ENGAGE: Automated Gestures for Animated Characters

Marcin Nowina-Krowicki, Andrew Zschorn, Michael Pilling and Steven Wark
Command, Control, Communications and Intelligence Division,
Defence Science and Technology Organisation,
Edinburgh, South Australia
{firstname.lastname}@dsto.defence.gov.au

Abstract

There is a rapidly growing body of work in the use of Embodied Conversational Agents (ECA) to convey complex contextual relationships through verbal and non-verbal communication, in domains ranging from military C2 to e-learning. In these applications the subject matter expert is often naïve to the technical requirements of ECAs. ENGAGE (the Extensible Natural Gesture Animation Generation Engine) is designed to automatically generate appropriate and 'realistic' animation for ECAs based on the content provided to them. It employs syntactic analysis of the surface text and uses predefined behaviour models to generate appropriate behaviours for the ECA. We discuss the design of this system, its current applications and plans for its future development.

1 Introduction

The Defence Science and Technology Organisation has an active research program into the use of multimedia narrative to provide situational awareness for military C2 (Wark and Lambert 2007). In common usage, face-to-face communication is the predominant, and often most effective, way for people to give and obtain complex contextual information. Embodied Conversational Agents (ECA) provide verbal and non-verbal communication modes similar to face-to-face communication. Gestures such as nods and facial expressions are very important in listener engagement with the speaker and their message.

Programming these gestures into an ECA animation is time consuming and requires specialised expertise. The subject matter experts developing content for ECAs are often naïve with respect to these technical requirements. A system to automatically generate appropriate non-verbal behaviour allows the content creator to concentrate on the information and not on how the ECA will animate it.

Figure 1 – Virtual Advisers present photo-realistic models of people

The BEAT system from MIT (Cassell et al. 2001) demonstrated this capability. DSTO has developed ENGAGE (Extensible Natural Gesture Animation Generation Engine) based on the principles demonstrated in BEAT, and extended them to incorporate modifiers such as confidence, importance, and urgency.

1.1 Virtual Adviser

DSTO has been using ECAs dubbed Virtual Advisers (VAs) as a mechanism for augmenting situational awareness in military C2 (Taplin et al. 2001; Wark and Lambert 2007; Wark et al. 2009). Virtual Advisers are computer generated talking heads using photo realistic textures with real-time animation and commercial-off-the-shelf text-to-speech (TTS) generation. Virtual Advisers can also include rolling text captions and multimedia monitors à la television news services. Virtual Advisers have been designed for modularity and can be delivered to users in a number of ways.

VAs are used to present situation briefs incorporating other media such as tables and diagrams, images, video, 3D models and so on. They are being used to provide prepared presentations, or dynamically generated content incorporating a dialog management system with a conversational interface (Estival et al. 2003). When connected to a decision support system they can also alert people to new or changing situations (Lambert 1999; Wark et al. 2003).

Virtual Advisers augment human support staff by providing a capability that can be deployed and accessed simultaneously from multiple geographically distributed locations. They can present the same information numerous times, on demand, without imposing an additional staffing burden. Virtual Advisers can augment existing decision support systems by explaining the information produced, not just showing it.

1.2 Talking Head Markup Language

Content is provided to VAs in the form of Talking Head Markup Language (THML). THML is tagged text that describes what the VA is to say and do. It includes commands to direct the VA: to say text; to adopt degrees of fundamental facial expressions (happy, sad, angry, afraid, surprised, contempt, disgust) (Ekman and Friesen 1977); to make eyebrow and head movements; and to direct gaze. It also includes commands to control the underlying TTS system, the appearance of the VA and its environment, and synchronise with other applications.

THML is designed to be simple for humans to read and write and to support on-the-fly authorship.

2 VA Architecture

Virtual Advisers are implemented using a modular, distributed architecture. All components communicate using a client-server model. The system consists of three core components; a rendering engine, system controller (THConsole), and Text-to-Speech service. Automated behaviour generation can be provided by ENGAGE. The content to be delivered by the VA can either be authored by a user or by a dynamic content generation system that feeds the THConsole the THML to be presented on demand.

