PAPER 5.

Beskow, J., Edlund, J., and Nordstrand, M. (submitted1). A Model for Generalised Multi-Modal Dialogue System Output Applied to an Animated Talking Head. To appear in *Minker, W., Bühler, D. and Dybkjær, L. (Eds) Spoken Multimodal Human-Computer Dialogue in Mobile Envirnonments*. Dordrech, The Netherlands: Kluwer Academic Publishers.

A MODEL FOR GENERALISED MULTI-MODAL DIALOGUE SYSTEM OUTPUT APPLIED TO AN ANIMATED TALKING HEAD

Jonas Beskow, Jens Edlund and Magnus Nordstrand[1]

Centre for Speech Technology
Royal Institute of Technology
Stockholm, Sweden

ABSTRACT

We present a high level formalism for specifying verbal and non-verbal output from a multi-modal dialogue system. The output specification is XML-based and provides information about communicative functions of the output, without detailing the realisation of these functions. The aim is to let dialogue systems generate the same output for a wide variety of output devices and modalities. We give examples from an implementation in the multi-modal spoken dialogue system AdApt, and describe how facial gestures are implemented in the 3D animated talking head used within this system.

Keywords: gesom, dialogue system, multi-modal output, talking head

1. INTRODUCTION

Speech is a useful modality, especially if one considers mobile devices, where e.g. a keyboard is not available. Using other modalities together with speech may alleviate some of the problems associated with spoken human-computer dialogue. Spoken dialogue systems incorporating some form of animated characters are becoming increasingly popular. There are many compelling reasons to include an animated agent in the interface. Since people have life-long experience at interpreting facial expressions and gestures (McNeill, 1992), it is one of the most intuitive and non-intrusive interfaces imaginable. Using gestures, an agent can continuously provide the user with feedback about the progress of the dialogue. This is an elegant way to handle problems with turn taking, potentially resulting in a smoother dialogue flow, while at the same time making the system appear more responsive. Given proper speech-synchronised articulatory movements and emphatic gestures, the agent will boost the intelligibility of the spoken output (Agelfors et al., 1998).

When considering dialogue systems in mobile environments, it is clear that coding output separately for each conceivable output device and modality is a complicated and time-consuming task. The task gets manageable if one leaves the output device to choose to present the output in a manner suitable for its capacity. Text, for example, may be presented as written text or speech, and emphasis signalled with either boldface, prosody and/or facial gestures. This chapter presents GESOM (GEneric System Output Model), a high level abstraction layer for specifying verbal and non-verbal output in a way that frees the dialogue system's output generation from the need to know details about the capabilities of the output device (see fig. 1) The aim is to allow the same dialogue system to work with a variety of output devices and modalities with a minimum of adaptation. A beneficial side

[1] Names in alphabetic order

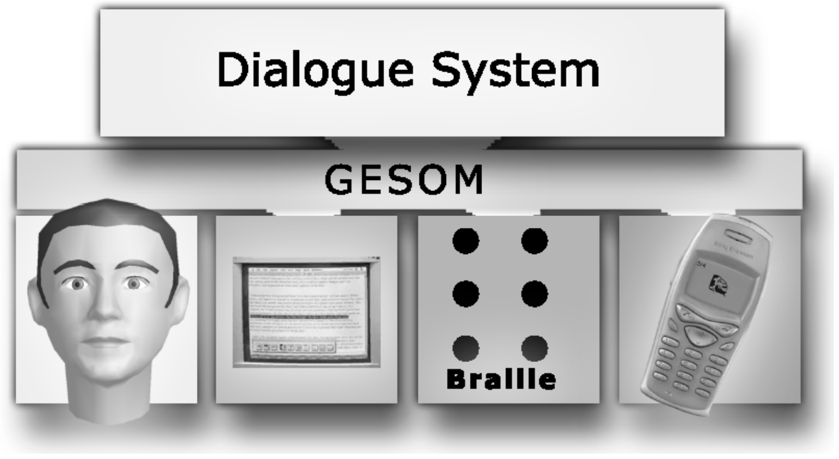


Figure 1. GESOM - a layer between the dialogue system and an output device.

effect is that output devices implementing GESOM may be used with any dialogue system conforming to GESOM.

1.1 Related work

Several models for automatic generation of gestures for animated characters in conversational systems have been proposed. Nagao and Takeuchi, 1994 present static facial displays for signalling communicative functions in a dialogue system. Pelachaud and Prevost, 1994 present a model for generating facial expressions and intonation from a common representation. Poggi and Pelachaud, 2000 present an agent capable of signalling its communicative goal, e.g. by showing emotions in the face. Thòrisson (1999) and Cassell et al. (2000) both describe complete frameworks for conversational dialogue systems that incorporate animated agents capable of generating deictic gestures, turn-taking signals, and emblematic gestures, relying on input from several sources. In contrast, our more limited model aims at separating the dialogue system from the realisation of output in order to facilitate rapid development and portability, which is a goal perhaps more closely related to that of proposed mark-up languages such as SSML (Burnett et al., 2002) and VHML (Gustavsson et al., 2001). Thorough discussions of these mark-up languages and others are found in Pirker and Krenn, 2002 and Gustavsson et al., 2002. In their present state, these languages tend to specify the output on a low level, with considerable detail, which makes them less well suited for our purposes. The work in this paper also builds on experiences gained from previous efforts at integrating animated characters into dialogue systems developed at CTT (Bertenstam et al., 1995, Beskow et al., 1997, Gustafson et al., 1999 and Granström et al., 2002).

