

# humaine

## **D6e: Report on Representation Languages**

*Workpackage 6 Deliverable*



**Date: 30<sup>th</sup> June 2006**

<b>IST project contract no.</b>	507422
<b>Project title</b>	<b>HUMAINE</b> <b>Human-Machine Interaction Network on Emotions</b>
<b>Contractual date of delivery</b>	<i>June 30, 2006</i>
<b>Actual date of delivery</b>	<i>June 30, 2006</i>
<b>Deliverable number</b>	D6e
<b>Deliverable title</b>	Report on representation languages
<b>Type</b>	Report
<b>Number of pages</b>	66
<b>WP contributing to the deliverable</b>	WP6
<b>Task leader</b>	Paris8
<b>Author(s)</b>	Brigitte Krenn, Myriam Lamolle, Catherine Pelachaud, Hannes Pirker, Marc Schröder
<b>EC Project Officer</b>	Philippe Gelin

Address of lead author: Catherine Pelachaud  
IUT de Montreuil  
Université de Paris VIII  
140 rue de la Nouvelle France  
93100 Montreuil  
France

## Table of Contents

<b>1. INTRODUCTION .....</b>	<b>10</b>
<b>2. GESTICON.....</b>	<b>10</b>
<b>3. STATE OF THE ART .....</b>	<b>10</b>
<b>3.1. Representation Language .....</b>	<b>10</b>
<b>3.2. Facial expression coding .....</b>	<b>13</b>
<b>3.3. Gesture coding .....</b>	<b>14</b>
<b>4. THE OVERALL STRUCTURE OF A GESTICON ENTRY .....</b>	<b>14</b>
<b>5. REPRESENTING THE FORM OF A GESTICON ENTRY .....</b>	<b>15</b>
<b>5.1. The form element: general structure.....</b>	<b>16</b>
<b>5.2. Gesture entry .....</b>	<b>17</b>
<b>5.3. The symmetrical element.....</b>	<b>17</b>
<b>5.4. The arm element.....</b>	<b>18</b>
<b>5.5. The base_gesture element .....</b>	<b>19</b>
<b>5.6. The movement element .....</b>	<b>19</b>
<b>5.7. The timing element.....</b>	<b>20</b>
<b>5.8. The arm element and its subelement point .....</b>	<b>22</b>
<b>5.9. The hand element: general structure .....</b>	<b>23</b>
<b>5.10. The hand element and its subelement point.....</b>	<b>23</b>
<b>5.11. The hand_configuration element .....</b>	<b>24</b>
<b>5.12. The point element under hand_shape .....</b>	<b>25</b>
<b>5.13. The thumb_orientation element.....</b>	<b>26</b>

<b>5.14. The finger element</b> .....	<b>26</b>
<b>5.15. Example of arm gestures</b> .....	<b>26</b>
<b>5.16. Face entry</b> .....	<b>27</b>
<b>5.17. Gaze entry</b> .....	<b>30</b>
<b>5.18. The head element</b> .....	<b>31</b>
<b>5.19. The upper_body element</b> .....	<b>32</b>
<b>5.20. The posture element</b> .....	<b>33</b>
<b>6. EMOTION ANNOTATION AND REPRESENTATION LANGUAGE</b> .....	<b>36</b>
<b>6.1. Introduction</b> .....	<b>36</b>
<b>6.2. Different Descriptive Schemes for Emotions</b> .....	<b>36</b>
<b>6.3. Use Cases and Requirements for an Emotion Annotation and Representation Language</b> .....	<b>37</b>
<b>6.4. Proposed Realisation in XML</b> .....	<b>38</b>
1.1.1 Simple emotions .....	38
1.1.2 Complex emotions.....	39
1.1.3 Annotating time-varying signals .....	40
<b>6.5. A family of EARL Dialects: XML Schema design</b> .....	<b>41</b>
<b>6.6. Inventories of categories, dimensions and appraisals</b> .....	<b>42</b>
1.1.4 Categories.....	43
1.1.5 Dimensions.....	43
1.1.6 Appraisals.....	44
<b>6.7. Examples of applying EARL in different use cases</b> .....	<b>45</b>
1.1.7 Use case 0: Annotating text.....	45
1.1.8 Use case 1a: Annotating multi-level audio-visual databases with ANVIL.....	45
1.1.9 Use case 1b: Continuous annotation of emotional tone with Feeltrace .....	46
1.1.10 Use case 2a: Emotion recognition/classification of multi-modal input .....	46
1.1.11 Use case 2b: Emotion generation in ECAs .....	48
<b>6.8. Mapping emotion representations</b> .....	<b>50</b>
<b>6.9. Outlook</b> .....	<b>50</b>