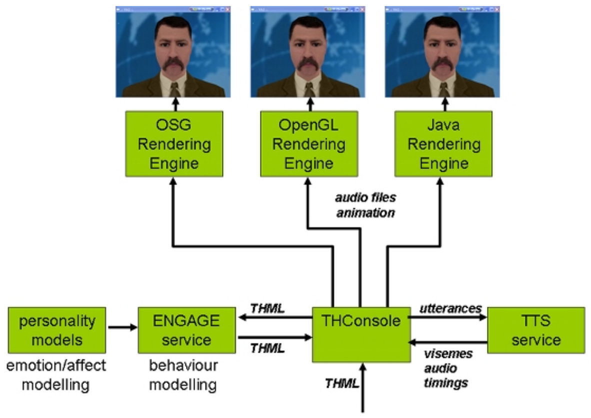

Figure 2 – The Virtual Adviser system

2.1 Rendering Engines

Rendering Engines are used to display the VA. They receive low-bandwidth rendering and timing instructions from the THConsole and output correctly synchronised 3D graphics, video, audio and application control.

C++ and Java toolkits have been developed to provide reusable, cross platform, core components to help facilitate the rapid development of new Rendering Engines for novel delivery mediums. These toolkits provide common underlying functionality such as: character animation; pluggable audio; instruction parsing; event based timeline; and networking support. The character animation system is built on top of the Cal3D library (Cal3D Team 2011). It provides skeletal and morph target character animation and a flexible model loading system. The Java Abstract Gaming Tools library (JAGaToo 2011) provides a port of the Cal3D library from C++ and is used in our Java toolkit.

Rendering Engines are developed by extending the core toolkits and providing environment specific support, such as accelerated 3D graphics and any other capabilities appropriate for the target medium.

Currently, VAs can be delivered in one of three ways: as a Desktop Application that can be controlled via an integrated Desktop service or invoked independently; embedded as an overlay or 3D model inside other applications such as DSTO's Virtual Battlespace II geospatial display (Wark et al. 2009); and as an Applet displayed on web pages and integrated into mainstream wiki systems, such as Atlassian's Confluence and the ubiquitous, open source, MediaWiki.

Desktop and embedded delivery is facilitated by a Rendering Engine built with the high performance OpenSceneGraph 3D library

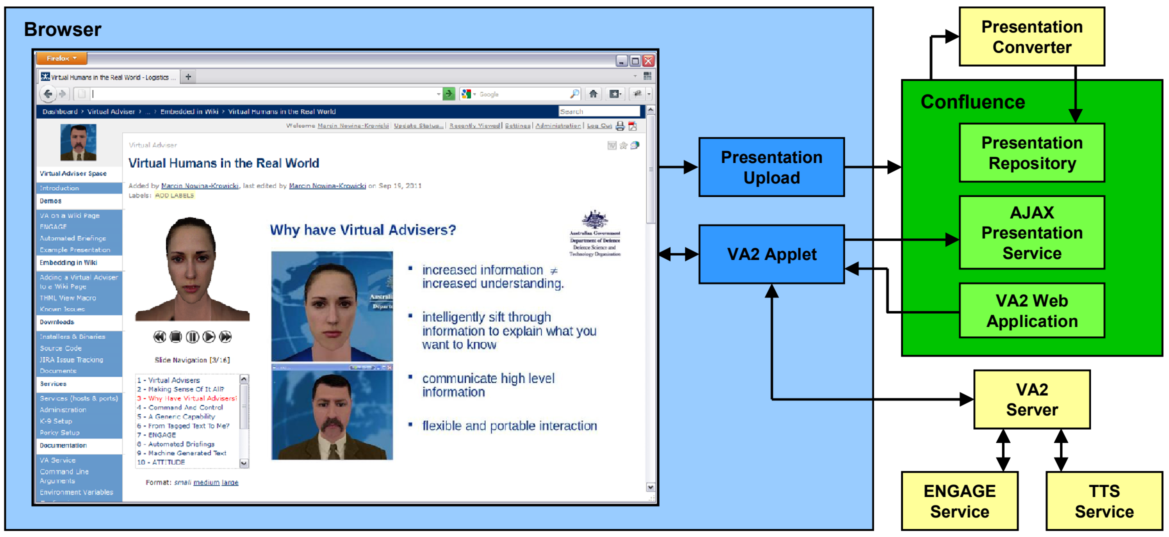
Browser
Virtual Humans in the Real World
Why have Virtual Advisers?
increased information ≠ increased understanding.
intelligently sift through information to explain what you want to know
communicate high level information
flexible and portable interaction
Presentation Upload
VA2 Applet
Presentation Converter
Confluence
Presentation Repository
AJAX Presentation Service
VA2 Web Application
VA2 Server
ENGAGE Service
TTS Service