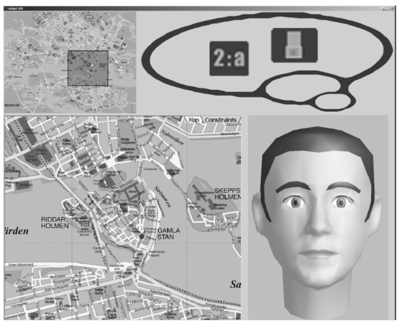


Figure 2a.

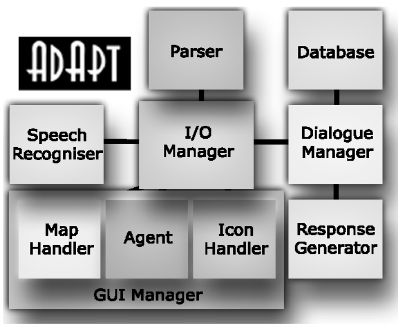


Figure 2b. Overview of the AdApt architecture

1.2 Background

The GESOM specification was created to cope with needs that arose during the development of the AdApt system (see fig. 2a), which was built at CTT, with Telia Research as an industrial partner (Gustafson et al., 2000). The system allows users to browse the real-estate market in downtown Stockholm, and features multi-modal input (speech, clicks) and a 3D-animated talking head producing lip synchronised synthetic speech (Beskow, 1997). It is used as a research platform for development and user testing, e.g. of multi-modal input and output. The system is modular (fig. 2b), in order to facilitate rapid implementation and integration of new functionality. When the output side of the AdApt architecture was developed, it was important to allow for rapid testing of different types of non-verbal output in the animated talking head. One of the goals of the system was to implement meaningful facial gestures, inspired by results in e.g. Cassell et al., 2000. In this process problems arose with issues such as backwards compatibility (the introduction of new entities in the output generated by the dialogue system would cause output modules to malfunction). Inter-modular communication in AdApt is encoded in XML, which is good for backwards and forwards compatibility, but the original specification for output generation clearly needed some work. In order to make the output robust and general, the following had to be addressed:

- The dialogue system should preferably not have to know too much about the output device and its capabilities.
- Some of the events one would want the dialogue system to signal are of unpredictable length, e.g. listening to or waiting for speech input, processing speech, or waiting for a database search to complete. The dialogue system does not know when these tasks will end until they in fact have. A method for handling this was needed.
- An animated talking head, which uses exactly the same gesture every time a particular event occurs, makes the dialogue system very repetitive. By specifying nothing more than the general pragmatic function to be signalled, the animated talking head could be allowed to choose any means available to realise the signal. This would not work if the dialogue system generates specific instructions for non-verbal output.

Whilst an XML specification that catered to these needs was written and tested for the animated talking head, other user interfaces were used in the development of the other parts of the dialogue system. Text input and output were used for regression tests, and for debugging purposes there were GUIs with coloured indicators signalling what the system

```
<!ATTLIST  output
              blocking      (0|1)       "0"
              callback      (0|1)       "0"
              background       CDATA       #IMPLIED   >
<!ELEMENT  state
              EMPTY                     >
<!ATTLIST  state
              type       CDATA      "default"
              name       CDATA      #REQUIRED
              background       CDATA       #IMPLIED   >
<!ELEMENT  event
              (#PCDATA)                 >
<!ATTLIST  event
              type       CDATA      "default"
              name       CDATA      #REQUIRED
              background       CDATA       #IMPLIED   >
```