<b>6.10. REFERENCES .....</b>	<b>51</b>
<b>APPENDIX A: SPECIFICATION OF THE HUMAINE EMOTION ANNOTATION AND REPRESENTATION LANGUAGE (EARL) .....</b>	<b>55</b>
Attributes common to all emotion elements.....	55
Samples Element <samples>.....	55
Simple Emotion Element <emotion> .....	56
Sub-elements .....	57
Complex Emotion Element <complex-emotion> .....	58
Sub-elements .....	58
EARL root element <earl> .....	59
Sub-elements .....	59
<b>APPENDIX B: EARL XML SCHEMA.....</b>	<b>60</b>
<b>B.1 The core schema defining the EARL structure .....</b>	<b>60</b>
<b>B.2 Schemas defining default sets of categories, dimensions or appraisals .....</b>	<b>63</b>
B.2.1 Categories .....	63
B.2.2 Dimensions .....	65
B.2.3 Appraisals .....	66
<b>B.3 Schema defining the default EARL dialect by binding together the EARL structure and the three default sets of content concepts .....</b>	<b>67</b>
<b>B.4 Defining other EARL dialects.....</b>	<b>68</b>
B.4.1 Defining new category sets.....	68
B.4.2 Defining new sets of emotion dimensions.....	69
B.4.3 Defining new appraisal sets .....	70
B.4.4 Defining new EARL dialects.....	71

# 1. INTRODUCTION

This deliverable reports on ongoing-effort on two representation languages, one related to behavior description, Gesticon, and the other one to emotion description, EARL. Both languages have been defined having in mind to be as broad as possible. In particular Gesticon aims to be player and graphics model independent. That is the description of a behavior should not be specific to any particular model geometry nor model animation parametrization. EARL has been defined to represent emotion description for different technological tasks: annotation of video corpus, analysing of video and synthesis of animation. Both languages are still being elaborated, in particular through discussions among Humaine partners across Workpackages (WP4, 5 and 6). The remaining part of the deliverable describes in detail each of these languages.

## 2. GESTICON

This is a proposal for a representation language for communicative gestures. We use the term gesture here in a broad sense referring to a variety of body behaviours such as facial expression, gaze behaviour, head movements, hand-arm gesture, posture as well as combinations of these. Thus a GesticonEntry can comprise the representation of unimodal as well as multimodal behaviours. For instance it can range from the representation of a specific gaze behaviour to the representation of a combination of facial expression, hand-arm gesture and posture.

Dictionaries of behaviors, in particular of communicative gestures, have been elaborated for German (Posner et al., 2003), Italian emblematic gestures (Poggi, 2002) and for other languages within the human studies community. A strong property of Gesticon is its independence of graphics models, player technologies and applications. GesticonEntry can be used within various agent systems. Creating gesture shape, facial expression, body posture etc. can be very time consuming. Allowing for the mutualisation of work and the sharing of behavior definition would greatly help the agent community. Gesticon has been designed for such a purpose.

## 3. State of the art

This state of the art has been taken from Deliverable 6b. It describes works that have been done in the area of representation language, facial expression and gesture coding.

### 3.1. Representation Language

A central issue that regards any work on agents is the language formalization that controls the agent's expressive behaviour. In fact this language bridges the gap between the agent's body and its mind. Technically speaking, such a language is part of the interface between modules which are responsible for determining an agent's behaviour on the one hand, and player technologies which are responsible for animation rendering and speech synthesis. This language is based on the emotion and personality theories that are used to implement the agents. The language also represents the various communicative acts an agent is able to perform.

Several efforts of creation of representation languages may be reported. A few of them have very large scope, such as HumanML and VHML. Other languages have been developed for a narrower scope. We present a brief summary of some examples of these languages.