Figure 3 – Virtual Advisers can be embedded on web pages to give dynamic presentations

(OSG Community 2011). Highlights of this solution include: integrated video and multimedia display; tickertape captioning; stereoscopic viewing; and Render-to-Texture support. The Render-to-Texture support allows the Virtual Adviser to be rendered as an overlay or texture in other applications.

Web delivery is via a Java-based Applet using the Java OpenGL (JOGL) bindings. The Applet works on all major platforms and web browsers that support the Java plug-in. Wiki integration for Confluence and MediaWiki allow users to easily embed and control a Virtual Adviser on a wiki page. A system for automatically presenting a converted PowerPoint presentation on a web or wiki page has also been developed and demonstrated.

2.2 THConsole

The Talking Head Console (THConsole) acts as the system controller. It interprets THML provided to it by a content generation system or user and coordinates the use of ENGAGE and the TTS service to produce the necessary animation instructions, synthesised audio and timing information, which is used by Rendering Engines to display VAs. Where ENGAGE is unavailable, the THConsole will process the script as is using only the marked up behaviour in the input THML.

The THConsole provides a flexible deployment capability. It is written as a small Java library that can be run in a number of different ways including: as an interactive CLI application that can have data either typed directly into it or piped from other processes or files; a TCP server that can be controlled via remote clients; or embedded as a component inside other applications and controlled using its public API.

How THML is handled depends on its context. Where commands are not inside an utterance, the THConsole can process them directly and send them to the Rendering Engine for immediate display. In contrast, utterances and the commands nested inside utterances are handled using a three pass process that requires the use of external services. The first pass optimises the input by chunking the say statement at sentence boundaries. The advantage of chunked input is that it greatly improves the throughput of both ENGAGE and the TTS and provides concurrency by allowing the Rendering Engine to begin executing one sentence while subsequent sentences are still being processed by the THConsole. The second pass expands the script using ENGAGE, if the service is available, to automatically generate behaviour. The final pass calls on the TTS service to generate synthesized audio and timing information for all events in utterance. The TTS results are then processed by the THConsole with timing information applied to all behaviour and actions in the utterance. Finally, rendering instructions are sent to the Rendering Engine for display.

2.3 Text-to-Speech Service

The Text-to-Speech service provides synthesized audio and timing information to enable synchronization of audio with animation and other events. In addition the service provides the ability to change the current voice, alter the speech rate and control volume. A TCP client-server architecture is used for service control. Generated files are served using a HTTP server to allowing a pull model where clients retrieve the audio and timing information as they need them.

Currently the TTS service uses Nuance's RealSpeak Solo 4 TTS engine (Nuance 2011). Other systems that have been used include Rhetorical System's rVoice TTS and the open source Festival Speech Synthesis System.

3 ENGAGE System

ENGAGE uses syntactic analysis of THML and behaviour models to generate appropriate synchronised behaviour. Parameters that control the application of these behaviour models can be embedded in the input speech instructions.

