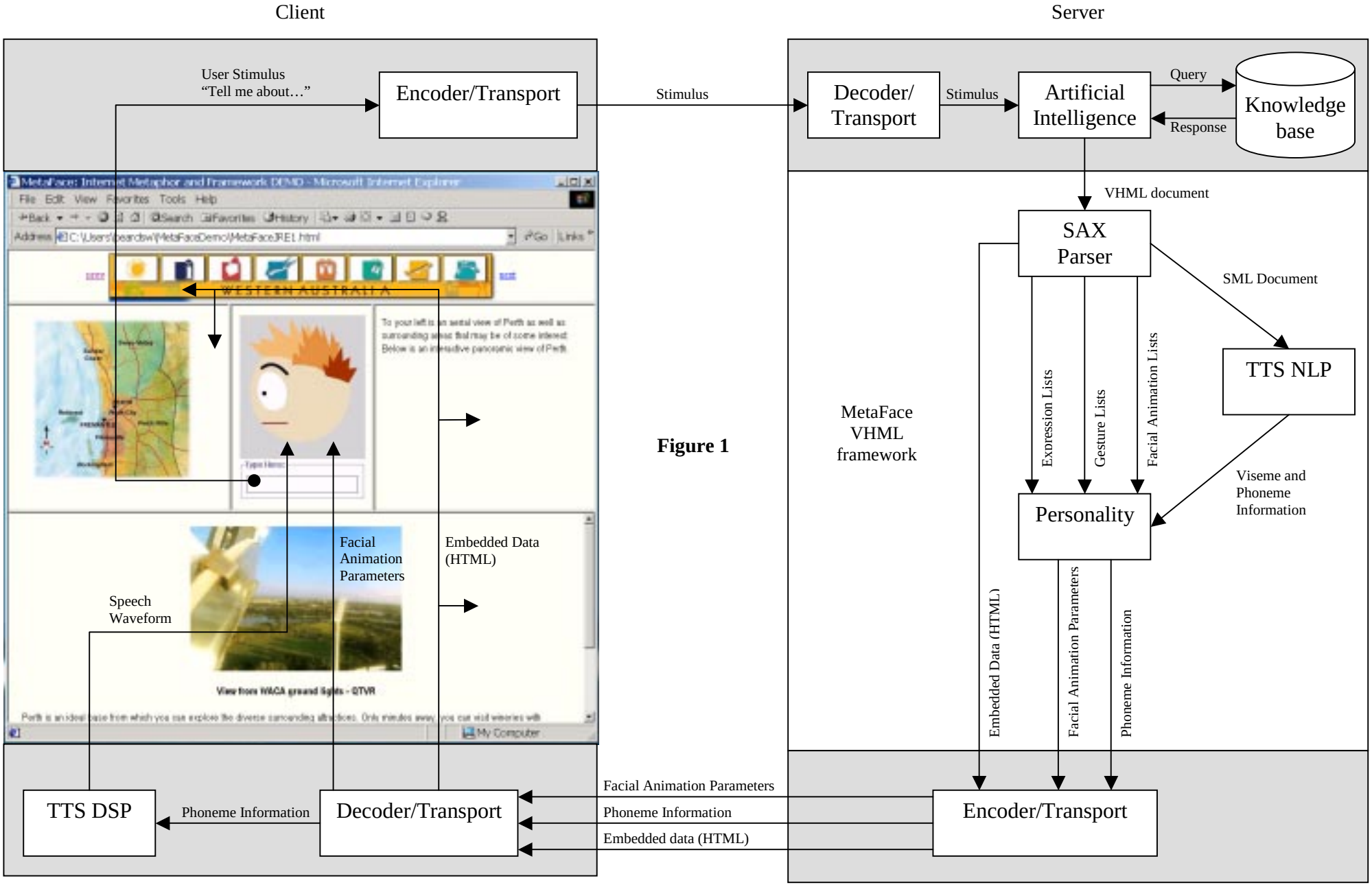
Client
Server
User Stimulus
"Tell me about…"
Encoder/Transport
Stimulus
Decoder/
Transport
Stimulus
Artificial
Intelligence
Query
Response
Knowledge
base
VHML document
SAX
Parser
SML Document
TTS NLP
MetaFace
VHML
framework
Expression Lists
Gesture Lists
Facial Animation Lists
Viseme and
Phoneme
Information
Personality
Embedded Data (HTML)
Facial Animation Parameters
Phoneme Information
Facial
Animation
Parameters
Embedded Data
(HTML)
Speech
Waveform
TTS DSP
Phoneme Information
Decoder/Transport
Facial Animation Parameters
Phoneme Information
Embedded data (HTML)
Encoder/Transport