Figure 3. GESOM 1.0 DTD.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE gesom PUBLIC "-//CTT//DTD GESOM 1.0//EN"
"http://www.speech.kth.se/gesom/v1/dtd/gesom.dtd">
<gesom xmlns:ges="http://www.speech.kth.se/gesom/">
<head/>
<body>
<output>this is as simple as it gets</output>
</body>
</gesom>
```

Figure 4. GESOM message: nothing but text.

was occupied with at any given time. Although the data sent to each of these interfaces was generated separately at the time, it became clear that most of the information could be unobtrusively built into a general XML specification, so that the same dialogue system output could be used as input for a variety of user interfaces. These considerations led to the GESOM specification.

1.3 Overview

In order for a formalism such as GESOM to work, a few requirements must be met. Firstly, the dialogue system must generate system output following the specification, which is proposed and explained next. Secondly, the dialogue system must be able to send the GESOM message to the output device. This is straightforward and can be done through any means available to both modules, e.g. TCP/IP sockets or pipes. Thirdly, the output device must know how to decode, interpret and realise, or render, the mark-up. Section 3 describes how to interpret the mark-up using standard XML techniques. Next, some examples of how GESOM messages might be realised in different output devices are given, and finally in section 4 we give a detailed description of how GESOM is interpreted and realised in the animated talking head used in the AdApt system. A brief summary and a listing pointing to future work conclude the chapter.

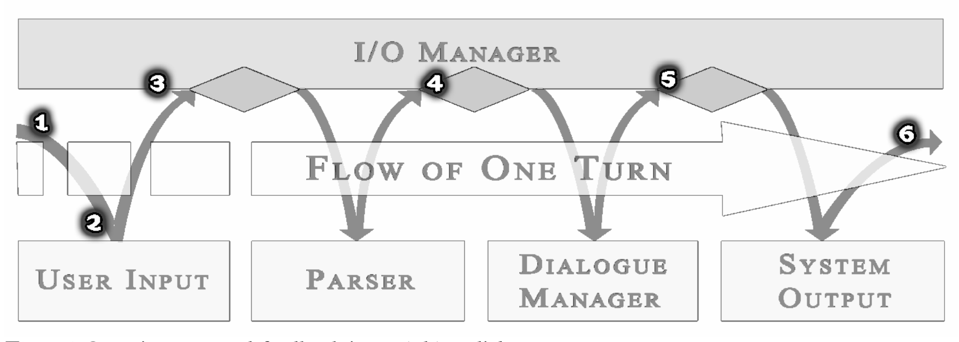


Figure 5. Occasions to send feedback in an AdApt dialogue turn.

2. SPECIFICATION

GESOM is fully compliant with the XML 1.0 specification (Bray et al., 2000). Complying with a standard has the advantage that there are many tools for viewing, editing, and, not least, validating messages. The GESOM 1.0 DTD (Document Type Declaration, i.e. XML grammar) is presented in simplified form in fig. 3. The actual DTD defines `<head>` and `<body>` elements, which are forwards compatibility considerations, as well as a number of formal XML entity definitions. These have been hidden here, but the semantics of the actual DTD are virtually identical to fig. 3.

Any textual content is sent as CDATA, and XML tags are used to mark other aspects. A minimal GESOM message, then, would look something like fig. 4. All GESOM examples in this chapter have identical elements down to, and including, the `<body>` element. For space and legibility reasons, the remaining GESOM examples will contain only the `<output>` part of the message. The example in fig. 4 contains nothing but the text to be communicated to the user. The remainder of the specification is somewhat more interesting. The next two sections describe the motivation and function of the `<state>` and `<event>` elements, as well as the background attribute.

2.1 The event and state elements

We found reason to categorise non-textual (or non-verbal) output into two categories: states and events. In addition, we use the attribute `background` to allow the dialogue system to pass general information to the output device. This categorisation is simple (probably too simple for certain applications), but as our goal was to allow a dialogue system to generate output without much knowledge of the output device, and the output device to realise the output without much knowledge of the dialogue system, a general, high-level abstraction was needed. The specification is largely based on what output a distributed dialogue system can be expected to produce, which seems necessary in order to keep the specification free from strong demands on the capabilities of dialogue systems and output devices alike.

Most dialogue systems, and indeed most interactive systems in general, are based around an event-driven model, i.e. actions that are carried out by the system are triggered by some kind of event. The events occurring in a dialogue system can either be the direct result of a user action (such as speaking) or internally generated during data processing. A dialogue system has, at the bare minimum, one user event, which we can call "input done": the user has finished speaking and the utterance is available to the system. The system will process

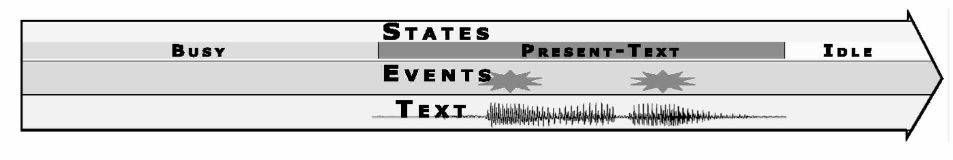


Figure 6. Timeline: a state lasts until another state is entered, whilst events are transient.

```
---
<output>
<event name="emphasis">this</event>
is an emphasised word and this
<event name="break" value="1000"/>
was followed by a 1 second pause
</output>
---
```

Figure 7. GESOM snippet: an emphasis and a break event.

the utterance and respond in some way, after which it will wait for the next "input done" event. This one-event-per-turn model is sufficient for simple systems, but in more complex systems we may want to give feedback during several stages in a dialogue turn, not just during the system's speech output. In the AdApt system, the following events at which the system may produce feedback occur during a dialogue turn (see fig. 5):

1. *Start of speech* - the speech recogniser has detected that the user is speaking.
2. *End of speech* - the speech recogniser has detected that the user has stopped speaking.
3. *Recognition done* - the speech recogniser has processed the utterance and passed the result on to the parser.
4. *Semantics done* - the parser has processed the recogniser output. The parser will categorise an utterance as either closing (the utterance can be interpreted in its own right) or non-closing (more input is needed to make sense of the utterance) (Bell et al., 2001). Closing utterances will be passed on to the dialogue manager. Non-closing utterances cause the system to go back to listening.
5. *Planning done* - the dialogue manager has decided what to do next, and a response is generated.
6. *Response done* - the system has presented its response.

A dialogue system may want to send non-textual transient output at times. By transient we mean that the duration of the output is finite, and predictable.This type of output is encoded in the `<event>` element. Examples from the AdApt system include emphasis and pauses (see fig. 7). Some of the things one would want the agent to communicate are poorly modelled by transient gestures, though. If we want to signal that the agent is performing an action, e.g. is listening, searching a database ("thinking") or just being idle, we need signals that are visible for an arbitrary amount of time, i.e. until the system stops doing whatever it is doing. This type of output is encoded in the `<state>` element. However, if the system is waiting for a database search to finish or for someone to pick up

```
---
<output>
<state type="turn-taking" name="present-text"/>
the present text state is used when outputting text, and the
idle state when the system is idle
<state type="turn-taking" name="idle"/>
</output>
---
```

Figure 8. GESOM snippet: the present-text and idle turn-taking states.