There are five main components in the ENGAGE system. Four of these components are arranged in a strict processing chain. The input stream is first sent to the Pre-processor, which prepares the input's speech instructions for syntax analysis. The Language component then adds syntax analysis to the speech parts of the input. The Behaviour component generates appropriate behaviours. Finally the post-processor produces mark-up for the virtual character system consisting of speech and synchronised behaviour. The fifth component of the system, the Behaviour Models, are used in both the language and behaviour components of the system to produce behaviour that is tailored to the current personality profile in use and model parameters provided in the input.

Following Cassell et al. (2001) and Lee and Marsella (2006), we use an XML document to store the processing results of each stage in the pipeline process. Each processing node is implemented as XSL transforms that can modify and augment the XML document. This pipeline approach ensures the separation of gesture generation from gesture realisation. This means that different behaviour models can be easily plugged in to achieve different behaviours in the VAs. The following sections examine each component in detail.

3.1 Pre-processor

The pre-processor prepares the THML input for processing. It takes as input a character stream of speech and other instructions and produces as output an XML tree ready for language syntax analysis. In the current implementation the Pre-processor uses a three stage process where input is first tokenised, then filtered and finally serialised to XML.

3.1.1 Tokeniser

The tokeniser is responsible for separating and extracting the various components of the input ready for filtering and serialisation to XML. It takes the character stream as input and produces an ordered list of "word" and "tag" tokens as output. The "word" tokens represent the dialogue that is to be spoken by the animated character, while the "tag" tokens represent all other instructions in the input stream, usually THML tags or

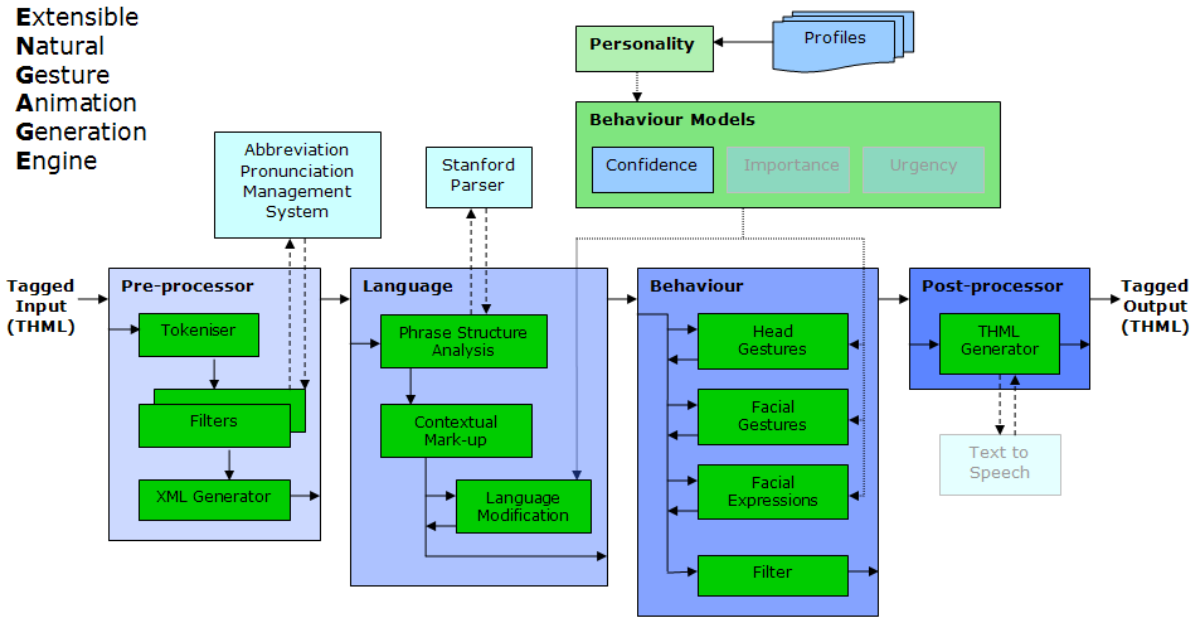

Figure 4 – The ENGAGE system.

ENGAGE processing instructions. The ordered list of tokens is then returned ready for filtering.