Figure 1

displayed in the web browser, and FAP's are synchronized with the speech during playback, producing character animation and multimedia browser content in response to the user's stimulus. The user's anthropomorphic experience thus originates from a VHML document.

The Virtual Human Markup Language is being developed independently of the MetaFace framework, and to date Curtin University has been involved in the development and evaluation of three dimensional, realistic talking heads. The MetaFace application is viewed as a good testing platform for VHML because it was not developed in conjunction with any markup language, and uses two-dimensional cartoon animation.

With any standardisation process, there are bound to be discrepancies in the way that people interpret the not yet finalised content of the standards document. The following section discusses those semantic issues found when implementing VHML for the first time.

3.0 Semantic issues

An initial problem encountered was the role of "*wait*", "*duration*" and "*repeat*" attributes of the Emotion Markup Language (EML), Gesture Markup Language (GML) and Facial Animation Markup Language (FAML) sub languages and their application to character animation.

<disagree wait="1000ms" intensity="75" duration="2s" repeat="2"/> I do not agree with your insulting ideas.

Example 1

Example 1 demonstrates the use of "*wait*", "*duration*" and "*repeat*". There are many possible ways in which the facial animation can be interpreted. The "*wait*" and "*duration*" attributes could manipulate the character such that the head is shaken once during the one hundred millisecond "*wait*" period and again during the two second "*duration*" period. Alternatively one shake of the head could occur within the 3000 millisecond time period, and of course there are further ways to interpret the language. This problem is compounded when considering the "*repeat*" attribute. "*Repeat*" could mean that the head is shaken twice within the "*wait*" and "*duration*" specified (making the head shakes faster), or interpreted such that the "*wait*" and "*duration*" period must be repeated. The semantics of these attributes needed to be resolved. After careful consideration and discussion of this issue it was decided that the following semantic definition of the "*wait*", "*duration*", and "*repeat*" attributes would be used by the MetaFace system and recommended for the VHML specification.

- "*Wait*" specifies the time to delay before continuing with spoken text, and in which a whole or partial animation action may be completed.
- "*Duration*" specifies the time in which to either complete the rest of an animation action (if wait is specified), or to render an animation action from beginning to end (if wait is not specified).
- "*Repeat*" specifies the amount of times to repeat the animation action within the total wait and duration period, meaning that the larger the repeat attribute value, the faster each individual animation action will occur.

Using the above definition, example 1 should be rendered in such a way that the character would shake its head twice within a period of 3000 milliseconds, the speech starting one thousand milliseconds into this animation.

Another semantic issue arose with the use of some FAML elements. The problem appeared when it was not clear whether animation actions should finish back in the state in which they started or stay in the altered state. For example, consider the following situation:

<look-right> The window to your left </look-right> shows a map of Perth.

Example 2

From example 2 it is not evident whether the head should look to its right (the user's left) then finish back at its original state (facing the user) by the end of the text, or should start to look to its right and be looking directly to its right by the end of the text. There is also no explicit specification as to whether directions should be specified in respect to the character or the image of the character.

To allow a full range of motion and a high level of control it was decided that these animation actions should finish in the altered state. That means that the above example would result in the character facing to it's right

by the end of the text. This means that a "*look-left*" tag of equal intensity should be used to return the character to its original position. It is therefore important when implementing an animation action to ensure that opposite animation actions (eg. "*eyes-up*" and "*eyes-down*") animate the character in equally opposite ways so that it is always possible to return to the original starting point. An alternative or additional solution could be for the appendage of <look-straight> <head-straight> <eyes-straight> etc. to the standard. It was also decided that the direction of animation actions should be specified in respect to the character. In this way the animation will produce consistent rendering regardless of the user's viewpoint.