```
---
<output>
<state type="turn-taking" name="busy"/>
</output>
[database search completing]
<output>
<state type="turn-taking" name="present-text"/>
the apartment has three rooms
<state type="turn-taking" name="listening"/>
</output>
---
```

Figure 9. GESOM snippet: the busy, present-text and listening turn-taking states.

the phone, it has no way of predicting when this state will end. A state, then, has the following properties:

1. The output device must always be in one and only one state at any given time
2. A particular state lasts until another state is entered

Figure 6 shows a timeline where we pass through the states `busy`, `present-text` and `idle`. During `present-text` two transient `emphasis` events are marked. Synthesised text is represented as a speech waveform in the figure. The two properties of states listed above are valid within one state tier. In the present implementations, we have

only used one tier at a time. The specification, however, allows for an arbitrary number of tiers. The states in one tier are separate from the states in another. Tiers are encoded in the `type` attribute of the `<state>` element. If no type attribute is given, a default, single tier should be assumed. The tier used in the AdApt system is concerned with feedback for improving dialogue flow, and is called `turn-taking`. Figure 8 shows a GESOM message encoding two turn-taking states, `present-text` and `idle`, and fig. 9 demonstrates the `busy` and `listening` states. The latter example occurs when the AdApt system makes a database search or prepares a reply. The busy state ends when the reply is presented, since the present text state is triggered.

2.2 The background attribute

Finally, the specification defines an attribute `background`, which may be attached to any element. If the states and events are to be manageable and useful, they need to be limited in number. In some cases, however, the same state or event would best be realised in different ways depending on some other parameter. An example would be emphasis gestures (events) in an animated talking head: a small nod works well in most cases, but not if the sentence with the emphasised word is a negative reply, as in fig. 10. Defining the events negative emphasis, positive emphasis, and neutral emphasis would solve the problem. However, this

```
---
<output background="attitude:negative">
<state type="turn-taking" name="present-text"/>
the apartment does
<event name="emphasis" value="5">not</event>
have three rooms
<state type="turn-taking" name="listening"/>
</output>
---
```

Figure 10. GESOM snippet: the background attribute.

```
---
<output background="call synthesis(gesture(shake), no, I,
shall, not, conform, gesture(smirk))">
</output>
---
```

Figure 11. GESOM abused: some foreign code snuck into a background attribute. The message is otherwise empty.

kind of solution would cause the set of events to grow rapidly, thus defeating the purpose of this specification. Instead, the specification allows the dialogue system to send background attributes, which work much like global variables.

2.3 Permissible attribute values

The specification does not attempt strict control over what states and events are used. GESOM is designed to give as many output devices as possible a fair chance at doing something useful with any message. For the formalism to work well, however, some degree of agreement about what the available states, events and background attributes is necessary. To reach such an agreement, an ongoing discussion amongst dialogue system developers is necessary, and at this stage, we merely describe what we found useful in the AdApt system. Table 1 lists the states, events and background attributes that are used in the AdApt system. Note that all the states are on the tier `turn-taking`.

It should be noted that the background attributes should be used with care, since they permit any kind of content to be sent, and no particular restrictions are placed on their interpretation. They could easily be misused and thoroughly undermine the purpose of GESOM. Figure 11 gives an example of this.

3. INTERPRETATION

An obvious inspiration in developing GESOM has been the development of the Web protocols: HTML encoding in general, and browser, or user agent, compliancy guidelines in particular. An output device interpreting a GESOM message is analogous to a Web browser interpreting an HTML document. Thus, the output device should meet the criteria listed in e.g. the XHTML 1.0 recommendation, 3.2 User Agent Conformance (W3C HTML Working Group 3.2, 2002). In summary, the output device should:

Table 1. States and events that are implemented in the AdApt system.

States	
Name	**Description**
idle	System is inactive
attentive	System is ready for input
cont_attentive	System has received input that is not sufficient to prepare a response
busy	System is busy preparing a response
text_presentation	System is presenting a response
sleep	System is off-line
Events	
Name	**Description**
break	System should pause the presentation at this event
emphasis	Marked text should be emphasised
Background attributes	
Name	**Description**
attitude:negative attitude:positive attitude:neutral attitude:question	Relates to type of response

1. be able to parse GESOM messages for XML well-formedness and, preferably, to validate the message against the referenced DTD

2. disregard unrecognised elements, but process their content

3. ignore any unrecognised attributes

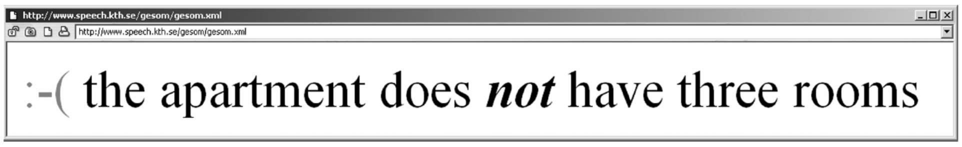
4. substitute any unrecognised attribute values for the attribute's default

These criteria serve to safeguard backwards and forwards compatibility, and make it possible for output devices with sparse capabilities to still do their best to interpret and render the content.

3.1 Existing tools and standards for XML rendering

Using XML encoded messages lets one take advantage of many existing tools and standards. The remainder of this section shows how to interpret GESOM messages with very little coding effort. With simple Extensible Style Sheet Language (XSL, W3C) transformations (XSLT, W3C) and/or Cascading Style Sheets (CSS, W3C), which can be used with any available XSLT and CSS processors, GESOM messages can be interpreted, transformed and/or realised. Note that the non-transient signals represented by state elements are not

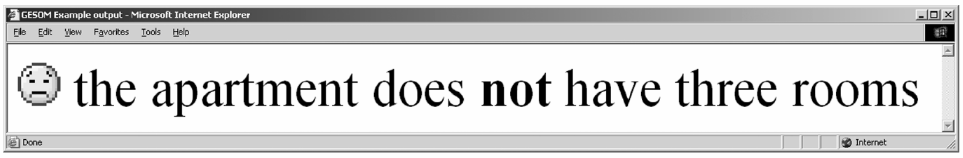
```
/* gesom2browser.css */
[background="response-type:negative"]:before  color: red; content:
":-( ";
[background="response-type:positive"]:before      color:    green;
content: ":-) ";
event[name="emphasis"] font-weight: bold; font-style: italic;
```


http://www.speech.kth.se/gesom/gesom.xml
http://www.speech.kth.se/gesom/gesom.xml

:-(the apartment does ***not*** have three rooms

Figure 12. Example of CSS.