3.1.2 Filters

The filtering stage of the Pre-processor examines the input tokens and performs any additional processing on them ready for XML serialisation and processing. The filter set currently consists of an acronym and abbreviation filter and an ENGAGE tag filter for marking specific "tag" tokens as processing instructions for ENGAGE.

The acronym and abbreviation filter uses a context sensitive Abbreviation Pronunciation Management System (APMS) to expand any acronyms and abbreviated words. This allows correct phrase structure analysis in the language module and provides the Text-to-speech system with contextually correct phonetic spellings to facilitate correct pronunciation of the abbreviations.

3.1.3 Abbreviation Pronunciation Management System (APMS)

The correct pronunciation of some words, particularly abbreviations, can be difficult to determine from their written form with pronunciations often varying depending on the context in which they appear. Text-to-speech engines can do a very good job of inferring correct pronunciation of written forms, including initialisms and acronyms, but are not perfect, and don't have mechanisms to distinguish how different contexts can change pronunciations.

Large numbers of abbreviations are used in the defence domain, both in written forms such as reports, and in the spoken language. To always replace written forms with pronunciation forms directly in THML scripts would be tedious, and it would make the script harder for a reader to understand. Also, in the future we expect that THML scripts will be automatically generated from text that was never intended to be spoken by a VA. We want to make the process of authoring THML scripts simple and natural, to aid both authors and future automation. Thus, we want to move the problem of deciding how to pronounce abbreviations to the Virtual Adviser and away from the author.

We have developed the Abbreviation Pronunciation Management System (APMS) to replace written abbreviations with pronunciation spellings in a context-sensitive way in ENGAGE.

Consider the written abbreviation "RAAF", which can be pronounced as "R double-A F", "raff", or "Royal Australian Air Force". The pronunciation chosen can have a significant effect on comprehensibility of the speaker's message. For instance, it could be confusing to use the pronunciation "raff" when talking of a coalition military operation. On the other hand, using the longest form, "Royal Australian Air Force", could distract from the content of the message and socially separate a speaker from their audience if the context were an Australian military operation, where "raff" is the most common pronunciation.

In the APMS we use string tokens to identify contexts of abbreviation pronunciation. We allow contexts to inherit pronunciation replacements from a single parent, forming a branching hierarchy of contexts, or ontology. Child contexts may include different pronunciation replacements than its ancestors. This enables the addition of more specific contexts to handle more specific pronunciation replacements, while inheriting more general pronunciation replacements. For example, the context "general.australia.gov" may include the pronunciation replacement RAAF ⇨ "R double-A F", while the context "general.australia.gov.mil" may include the pronunciation replacement RAAF ⇨ "raff".

Database: The APMS database provides the persistent store of translations between text inputs and more vocally accurate textual or phonetic spellings. Each of these translations is given for a particular context. If no translation can be found in the given most specific context, progressively more general contexts are searched until a translation is found. For instance, the search may progress from "ship" to "Navy" to "military" contexts.

The system is implemented using PostgreSQL because of the richness of its stored procedure language and integral support for recursion for hierarchical data. This allows recursive searches to occur entirely within the database, avoiding returning intermediate results and executing recursion from the client which could be prohibitively expensive. Pronunciation lookup is done entirely server side. In normal operation, the system is further optimised by pre-calculating the best answer between voice, accent and context for any defined word and storing these answers in a cache. This noticeably enhances speed at the acceptable expense of higher disk usage.

As pronunciations necessarily drift and evolve, a script that had been rendered correctly may degrade as the underlying database evolves. The database records the times that pronunciations are added and when a pronunciation is revised the old version is retained. To access prior pronunciations, the caller need only specify a reference time and the database provides the pronunciation as it was then. For efficiency the system allows each caller their own cache which pre-calculates pronunciations for the caller's preferred reference time. It also provides a table in which to record such reference times, along with a short name and comment. This schema has the advantage of archiving all "snapshots" of the database online at very low storage cost, with only the snapshots in current use being instantiated out into the cache.