Another semantic issue arose when implementing the SML "*prosody*" tag and its attributes: "*pitch*", "*range*", and "*rate*". The problem concerns the specified range of 0-100 for these attributes. Semantically, pitch, range and rate should be altered by deviation away from the norm (relative values), not values in the range 0-100. Therefore it was decided that as per the Speech Synthesis Markup Language (SSML) standard [11] and VoiceXML standard [12] that numerical values should be specified as plus or minus a percentage with no imposed limit as to the magnitude of values (the limit is application specific). The use of these relative values is illustrated below.

<prosody rate="-100%"> I feel so tired today. </prosody>
Example 3

Version 0.4 of VHML, used in the MetaFace implementation, does not clarify the timing of sequences of elements and intervening plain text. In some situations it could mean a sequence of actions, in others, an overlap of actions. Looking at the example below, the "*look-up*", "*look-down*", and "*smile*" animation actions could possibly occur at the same time, the speech beginning one second later, or they could all occur one after the other in a sequence. There are also other ways to interpret the intent of the example.

<look-down wait="500ms"/><look-up wait="500ms"/><smile wait="1s"/>hello.
Example 4

To clarify the situation and allow a high degree of flexibility for character scripting, it was decided that tags should be read in a sequential way, the "*wait*" attribute not allowing an overlap with subsequent tags or spoken text, but the "*duration*" allowing only overlapping with subsequent tags and spoken text. This is further illustrated in example 5. The character will look down, then look up while smiling, continue to smile for one second, then continue to smile while talking (the speech will start three seconds from the start of the animation). This interpretation also means that the animated character is not controlled and limited by the presence of speech.

<look-down wait="1s"/><look-up duration="1s"/> <smile wait="2s" duration="2s"/> hello, how are you today.
Example 5

The semantic issues involving "*wait*", "*duration*", "*repeat*", "*prosody*", and animation states have been resolved during implementation of the MetaFace Framework. Whilst there are some recommendations for the next version of VHML, it was evident that other areas of VHML were straightforward in intent.

4.0 Functional issues

After consideration of semantic issues, the next step was to examine as to whether the VHML standard could support the functionality needed by the MetaFace framework for a complete implementation.

VHML was designed to give a range of functionality that could be used be many disparate systems. The elements central to this achievement are the "*paragraph*" and "*embed*" tag. The main use of these tags in the MetaFace framework is to update the content of the web browser. To show how this is achieved in VHML, consider the following example:

```
<p target="browser-south-frame">
        This is the MetaFace site:
        <embed type="html" src="http://www.metaface.computing.edu.au"/>
</p>
```
Example 6

As can be seen, "*target*" directs MetaFace to output of the paragraph to the "*south-frame*" of the browser (if there is no target, then the output is directed to the primary character). The problem with this example is that

the browser could either append the html file to the end of the paragraph text, or it could replace the paragraph text. When considering that there could be many embed tags following each other and that the "*type*" could conceivably be anything, then it is clear that not enough functionality is available to achieve specific goals for a system. It was thus decided that there should be an application specific attribute. To illustrate how this can be used to good effect, consider the following example:

```
<p target="fred">
        <look-down>Hey Bill, sing this song for me</look-down>
</p>
<p target="browser-south-frame">
        <embed type="html" src="http://www.metaface.computing.edu.au/songs/title.html"/>
        <embed type="html" src="http://www.metaface.computing.edu.au/songs/sheetmusic.html"
        application-data="3s;replace"/>
</p>
<p target="bill">
        <look-down>O.K. </lookdown><break/><look-up>Here goes</look-up><break/>
        <embed type="mml" src="http://www.metaface.computing.edu.au/songs/song.mml"
        application-data="transpose;tenor"/>
</p>
```

Example 7

Example 7 shows that "*application-data*" is used (in this hypothetical application) to control how one embedded html file replaces the other after three seconds, and later how a song is transposed to tenor. Adding the "*application-data*" attribute to the "*embed*" element means that all application specific data is limited to this one tag and the paragraph "*target*" attribute. The only requirement is that developers must accurately maintain documentation of the "*type*" and "*application-data*" attributes of "*embed*" as well as the "*target*" attribute of "*paragraph*" in order to allow reuse of VHML documents between different systems.

