

# GESOM - A MODEL FOR DESCRIBING AND GENERATING MULTI-MODAL OUTPUT

Jens Edlund  
*CTT (Centre for Speech Technology)*  
*KTH, Sweden*  
edlund@speech.kth.se

Jonas Beskow  
*CTT (Centre for Speech Technology)*  
*KTH, Sweden*  
beskow@speech.kth.se

Magnus Nordstrand  
*CTT (Centre for Speech Technology)*  
*KTH, Sweden*  
magnusn@speech.kth.se

**Abstract** This paper describes GESOM, a model for generation of generalised, high-level multi-modal dialogue system output. Its aim is to let dialogue systems generate output for various output devices and modalities with a minimum of changes to the output generation of the dialogue system. The model was developed and tested within the AdApt spoken dialogue system, from which the bulk of the examples in this paper are taken.

**Keywords:** dialogue systems, multimodal output, architecture

## 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, e.g. synchronised visual articulation will add to the intelligibility (Agelfors et al., 1998), and facial expressions may be used to convey extra information through, e.g. turn-taking ges-

tures (McNeill, 1992). When considering dialogue systems in mobile environments, however, it is clear that coding output for each conceivable output device and modality is a complicated task. The task gets manageable if one leaves the realisation of the output to the output device, which may choose to present the output in a suitable way. Text, for example, may be presented as written text or speech, and emphasis with either boldface or prosody and facial gestures.

The present paper describes GESOM (GEneric System Output Model), a system for generating and realising dialogue system output. The goal was to create a system that allows the same dialogue system to work with different output devices and modalities with a minimum of adaptation.

## 1. Background

The AdApt system (Gustafson et al., 2000) was developed at CTT with Telia Research as an industrial partner. It 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).

GESOM was developed within the AdAptsystem, which was developed partially as a platform for development and user testing of multi-modal input and output. The AdApt system is modular, 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 be able to easily test different types of non-verbal output in the animated talking head. One of the goals of the system was to try to implement facial gestures in a meaningful way, 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 the output modules to malfunction). The AdApt system uses XML, which is good for backwards compatibility and generality, but the specification for the output generation clearly needed work. In order to make the system work well, the following had to be addressed:

- 1 The dialogue system should preferably not have to know too much about the output device and its capabilities.
- 2 Some of the events one would want the dialogue system to signal are of arbitrary length, e.g. listening to or waiting for speech input, processing speech, or waiting for someone to pick up the telephone. The dialogue system does not know when these tasks will end until they already have. A method for handling this was needed.

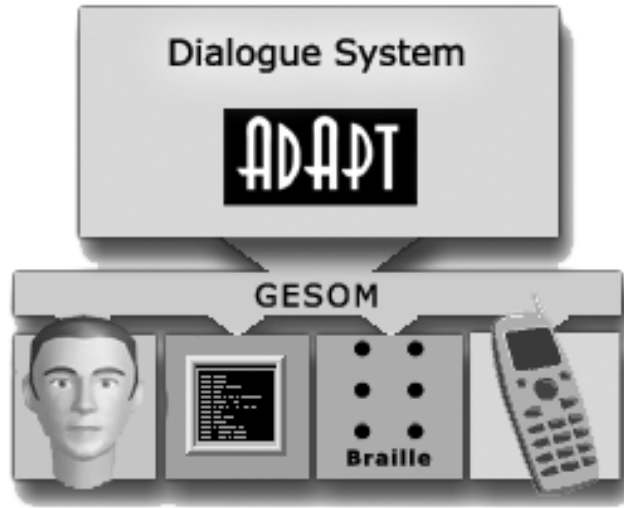


Figure 1. A layer between the dialogue system and an output device

- 3 When using an animated talking head, using exactly the same gesture every time a particular event occurs makes the dialogue system very repetitive. We addressed this problem by defining groups of gestures with the same pragmatic function that the output module can choose from at will, but this would not work if the dialogue system generated very specific instructions for non-verbal output.

At the same time as an XML specification that catered to these needs was developed and tested for the animated talking head, other user interfaces were used in the development of the other parts of the system. Text input and output were used for regression tests, and for debugging purposes there were GUIs with coloured indicators signalling what the system did. While the input for these interfaces was generated separately, it became clear that most of the information could be unobtrusively built into a general XML specification (see fig. 1).

## 2. Overview

The next few sections describe the requirements that must be fulfilled in order for GESOM to work. Firstly, the dialogue system must generate system output following the specification. Secondly, the dialogue system must be able to send this to the output device. This is straightforward

and can be done through any means available to both modules, such as TCP/IP sockets, pipes or Unix sockets. Thirdly, the output device must know how to decode and interpret the markup. The first and last requirements are discussed in the following sections, and examples of interpretations are given in section 5. A brief summary and listing directing to future work concludes the paper.