Usage: To use the APMS, THML scripts are marked-up to identify the context ontology it is to use. Scripts are then marked-up throughout to identify the current context for abbreviation replacement. As scripts are processed by ENGAGE, each space-separated word within '<say>' tags is analysed, given the current abbreviation context, to see if it should be replaced by a pronunciation spelling.

3.1.4 XML Generator

The XML Generator concludes the pre-processing stage by producing an XML tree from the tokenised and filtered input ready for Language and behaviour processing. The filtered "word" and "tag" tokens are marked up as XML elements in the pre-processed XML tree. Any "tag" tokens that have been marked as processing instructions for ENGAGE (such as behaviour models and parameters) are expanded and added as either attributes or elements depending on the scope of their behaviour.

3.2 Behaviour Models

Behaviour models are used to control and tailor the language and behaviour produced by ENGAGE. In this first version of the system, behaviour can be controlled using a Confidence Engine to manage the level of uncertainty displayed by the character. It is envisaged that future incarnations of the system will feature Behaviour Models for controlling the level of importance and urgency in the information being presented.

3.2.1 Personality and Personality Profiles

The Personality component provides the system with a means of varying language and behaviour parameters for the Behaviour model based on different Personality profiles.

Each Personality profile represents a set of language and behaviour parameters that can be used to alter the output of the various Behaviour Models. A Personality profile can inherit parameters from other personality models. This allows common traits to be pushed up to a common ancestor personality profile. In the first cut of the system this is achieved through cascading, where parameters are overridden by each successive include and can be further specialised in the child personality profile. In future a more powerful inheritance model will be implemented that allows groups or individual parameters to be included from specified parent profiles.

A User Interface has been developed to help generate personality profiles and tweak output behaviour. This interface provides the user with a set of parameter sliders that allow the various Behaviour Model parameters to be modified either individually or as grouped sets. The results of these changes can be tested and tweaked in real time allowing the user to see the results immediately.

3.2.2 Confidence

The Confidence Engine allows the system to control the level of uncertainty displayed by the character based on a confidence measure and parametric personality profile that can be assigned to the input utterance. Personality profiles are used to provide the Confidence module its parameters and allow the behaviour to be tailored for different personality types.

Currently the Confidence Behaviour Model is used in the Language Modification and Behaviour Generation phases of ENGAGE processing; how the Confidence Engine is applied will be discussed further in their Langauge Modification and Behaviour sections.

3.3 Language

Our primary intent is to generate natural-looking gestures to accompany the VAs speech. Therefore, and following Cassel (2000), Cassel et al. (2001) and Lee and Marsella (2006), syntactic analysis of the text to be spoken is important for behaviour generation and realisation. The text to be spoken is found within '<say>' tags in the THML scripts that drive the VA.

3.3.1 Phrase Structure Analysis

Each sentence found in THML '<say>' tags are sent to an automatic English parser for a full phrase structure analysis. At present we use the Stanford Parser to perform this function (The Stanford NLP Group 2011). The Stanford Parser uses a statistical method to perform phrase structure analysis. The tag set used is from the Penn treebank.

Syntactic and POS attributes are assigned as attributes to the individual word elements in the XML tree. These attributes can then be used in later processing stages such as contextual markup, language modification and behaviour generation.

3.3.2 Contextual Mark-up

Hiyakumoto et al. (1997) and Cassell et al (2001) use automatic theme/rheme analysis to aid behaviour generation, as it is stated that gestures are more frequently found in the rheme, or comment, part of the sentences (Cassell 2000). In order to perform automatic theme/rheme analysis these systems keep a record of all terms mentioned, and, broadly, determine that re-occurrences of those terms or closely related terms constitute the theme, or topic, of the clause.

At this stage the ENGAGE system does not maintain a history of words previously spoken by virtual characters. It is possible to add such a capability, and once done this will provide the system with a context-based approach for identifying the theme and rheme. Currently, for most suitable parses we simply identify all those words up to and including the head verb of the top-level phrase as forming the theme, and the remainder forms the rheme. Where the parser output is unrecognized, all the words up to and including the first verb in the sentence is labelled as the theme, and the remainder labelled as the rheme.