Another functionality issue that can be seen in the above example is that there is no way to know which paragraphs are concurrent and which are sequential. For example Fred could be looking at Bill, blinking, and smiling while Bill is talking.

```
<par>
        <p target="fred" >
                <look-right wait="3s"/><blink wait="1s"/><smile wait="1s" duration="4sec"/>
                <break="large"/><blink wait="2s"/>
        </p>
        <p target="bill" >
                <look-left>Now that I have your attention Fred</look-left> <look-right>I would like to
                talk about </look-right> an issue that I find very important.
        </p>
</par>
```

Example 8

Example 8 (for another hypothetical application) shows the use of the "*par*" element that is being considered for the VHML standard to control character scripting that occurs concurrently. Alternatively the "*seq*" element could be used to define a sequential order (this should also be considered the default). These particular elements were chosen because this is a particular concern of the Synchronized Multimedia Integration Language [13] (SMIL) and much attention has been paid to synchronisation and parrallelism.

During this first implementation of the VHML standard, there was attention paid to functional issues including application specific data and consequences there of, as well as the issue of concurrency. It was determined that VHML functionality could indeed support the MetaFace framework; the next step was to actually implement the system.

5.0 Implementation issues

XML document parsing is generally considered to be a straightforward task. The fact that in VHML there are two different ways to use tags (both atomic and around text) makes the job considerably harder. Couple this

with "*Simple API for XML*" (SAX) parsing instead of the relatively easier Document Object Model (DOM) tree approach and implementation issues become prevalent.

DOM tree parsing reads a document from start to finish, creating a tree structure for the developer's program process. SAX parsing allows the use of developer-implemented callbacks that are executed when encountering tags or text, thus is faster, and uses less memory. The disadvantage of SAX is that VHML is very much a state based language, and as such is better suited to the DOM-tree approach. However, in MetaFace speed is a priority. VHML documents are parsed on the server and Facial Animation Parameters (FAP's) are sent to the client, therefore the server needs to be able to output data in manageable sized segments with minimal delay even if the document is very large. This is not possible with DOM-tree parsing as the delay, due to the time the whole document is parsed, tree structure created and processed, would be apparent. SAX parsing would ensure that there is a small initial delay for the first segment, and then when the client is rendering that segment, the server is already creating and transmitting the next segment. Even though MetaFace uses MPEG-4 and client/server architecture these parsing issues should still be relevant to most systems.

Keeping track of state when using SAX parsing is a difficult process. The way that the MetaFace framework achieves this is by segmenting a VHML document into a series of paragraphs (denoted by or <paragraph></paragraph>). As text is encountered between tags it can be synthesised by the TTS NLP into phoneme information and from this, raw FAP's containing just visemes can also be constructed. The phoneme information and FAP's are then concatenated to the end of their respective lists. In this way a list of FAP's and phonemes can be kept as well as a list of animation actions that is updated when tags are encountered during a callback. The animation action list must also store where each animation action begins and ends in relation to the FAP list. At the end of a paragraph the animation list is iterated and the FAP list updated to reflect the animation tags of that paragraph. The phoneme information, modified FAP's, along with any embedded data is then transported to the client. In effect, each paragraph becomes a segment of data that is sent to the client, and so long as each is not too large, uninterrupted playback on the client is possible.

Segmenting VHML documents at the paragraph level raised the issue of what to do if animation actions exceed the paragraph boundaries. This can happen through using an atomic tag with a "*duration*" exceeding the spoken text of a paragraph, or using FAML tags that leave the character in an animated state other than the original position (such as turning to the left and holding that position at the end of a paragraph). This means that the current animated state needs to be kept to ensure consistency between paragraphs. This was accomplished by keeping variables in memory that store the last known animated state of the character, and when creating new FAP's from TTS phonemes, these variables are first applied to the FAP's, and any animation actions are overlapped (discussed later).

An important issue that also needed to be resolved was the need for transitions between some tags. For example a character can't go from sad to suddenly happy, there needs to be a transition. This is where the personality module is used. This framework module could implement psychological paradigms like the five-factor model [14], but currently MetaFace simply defines rules for the amount of time it should take to transition between certain tags. Then it is just a simple case of blending FAP's depending on the transition time and frame rate. Theoretically, each character could have it's own personality module, so in effect each character could animate a VHML document differently.