```
<!-- gesom2browser.xsl -->
<xsl:template match="text()"> <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="@background">
<xsl:choose>
<xsl:when test=".='attitude:positive'">
<a href=http://www.speech.kth.se/gesom/img/positive.gif/>
</xsl:when>
<xsl:when test=".='attitude:negative'">
<a href=http://www.speech.kth.se/gesom/img/negative.gif/>
</xsl:when>
</xsl:choose>
</xsl:template>
<xsl:template match="event">
<xsl:choose>
<xsl:when test="@name='emphasis'">
<span style=''font-weight:bold''><xsl:apply-templates/></span>
</xsl:when>
</xsl:choose>
</xsl:template>
```


GESOM Example output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Done
Internet

the apartment does **not** have three rooms

Figure 13. Example of gesom XSL.

likely to be represented well in a static message, which is in itself transient. In the following web browser examples, the state elements are simply ignored. We start with a simple example using CSS to render the message in fig. 10 as a static message in a web browser. The CSS and the result of applying the style sheet on the GESOM message in a CSS compliant web browser are shown in fig. 12. Using a slightly more complex XSL transformation, we can get more interesting results. The essential bits of the XSL and the result in an XSL compliant web browser are shown in fig. 13.

The code in these examples can be changed to produce e.g. plain text, complete and highly formatted HTML messages, instructions for voice browsers, or the equivalent message in some other voice or speech mark-up language.CSS Mobile Profile (Wugofski et al., 2002) and similar technologies can be used to interpret the messages in PDAs. However, the range of output devices is not limited to XML based standard user agents. Any output device able to interpret the messages may be used.

In the next section we will show how the GESOM messages are interpreted and realised in the animated talking head, which is the AdApt system output module originally targeted by GESOM.

4. REALISATION IN AN ANIMATED TALKING HEAD

The AdApt system uses a 3D parameterised talking head that can be controlled by a TTS system to provide accurate lip-synchronised audio-visual synthetic speech (Beskow, 1997). The facial model includes control parameters for articulatory gestures as well as facial expressions. Parameters in the former category include jaw opening, lip closure, labiodental occlusion, tongue tip elevation, lip rounding and lip protrusion while the latter category includes controls for raising and shaping of eyebrows, smile, eyelid opening, gaze and head

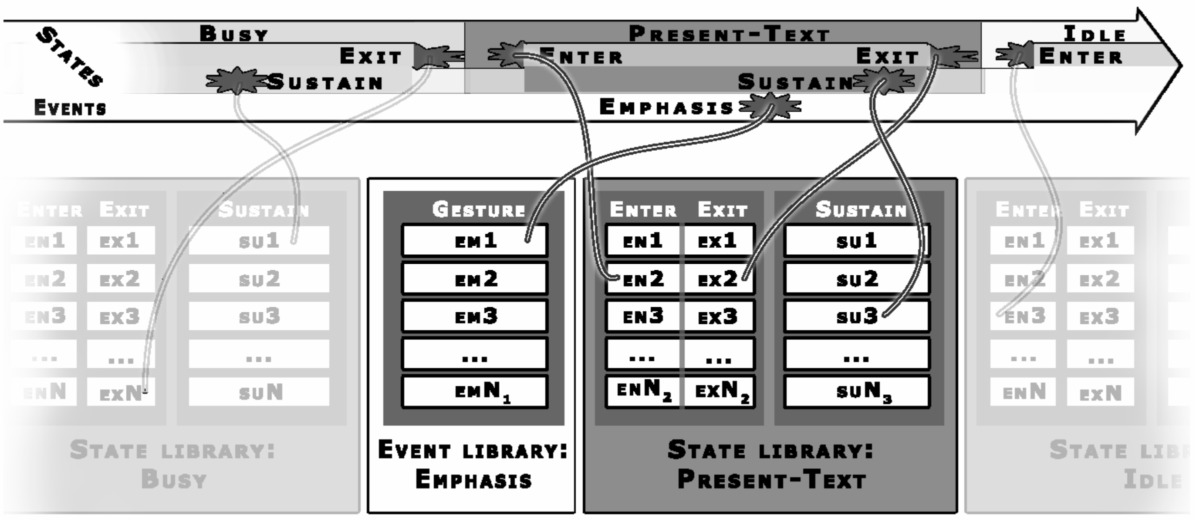


Figure 14. The structure of the gesture library. Gestures are picked from the library (bottom) and scheduled for realisation in the timeline (top).

movement. Gestures can be developed using an interactive parameter editor based on the WaveSurfer platform (Beskow and Sjölander, 2000).

4.1 Gesture library

The events, states and background attributes provide all the semantic information the animated agent needs in order to produce meaningful gestures, and the set of defined events and states constitute all the information that needs to be encoded both on the agent side and in the dialogue system. How the gestures should actually look is up to the agent. In our implementation, this information is encoded in a gesture library.
At the lowest level of the library are descriptions of the actual gesture realisations. For most states and events there are multiple realisations with subtle differences. For our parametrically controlled animated agent, the descriptions are coded in terms of parameter tracks. For other types of agents, they would be coded in other ways, for example as 2D-animation sequences. Our gesture realisation descriptions include the time offset to the stroke phase of the gesture, i.e. the core part that carries the meaning of the gesture. This information is used to synchronise the timing of gestures with other events such as stressed syllables of emphasised words.

The gesture library contains a separate entry for each event and state. For events, a set of alternative gesture realisations is defined. Gestures defined for a particular event will typically have similar semantic meaning - some gestures might only have subtle differences (e.g. in duration) whereas others may differ in style (such as a head nod vs. an eye widening gesture to signal emphasis).

Table 2. Example of how gesture weights can be modified in the presence of background attributes.

Event: emphasis			
Gesture	Weight	Modifier (default) attitude:positive	Modifier attitude:negative