3.3.3 Language Modification

Language Modification is performed by applying the Behaviour Models to the language tree.

The Confidence Engine can insert disfluencies (as interjections), hesitations and information to alter speech rate into the XML language tree. The Confidence Engine uses the current confidence value assigned to the utterance and personality profile to determine if disfluencies and hesitations are added at various points in the utterance. Currently, disfluencies and hesitations may be added at any of the following locations:

- the start of the utterance
- before prepositions
- before verbs
- before nouns
- before the introduction of new domain words

Speech rate changes are added at both an utterance level and around inserted disfluencies.

3.4 Behaviour

The Behaviour phase of ENGAGE processing seeks to assign contextually appropriate non-verbal behaviours and expressions to the XML processing tree based on the markup added during the Language phase. As ENGAGE development is driven by the needs of the VA system, the current library of behaviours covers head gestures, facial gestures and facial expressions. Other gestures such as arm and body motion are envisioned for future development iterations of the system.

To generate behaviour ENGAGE runs the XML processing tree through a number of Behaviour Generators and the Behaviour Models. The XML tree is then pruned and passed to the Post-processing stage.

3.4.1 Head Gestures

For characters to emphasise objects and actions that they are introducing to the context, head-nods are generated for nouns and verbs in rheme sections of their speech.

The Confidence Behaviour Model can also add head drops and tilts to control the level of uncertainty displayed by the character, depending on the current confidence value and personality model in use.

Head drops are generated at changes in confidence value and influence the amplitude of head nods generated.

Head tilts may be added where there are hesitations with no disfluency in the speech.

3.4.2 Facial Gestures

For characters to emphasise objects and actions that they are introducing to the context, eyebrow movements are generated for nouns and verbs in rheme sections of their speech.

Also, in accordance with the way English and some other language speakers behave, eyebrow movements are added to sentences that end with a question-mark or exclamation-mark.

The Confidence Engine may specify changes in the rate and duration of blinks, as well as insert frowns and cheek puffs into the output tree. Cheek puffs and changes to blink rate and duration may be added to hesitations where there is no disfluency, while frowns may be added during disfluencies.

3.4.3 Facial Expressions

The Confidence Engine may specify changes in the level of anxiety, a combination of both anger and fear, displayed by the character. Anxiety may be changed whenever the confidence value changes.

3.4.4 Filter

The final stage of behaviour processing generates a filtered animation tree representing just the information that should be marked up in the post processing stage. The filtered tree produced consists of just words, tags and behaviours, with all extra language and intermediate processing tags pruned from the XML tree.

3.5 Post-processor

The Post-processor takes the filtered XML tree generated by the Language and Behaviour modules and generates a character stream of processing instructions for the VA. In our current implementation a THML Generator is used to produce the final ENGAGE output.

3.5.1 THML Generator

The THML Generator takes the resulting filtered XML tree generated by the Behaviour module as input and generates a character stream of synchronised THML instructions as output.

The output THML stream includes the speech, behaviour and other tags to be processed by the THConsole to generate appropriate instructions for the Virtual Adviser Rendering Engine.

In the current architecture ENGAGE does not use the TTS system to provide timing information for any of the marked-up behaviour that it produces. Instead, all behaviour is marked relative to the start or end of word, context or utterance boundaries. Appropriate timing information will be applied in the TTS processing pass coordinated by the THConsole. This approach allows ENGAGE to be an optional component of the system and also allows the THConsole to do further processing before generating timing information from the TTS without adding an unnecessary second TTS pass.

3.6 Responsiveness

As one of the usage modes of VAs is as a conversational interface, the speed at which it can produce results is important. ENGAGE is generally quicker to respond than TTS engines, so its impact on the overall system response time is negligible.

4 Future Work

Future work will look at semantic analysis of surface text to provide more targeted, contextually appropriate gestural animation.