Another issue that became apparent in implementation was the need for overlapping actions that may animate a character in similar ways. Consider the following example:

<look-left><look-right>I should still be looking at you.</look-right></look-left>
Example 9

Example 9 shows a situation where the head is turning left as the head turning right. This could possibly create a conflict in some systems. MetaFace uses MPEG-4 facial animation techniques, so this is only a small problem. Since each facial parameter such as "*rotate head*" is updated for each FAP as the action list is iterated, when overlapping actions it involves of using the current parameter value as the origin value. In the example above this should result in the head staying still.

One of the most important issues encountered when implementing VHML involved the use of a non-VHML compatible TTS. For instance some of the Speech Markup Language (SML) functionality (volume and variety of voice) couldn't be implemented. It was also hard to accurately place where atomic tags would end in the synthesised speech, because they can implicitly affect the duration of the text. Also, because of the

way that MetaFace keeps expanding lists of FAP's, animation actions, and phoneme information, each segment of text between tags needs to be synthesised progressively to keep the animation synchronised (as visemes and timing depends on phoneme information). The disadvantage of this approach is that the intonation may be affected, as human speech usually follows a prosody contour for sentences, therefore synthesising segments of sentences can potentially degrade the intonation quality. Further, there are very few TTS NLP systems currently available that can synthesise emotion in speech, and even less that support atomic tags. Therefore when using a TTS NLP that uses tags around text and not atomic tags, the "*duration*" can only be estimated. The way that MetaFace supports atomic tags is by synthesizing a section of text between tags with the current emotion, the duration is then determined when the phonemes are returned. A counter can then be decremented, and when the counter reaches zero, the emotion is no longer used. This estimation of duration is of course affected by the size of the text segments being synthesized.

Many commercial TTS system developers are now supporting VoiceXML and SSML [15][16] on which the SML sub language of VHML is based. The future acceptance of these standards may make SML less of an issue in the future, but there still remains the lack of emotion support. Due to the face that VoiceXML and SSML deal with low level speech modification it could be possible for a developer to write their own emotion support which is a combination of low level tags and based on speech emotion theory, but it would require specialist knowledge in that area, as well as extensive output evaluation to ensure that users perceive emotions correctly. These problems are further compounded by the fact that facial animation depends on phonemes and their durations as generated by a TTS engine, and EML and SML tags directly change these phonemes and durations. VHML systems are therefore inherently dependant on their particular TTS engines.

6.0 Conclusion

This paper has outlined, discussed and resolved many issues from this first implementation of the VHML standard, both semantic, functional, and implementation based. All issues except the implementation of the SML sub language of VHML were resolved. The semantic and functional issues resolutions are being considered for the latest version of the VHML standard document (version 0.5). It is difficult to find a TTS system that supports emotion and is easily integrated with the application program. This will still be an issue even if SSML and VoiceXML become the standard for TTS systems of the future.

So far VHML has achieved a state in which there can be reuse of information whilst still allowing application specific complexity. VHML has proven thus far to be independent of the technology behind applications. Whilst not originally designed for the MetaFace framework and underlying implementation technologies (MPEG-4, SAX parsing, client/server architecture, personality, and two-dimension cartoon animation), VHML has supported them.

7.0 Future Work and Evaluation

An important future addition to VHML maybe the definition of implementation layers whereby, as the standard becomes more complex, allows developers to implement VHML tags to a certain level of complexity. This will allow for many developers of different skill levels and with various tools at their disposal to still interchange VHML data and adhere to a know level of conformance.

This first implementation of VHML has dealt with the practical issue of utilising and through this evaluating the standard. Future evaluation of VHML will be needed in order to further refine the standard. The complexity of animation also impacts with the development time and this deserves evaluation and refinement in future iterations of the VHML standard. Currently the MetaFace framework is being integrated with the Facial Animation Engine (FAE) from Eptamedia [17] to provide a three dimensional interpretation of VHML documents and offer further evaluation. Future evaluation should ascertain whether VHML inhibits or empowers believability and how the future standard could be further improved according to rules of believability such as: natural and unobtrusive behaviour, situated liveliness, complex behaviour, and control of visual impact [18]. Only with the implementation of further applications can it be truly known how effective VHML is and how it can potentially be improved. Some VHML demonstration animations, rendered by the MetaFace system, are available at the MetaFace website [19].