### 3. The dialogue system: markup generation

As noted above, GESOM uses XML markup. The specification says that any textual content is sent as CDATA, and XML tags are used to mark other aspects. This means that a minimal GESOM XML document could look something like this:

```
<?xml version='1.0'?>
<!DOCTYPE gesom>
<gesom>this is as simple as it gets</gesom>
```

Figure 2. XML message: a minimal GESOM message

In the AdApt system, we found that we needed two basic categories of non-textual output: *events* and *states*. In addition, we use background attributes to allow the dialogue system to pass general information to the output device. This categorisation is simple (too simple for certain applications), but since 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. This seems necessary to keep the specification free of strong demands on the dialogue system's and output devices' capabilities.

The specification does not attempt strict control over what states and events are used. As is discussed in section 4, GESOM is designed to give as many output devices as possible a fair chance at doing *something* with any input. For the system to work well, however, it would be best to have at least some agreement about what the available states and events should be.

#### 3.1 Events

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 as an *event*. Examples from the AdApt system include emphasis on a particular word (see fig. 3) and a pause (see fig. 4).

```
...the apartment does
<event name='emphasis'>not</event>
have a fireplace...
```

Figure 3. XML snippet: an emphasis event

```
...i have found such an apartment
<event name='pause' value='500' />
it has wooden floors and a balcony...
```

Figure 4. XML snippet: a pause event

Thus events are realised and then disappear.

### 3.2 States

Some things the dialogue system might want to communicate are radically different in that they are of arbitrary and unpredictable length. In many cases we wanted to signal what the system was doing - waiting for input, performing a database search, etc. These signals should preferably remain as long as the system is doing the same thing, but if the dialogue system wants to signal that it is busy waiting for some external event, e.g. for a database search to finish or for someone to pick up the phone, it has no way of predicting when this state of affairs will end. Signals such as these are encoded as *states*. A state is defined by the following properties:

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

These properties are valid within one *state dimension*. In the present implementations, we have only used states in one single dimension, i.e. *turn-taking*. However, the specification allows for more than one dimension, and the states in one dimension are separate from the states in another. In the XML mark-up, the dimension of a state is encoded in

the attribute *type*. If no type attribute is given, a default, single dimension is assumed. Examples of states from the AdApt system are shown in figs. 5 and 6.

```
<state type='turn_taking' name='present_text' />
the apartment has a fireplace
<state type='turn_taking' name='idle' />
```

Figure 5. XML snippet: the present text state followed by a text and the idle state

```
<state type='turn_taking' name='busy' />
...
database search completes
...
<state name='present_text' />
i could not find what you are searching for....
```

Figure 6. XML snippet: the busy state, a pause, and the present text state with a text

The latter example would occur when the 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.

### 3.3 Background attributes

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 so if the sentence with the emphasised word is a negative reply, as in fig. 7.

The problem could be solved by defining the events *negative\_emphasis*, *positive\_emphasis*, *neutral\_emphasis*. However, this 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 (see fig. 7).

The background attributes should be used with care, since they permit any kind of content to be sent, and no particular restrictions are

```

<gesom background='responsetype:negative'>
...
the apartment does
<event name='emphasis'>not</event>
have a fireplace
...

```

Figure 7. XML snippet: a negative response in the AdApt system

placed on their interpretation. They could easily be misused and truly undermine the purpose of GESOM. Fig. 8 gives an example of this.

```

<gesom
  background='call synthesis(gesture(shake), no, \
    I, shall, not, conform, gesture(smirk))'>
</gesom>

```

Figure 8. XML snippet: some foreign code in a background attribute. The message is otherwise empty.

#### 4. The output device: interpretation

Given that the dialogue system generating GESOM output should not have to know too much about the capabilities of the output device, some of the states and events generated may not be understood by a particular device. The same problem has haunted the Web since its conception, and a workable solution is often suggested for Web browsers: elements that are not understood should be *completely* ignored. In practice, an output device should be able to realise at least the textual content of a message. We believe that, given the right sets of states and events, a great many different output devices should be able to realise quite a lot of non-textual output from various types of dialogue systems. Some examples of how this works with the states and events used in our implementation are found in section 5.