The behaviour models currently used with ENGAGE have been developed as a proof of concept only. Further work is needed to refine these behaviour models to effectively communicate aspects such as uncertainty, importance, and urgency.

We also plan on investigating other Text-to-Speech solutions to provide finer control of prosody and expressive delivery of content to complement the animation.

5 Conclusions

The ENGAGE system developed at DSTO can be used to augment the real-time animation of ECAs by automatically inserting gesture animation based on the syntax of the sentences given to the system. This simplifies the task of generating 'realistic' behaviours based on surface text alone, supporting content authoring without requiring expertise in human behavioural modelling. In most cases observed so far, this has improved user engagement with the ECAs.

Acknowledgments

We wish to thank all the people that have contributed to the development of the Virtual Adviser system over the years. We would also like to thank the Research Leader C2 and Chief C3ID for their support and leadership.

References

Cal3D Team. (2011). Cal3D - 3D Character Animation Library, https://gna.org/projects/cal3d/

Cassell, J. (2000). "Nudge nudge wink wink: elements of face-to-face conversation for embodied conversational agents", *Embodied conversational agents*. MIT Press, pp. 1-27.

Cassell, J., Vilhjálmsson, H. H., and Bickmore, T. (2001). "BEAT: the Behavior Expression Animation Toolkit"*SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. City: ACM: New York, NY, USA, pp. 477-486.

Ekman, P., and Friesen, W. V. (1977). *Facial Action Coding System*, Pao Alto, U.S.A.: Consulting Psychologists Press Inc.

Estival, D., Broughton, M., Zschorn, A., and Pronger, E. (2003). "Spoken Dialogue for Virtual Advisers in a Semi-Immersive Command and

Control Environment"*4th SIGdial Workshop on Discourse and Dialogue*. City: Sapporo, Japan.

Hiyakumoto, L., Prevost, S., and Cassell, J. (1997). "Semantic and Discourse Information for Text-to-Speech Intonation"*ACL Workshop on Concept-to-Speech Technology*. City, pp. 47-56.

JAGaToo. (2011). JAGaToo - Java Abstract Gaming Tools, http://sourceforge.net/projects/jagatoo/

Lambert, D. A. "Advisers with attitude for situation awareness." *Presented at Proceedings of the 1999 Workshop on Defense Applications of Signal Processing*, LaSalle, Illinois.

Lee, J., and Marsella, S. (2006). "Nonverbal Behavior Generator for Embodied Conversational Agents", J. Gratch, M. Young, R. Aylett, D. Ballin, and P. Olivier, (eds.), *Intelligent Virtual Agents*. City: Springer Berlin Heidelberg, pp. 243-255.

Nuance. (2011). Nuance, http://australia.nuance.com/

OSG Community. (2011). OpenSceneGraph, http://www.openscenegraph.org

Taplin, P., Fox, G., Coleman, M., Wark, S., and Lambert, D. "Situation Awareness Using a Virtual Adviser." *Presented at Talking Head Workshop, OZCHI 2001*, Fremantle, Australia.

The Stanford NLP Group. (2011). The Stanford Parser: A statistical parser, http://nlp.stanford.edu/software/lex-parser.shtml

Wark, S., and Lambert, D. A. (2007). "Presenting The Story Behind The Data: Enhancing Situational Awareness Using Multimedia Narrative"*3rd IEEE Workshop on Situation Management (SIMA 2007)*. City: Orlando, FL.

Wark, S., Lambert, D. A., Nowina-Krowicki, M., Zschorn, A., and Pang, D. (2009). "Situational Awareness: Beyond Dots on Maps to Virtually Anywhere"*SimTecT 2009*. City: Adelaide, Australia.

Wark, S., Zschorn, A., Perugini, D., Tate, A., Beautement, P., Bradshaw, J. M., and Suri, N. (2003). "Dynamic Agent Systems in the CoAX Binni 2002 Experiment"*6th International Conference on Information Fusion (Fusion 2003)*. City: Cairns, Australia.