8.0 References

1. Marriott, A., 2002, *VHML*, [Online], Available: www.vhml.org [2002, March 22].
2. Beard, S., Reid, D. & Shepherdson, R. 2001, 'Believable and Interactive Talking Heads for Websites: MetaFace and MPEG-4', *6th International Computer Science Conference: Active Media Technology* (18-20 Dec) Hong Kong.
3. ISO/IEC, 2002, *Working Documents*, [Online], Available: http://mpeg.telecomitalialab.com/working_documents.htm#MPEG-4 [2002, April 9].
4. Marriott, A., Beard, S., Haddad, H., Stallo, J., Huynh, Q. & Tschirren, B. 2001. 'The Face of the Future', *Journal of Research and Practice in Information Technology*, Vol 32, No 3/4, August/November
5. W3C. 2002, *The World Wide Web Consortium*, [Online], Available: www.w3c.org [2002, March 22].
6. Beard, S., Marriott, A. & Pockaj, R. 2000, 'A Humane Interface' *OZCHI 2000 Conference on Human-Computer Interaction: Interfacing Reality in the New Millennium* (4-8 Dec) Sydney, Australia.
7. Stallo, J. 2000, 'Simulating Emotional Speech For a Talking Head', *Honours Thesis*, Curtin University of Technology, Perth, Australia.
8. CSTR. 2002, *The Festival Speech Synthesis System*, [Online], Available: http://www.cstr.ed.ac.uk/projects/festival/ [2002, March 22].
9. Beard, S., Stallo, J. & Reid, D. 2001, 'Usable TTS for Internet Speech on Demand', *OZCHI 2001 Talking Heads Workshop* (Nov 22) Perth, Australia.
10. Dutoit, T. 1999, *The MBROLA Project*, [Online], Available: http://www.festvox.org/mbrola/ [2002, March 22].
11. Walker, M. & Hunt, A. (eds) 2001, *Speech Synthesis Markup Language Specification for the Speech Interface Framework*, [Online], Available: http://www.w3.org/TR/speech-synthesis [2002, March 22].
12. McGlashan, S., Burnett, D., Danielsen, P., Ferrans, J., Hunt, A., Karam, G., Ladd, D., Lucas, B., Porter, B., Rehor, K., & Tryphonas, S. 2001, *Voice Extensible Markup Language (VoiceXML) Version 2.0*, [Online], Available: http://www.w3.org/TR/voicexml20/ [2002, March 22].
13. Ayars, J., Bulterman, D., Cohen, A., Day, K., Hodge, E., Hoschka, P., Hyche, E., Jourdan, M., Kim, M., Kubota, K., Lanphier, R., Layaïda, N., Michel, T., Newman, D., van Ossenbruggen, J., Rutledge, L., Saccocio, B., Schmitz, P., Kate, W. (eds) 2001, *Synchronized Multimedia Integration Language (SMIL 2.0)*, [Online], Available: http://www.w3.org/TR/smil20/ [2002, April 9].
14. McCrae, R., John, O. 1992, 'An Introduction to the Five-Factor Model and its Applications', *Journal of Personality*, vol. 60, pp. 175-215.
15. Nuance Communications. 2002, *VoiceXML*, [Online], Available: http://www.nuance.com/products/voicexml.html [2002, March 22].
16. Lucent Technologies. 2002, Speech Solutions, [Online], Available: http://www.lucent.com/products/subcategory/0,,CTID+2002-STID+10054-LOCL+1,00.html [2002, March22].
17. Eptamedia. 2002, *Eptamedia Srl*, [Online], Available: http://www.eptamedia.com [2002, April 8].
18. Johnson, W., L., Rickel, J., W. & Lester, J., C. 2000, 'Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments', *The International Journal of Artificial Intelligence in Education*, vol. 11, pp. 47-78.
19. Beard, S. 2002, *VHML Animation Examples*, [Online], Available: http://www.metaface.computing.edu.au/demos/vhml.html [2002, March 22].