Nod	2.0	1	0
Eyebrow widening	1.0	1	1
Eyebrow lowering	1.0	0	1

Each gesture has an associated weight that determines how likely it is to occur. By allowing alternative realisations, the agent will be less predictable and more natural in its behaviour.

In order to deal with the arbitrary length of states, they are divided into three phases: enter, sustain and exit. For each of the phases, as of the events, one out of a set of alternative gestures will be chosen. Gestures in the enter and exit phases are performed once per state (on state entrance and exit respectively), while the sustain gestures are executed at random intervals throughout the duration of the sustain phase. Enter and exit gestures are paired in such a way that, if a particular enter gesture is picked, the corresponding exit gesture will be chosen. This makes it possible for the exit gesture to restore the parameters that have been changed by the enter gesture. Figure 14 illustrates the layout of the gesture library.

4.2 Choosing between multiple realisations

Selection of a particular gesture is done in a weighted random fashion, based on the weights specified for each entry in the library. Although the gestures in each group are supposed to be semantically equivalent, there might be external factors making a particular gesture more or less appropriate at some given point. To reflect this in the library, the gesture weights could be dynamically updated, based on background attributes, or as a function of time. Table 2 shows how this can be represented. To begin with, each gesture within a category has a weight as seen in column two in the figure. The remaining columns are weight modifiers for different background attributes. The gesture weights are multiplied by the modifiers of the present background attributes. Note that several background attributes can be present at the same time. It is possible to specify a default modifier, that will be used if no modifiers match the present background attributes. This is needed when certain gestures should not be part of the default behaviour. In this case, the default modifier can be used to mask away the unwanted gestures by multiplying them with zero.

For example, when choosing a realisation for an emphasis event, an emphatic nod or an eye widening works well in the default case. If the utterance is of a negative nature ("there are *no* such apartments..."), the nod is less appropriate, but an eyebrow lowering would fit. Lets see what happens when the agent receives an emphasis event, given the gesture library representation in table 2. The agent needs to compute the probability *p(gesture)* for each of the three candidate gestures. If there are no background attributes, or if `background` is `attitude:positive`, the weights will be multiplied by the modifiers in the default (i.e. `attitude:positive`) column and then normalised, resulting in the probabilities *p(nod)*=2/3, *p(eye widening)* =1/3 and *p(eyebrow lowering)* = 0. If background is `attitude:negative`, the result will instead be *p(nod)* = 0, *p(eye widening)* =1/2 and *p(eyebrow lowering)*=1/2.

We can even let the weights be functions of time, by specifying them as an expression of t, where t is the time that has passed since the current state was entered. This makes it possible to model a gradual change of the agents behaviour during the sustain phase of a state, for example to simulate boredom or tiredness by increasing the probability of yawning as time passes. An example if this is shown in table 3.

Table 3. Example of gesture weights that change as a function of time (t).

State: idle, sustain gestures	
Gesture	Weight
Blink	5.0
Blink double	1.0
Yawn	t < 30 ? 0.0 : 1.0

4.3 Gesture co-articulation

Up until now we have considered each gesture realisation as being independent of preceding and following gestures. This is however an oversimplification - just as with speech, there is co-articulation among gestures. If a head nod is followed by a look-right-gesture, it may be unnatural if the agent returned to the neutral pose (straight ahead, which is the ending pose of the nodding gesture) before starting to turn the head sideways to the right. The natural thing would be to go more or less directly from the low point of the nod to the looking-right pose. To achieve this behaviour we have implemented a co-articulation algorithm that merges gestures that are overlapping or adjacent in time. The algorithm will always preserve the area around the stroke of each gesture, but segments before and after this area are subject to reduction. Reduced parts of the track are interpolated with a smooth spline curve.