#### 5. Implementation

GESOM was originally used to control an animated talking head in the AdApt system. Beskow et al., 2002 describes the gesture library used by the agent to realise states and events in a non-repetitive way, and gives a detailed description of how the animated head interprets the

```

<?xml version='1.0'?>
<!DOCTYPE gesom>
<gesom background='responsetype:negative'>
  <state type='turn_taking' name='present_text'/>
  <event name='emphasis'>no</event>
  you may not speak to a real operator
  <state type='turn_taking' name='idle'/>
</gesom>

```

*Figure 9.* XML: another negative response

specification. In 2002, the present authors performed a user study with the same system. One of the goals of the study was to compare the efficiency of turn-taking gestures in the talking head with an hourglass icon to signal that the system was busy preparing a reply. A third test configuration which offered no clues that the system was busy was also tested. The output of the dialogue system remained the same for all of the test configurations, but the animated agent's realisation of the output was changed. The first results of the test are described in Edlund and Nordstrand, 2002. As mentioned earlier, we have also found use for GESOM when developing the system in text-only setups. Finally, the method has proved very useful when comparing different facial gestures that could be used for the same purpose - the dialogue system can generate the same thing, e.g. a busy state or emphasis gesture, while the animated talking head uses different gestures to realise it.

## 5.1 Examples of output realisation

When the message in fig. 9 is realised by the animated talking head in AdApt, he does the following:

- 1 lowers his blink frequency considerably as a result of being in the `present_text` state
- 2 synthesises the text message, emphasising the word “no” with his voice as well as by lowering his eyebrows slightly
- 3 starts blinking at his normal frequency as a result of entering the `idle` state

On a Braille display, the text would simply be presented, perhaps with the word “no” in capital letters. On a text monitor, underlining or boldface might be used instead. A PDA or computer without synthesis

may do the same thing, and might present a red exclamation mark or some other icon to show that the response is negative.

```

Message 1:
<?xml version='1.0'?>
<!DOCTYPE gesom>
<gesom>
<state type='turn_taking' name='busy'/>
</gesom>
Message 2, some time later:
<?xml version='1.0'?>
<!DOCTYPE gesom>
<gesom background='responsetype:neutral'>
<state type='turn_taking' name='present_text'/>
you have been placed in a queue and we will get back
to you as soon as an operator is free
<state type='turn_taking' name='sleep'/>
</gesom>

```

Figure 10. XML: tired messages

The messages in fig. 10 are dealt with by the agent as follows:

- 1 frowns and looks away to signal that he is busy thinking as a result of being in the busy state
- 2 looks up as a result of leaving the busy state, lowers blink frequency and speaks the message
- 3 closes his eyes, dims the lighting and stops moving when entering the sleep state

Again, some output devices would be able to show the text only, possibly with some added message to say that the system is most likely unavailable when the sleep state is entered. Other devices might use an icon to signal that the system is asleep, i.e. a crossed out microphone or keyboard or a sleeping avatar. Other devices still may just take the sleep state as a key to disconnect.

## 6. 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 dimension. Even though the specification in itself would allow any set, coding output with markup 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. A more elegant solution would be to use style sheets of some kind to control varieties in execution of gestures. Work along these lines has started with the talking animated agent gesture libraries mentioned in section 5.
- 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 (Bell 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 much more useful tool. The representation of object presentation mentioned above might then extend to mouse clicks.
- Producing a complete and proper DTD, a standardised formal description, for the specification may not be very challenging, but it still needs to be done.

## 7. Acknowledgements

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

## References

- Agelfors et al., E. (1998). Synthetic Faces as a Lipreading Support. In *Proceedings of ICSLP*, Sydney, Australia.
- Bell et al., L. (2002). Constrain Manipulation and Visualization in a Multimodal Dialogue System. (this volume).
- Beskow, J. (1997). Animation of Talking Agents. In *Proceedings of AVSP'97*, pages 149–152, Rhodes, Greece.
- Beskow, J., Edlund, J., and Nordstrand, M. (2002). Description and Realisation of Multimodal Output Dialogue Systems. ICSLP'02, under review.
- Cassell, J., Sullivan, J., Prevost, S., and Churchill, E., editors (2000). *Embodied Conversational Agents*. MIT Press, Cambridge, MA.
- Edlund, J. and Nordstrand, M. (2002). Turn-taking Gestures and Hour-glasses in a Multi-modal Dialogue System. (this volume).
- Gustafson et al., J. (2000). Adapt - a Multimodal Conversational Dialogue System in an Apartment Domain. In *Proceedings of ICSLP 2000*, pages 134–137, Beijing, China.
- McNeill, D. (1992). *Hand and Mind: What gestures reveal about thought*. University of Chicago Press, Chicago.