5. DISCUSSION AND FUTURE WORK

The GESOM specification has only been tested extensively in the real estate browsing domain within the AdApt system. The only other output devices used have been computer emulated. Testing the system with real mobile phones, PDAs, Braille displays, as well as in other domains, would surely unearth shortcomings in the specification. Some known shortcomings:

- It would be useful to specify, or rather suggest, a number of fruitful sets of states, especially if states are to be used in more than one tier. Even though the specification in itself would allow any set, coding output with mark-up that is not implemented in any output devices would be a waste of time.

- The background attributes bear great similarity to the HTML/ XHTML style attribute, but are not as well thought through. An elegant solution is to use style sheets of some sort to control varieties in execution of gestures. Work along these lines has started, but could use more exploration.

- Although GESOM was developed within a system that, in addition to the animated talking head, uses a clickable map on which apartments are plotted, and visualises search constraints as icons Boye et al., 2002, the specification does not include this type of output. We would thus like to extend the specification so that it can handle the presentation of objects in a general manner.

- There is no good support for generating deictic non-verbal output. This seems straightforward to solve with respect to some output devices - nods, pointing and arrows could be used, but generalisation of the targets is troublesome.

- In most cases, the output device would be the same as the input device. The AdApt system uses the same input format for text whether it is typed, comes from a speech recogniser or from a stored dialogue log file. This format is, however, not particularly generalised or formalised. Extending the specification to cover (at least textual) input as well as output would make it a more useful tool. The representation of objects mentioned above might then extend to mouse clicks.

The system presented here is a first attempt at a generalised description formalism for multimodal output from a dialogue system. Our experiences from the implementation in the AdApt system indicate that it is successful in its pursuit, namely to form an abstraction layer between the dialogue manager and the output module, so that the dialogue manager does not need to know about the capabilities of the output module. The output module, e.g. an animated talking head, is responsible for suitable realisation of the communicative functions requested by the dialogue manager. Since the output description does not assume anything about the capabilities of the output device, it is fully possible to realise the output in some other way than through the gestures in an animated talking head. An alternative is to use familiar GUI metaphors, such as an hourglass for the busy state or a blinking red lamp for listening (recording) (Edlund and Nordstrand, 2002). This allows output to be generated on hardware incapable of rendering the animated agent, such as present day cell phones or PDAs.

Our current implementation of the animated agent uses a library of handcrafted gesture descriptions, grouped by communicative function. This is a very flexible model, since it allows us to model different attitudes, manners, personalities, moods or the socio-cultural identity of the agent simply by defining a new set of gesture descriptions (at least theoretically, assuming that the communicative functions are invariant). However, creating gesture realisations is a laborious process, and to convincingly model e.g. attitudes and emotions would require extensive studies of real-life subjects. A faster and more accurate way of obtaining the gesture realisations would be to record facial movement of an actor using a motion capture system such as Qualisys, 2002. Work towards this end is in progress at CTT.

ACKNOWLEDGMENTS

This research was carried out at the Centre for Speech Technology, a competence center at KTH, supported by VINNOVA (The Swedish Agency for Innovation Systems), KTH and participating Swedish companies and organisations.

REFERENCES

Agelfors, E., Beskow, J., Dahlquist, M., Granström, B., Lundeberg, M., Spens, K.-E., and Öhman, T. (1998). Synthetic Faces as a Lipreading Support. In *Proceedings of ICSLP*, Sydney, Australia.

Bell, L., Boye, J., and Gustafson, J. (2001). Real-time Handling of Fragmented Utterances. In *Proceedings of the NAACL Workshop on Adaption in Dialogue Systems*, Pittsburgh, PA. Bertenstam, J., Beskow, J., Blomberg, M., Carlsson, R., Elenius, K., Granström, B., Gustafson, J., Hunnicut, S., Högberg, J., Lindell, R., Neovius, L., de Serpa-Leitao, A., Nord, L., and Ström, N. (1995). TheWaxholm system - a progress report. In *Proceedings of Spoken Dialogue Systems*, Vigsø, Denmark.

Beskow, J. (1997). Animation of Talking Agents. In *Proceedings of AVSP'97*, pages 149–152, Rhodes, Greece.

Beskow, J., Elenius, K., and McGlashan, S. (1997). Olga - a dialogue system with an animated talking agent. In *Proceedings of Eurospeech'97*, Rhodes, Greece.

Beskow, J. and Sjölander, K. (2000). Wavesurfer - an Open Source Speech Tool. In *Proceedings of ICSLP 2000*, volume 4, pages 464–467, Beijing, China.

Boye, J., Gustafson, J., Bell, L., and Wirén, M. (2002). Constraint Manipulation in a Multimodal Dialogue System. *Proceedings of the ISCA workshop on Multimodal Dialogue in Mobile Environments*, Kloster Irsee, Germany.

Bray, T., Paoli, J. Sperberg-McQueen, C. M., and Maler, E. (2000). Extensible Markup Language (XML) 1.0 (Second edition). W3C Recommendation, last updated 2000-10-06, accessed 2002-10-15. http://www.w3.org/TR/2000/REC-xml-20001006.

Burnett, D. C., Walker, M. R., and Hunt, A. (2002). Speech Synthesis Markup Language Specification. W3CWorking Draft, updated 2002-04-04, accessed 2002-10-15. http://www.w3.org/TR/2002/WD-speech-synthesis-20020405/.

Cassell et al., J. (2000). Human Conversation as a System Framework: Designing Embodied Conversational Agents. In Cassell, J., Sullivan, J., Prevost, S., and Churchill, E., editors, *Embodied Conversational Agents*, pages 29–63. MIT Press, Cambridge, MA.

Edlund, J. and Nordstrand, M. (2002). Turn-taking Gestures and Hour-glasses in a Multi-modal Dialogue System. *Proceedings of the ISCA workshop on Multimodal Dialogue in Mobile Environments*, Kloster Irsee, Germany.

Granström, B., House, D., and Swerts, M. (2002). Multimodal feedback cues in human-machine interactions. *Prosody 2002*, Aix-en Provence, France.

Gustafson, J., Bell, L., Beskow, J., Boye, J., Carlson, R., Edlund, J., Granström, B., House, D., and Wirén, M. (2000). Adapt - a Multimodal Conversational Dialogue System in an Apartment Domain. In *Proceedings of ICSLP 2000*, pages 134–137, Beijing, China.

Gustafson, J., Lindberg, N., and Lundeberg, M. (1999). The August dialogue system. In *Proceedings of Eurospeech'99*, Budapest, Hungary.

Gustavsson, C., Strindlund, L., and Wiknetrz, E. (2002). Verification, Validation and Evaluation of the Virtual Human Markup Language (VHML) http://www.ep.liu.se/exjobb/isy/2002/3188/exjobb.pdf.

Gustavsson, C., Strindlund, L., Wiknetrz, E., Beard, S., Huynh, Q., Marriot, A., and Stallo, J. (2001). Vhml. W3C Working Draft, updated 2001-10-01, accessed 2002-10-15. http://www.interface.computing.edu.au/documents/VHML/2001/WD-VH%ML-20011021/vhml.html.

McNeill, D. (1992). *Hand and Mind: What gestures reveal about thought*. University of Chicago Press, Chicago.

Nagao, K. and Takeuchi, A. (1994). Speech dialogue with facial displays: Multimodal human computer conversation. In *Proceedings of the 32nd ACL'94*, pages 102–109.

Pelachaud, C. and Prevost, S. (1994). Sight and sound: Generating facial expressions and spoken intonation from context. In *Proceedings of the 2nd ESCA/AAAI/IEEEWorkshop on Speech Synthesis*, pages 216–219, New Paltz, NY.

Pirker, H. and Krenn, B. (2002). D9c: Report on the Outcome of the Markup Assessment Task. http://www.ai.univie.ac.at/NECA/publications/publication_docs/d%9c.pdf.

Poggi, I. and Pelachaud, C. (2000). Performative Facial Expressions in Animated Faces. In Cassell, J., Sullivan, J., Prevost, S., and Churchill, E., editors, *Embodied Conversational Agents*, pages 155–188. MIT Press, Cambridge, MA.

Qualisys (2002). http://www.qualisys.se.

Thòrisson, K. R. (1999). A Mind Model for Multimodal Communicative Creatures and Humanoids. *International Journal of Applied Artificial Intelligence*, 13(4-5):449–486.

W3C HTML Working Group 3.2 (2002). User Agent Conformance, http://www.w3.org/TR/xhtml#uaconf, in XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). W3C Recommendation, last updated 2002-08-01, accessed 2002-10-16. http://www.w3.org/TR/2002/REC-xhtml1-20020801/.

Wugofski, T., Dominiak, D., Stark, P., and Roy, T. (2002). CSS Mobile Profile 1.0. W3C Candidate Recommendation, last updated 2002-07-25, accessed 2002-10-16. http://www.w3.org/TR/2002/CR-css-mobile-20020725.