There are two main efforts to create languages with very large scope: VHML (Beard et al., 2002), and HumanML<sup>1</sup>. HumanML is based on human-to-human and human-to-machine communications in digital information systems. The aim of HumanML is to embed important information related to humans so that this information can be transmitted in digital messages. HumanML provides tags related to emotions, physiology, proxemics, kinesics, haptics, intentions and attitude. It also provides attributes related to community, culture, context/location of the conversation, personality, thoughts, and signals. This language allows one to encode information related to human communicative behaviour from high level (culture, emotion) to low level (signal, kinesics). Virtual Human Markup Language (VHML) includes several sub-languages for each different modality (speech, face, gesture); the elements of the language may refer to low level information (right raise eyebrow) or to high level information (emotion anger). The authors have developed several sub-languages, each of them specialised in a dimension: dialogue management, emotion, facial animation, body animation, hypertext, and speech. VHML has a hierarchical structure, i.e. elements of a low level would inherit information at a higher level.

Several languages have been developed for specific applications: for sign language, SiGML (Elliott et al., 2004); for reproduction of gesture kinematics, MURML (Kranstedt et al., 2002); for a presentation agent, MPML (Mori et al., 2003); to represent semantic information, RRL (Piwek et al., 2002); for agent communicative behaviour, APML (DeCarolis et al., 2004), AML and CML (Arafa et al., 2002); for verbal and nonverbal synchronization, BEAT (Cassell et al., 2001). We will describe further the languages that are closer to our interest: namely, MPML, RRL, AML, and CML. APML will be described in greater details in the next sections.

Multimodal Presentation Markup Language (MPML) aims at developing a language to easily create animated agents within interactive presentations. Agents may be set up on the web and the user can interact directly with the agents. The general goal of MPML is that, unlike most other web agents for presentation applications, the presentation of information is no more presented sequentially, but, its content is generated dynamically as the conversation between the agent and the user evolves (Mori et al., 2003). Furthermore, MPML has been designed for mouse control, voice control, text-to-speech, agent's action description (Tsutsui et al., 2000). A specialized scripting language, SCRepting Emotion-based Agent Minds (SCREAM), may be interfaced with MPML (Prendinger et al., 2004). SCREAM has been designed to create emotionally and socially appropriate responses of animated agents placed in an interactive environment. SCREAM is specialized in scripting the agent's mind. SCREAM may be used within applications that compute the verbal content of the interaction happening between the user and the agent. The role of SCREAM is to compute the emotion that may arise during the conversation. The instantiation of the signals and their intensity for a given emotion is then computed taking into account many factors, such as the social setting of the conversation as well as the agent's mental state.

Character Markup Language, CML and Avatar Markup Language, AML (Arafa et al., 2002) are both languages to drive the animation of avatars from, respectively, a top-down approach (CML) and a bottom-up approach (AML). CML has been developed to bridge the gap

---

<sup>1</sup> HumanML, Human Markup Language, <http://www.humanmarkup.org>

between the emotion modelling processes and the displayed behaviour. Thus, CML provides mechanisms to compute the proper visual signals associated to a given emotion while considering the personality of the agent and its role in the interaction. Apart from describing the basic facial expression of emotions (Ekman, 1989), CML provides the definition of visual behaviours in term of basic movements (such as move, point, grasp) that may be adapted using attributes such as the agent's personality and the intensity values to define the expressiveness of a gesture. On the other hand, AML aims at developing a complete end-to-end MPEG4 multimedia framework (Doenges et al., 1997), especially for multi-user chat rooms and autonomous 3D characters (Arafa et al., 2002). While CML specifies information at a high level and provides the mechanisms that link communicative information to low level information, such as visual behaviours, AML defines elements at the behaviour level. Examples of behaviours are: pointing, smiling, walking. Each behaviour is specified with MPEG-4 parameters, namely the Facial Animation Parameters, FAPs, and the Body Animation Parameters, BAPs (Pandzic & Forchheimer, 2002). AML allows also for the specification of temporal information such as the starting time and duration of an action.

The Rich Representation Language, RRL, has been developed to manage the interaction between two or more virtual agents (Piwek et al., 2002). RRL is used as a link between a scene generator, a multi-modal natural language generator, a speech synthesis component, a gesture assignment component, and finally a media player. A scene description contains information related to the set of acts and the temporal ordering of the acts. An affective reasoner is embedded in the scene description to compute the corresponding emotion (defined by its type, intensity value and optionally the object that causes the emotion) that may be triggered by given acts. A scene description is given input to the multi-modal natural language generator that computes the corresponding linguistic and non-linguistic forms of the acts. The role of the speech synthesis and gesture assignment components is to instantiate the visual and acoustic data for a given emotion and dialog act. RRL uses a common representation (semantically and linguistically based) to automatically drive the animation of virtual agents interacting with each others.

The Affective Markup Language (APML) is based on a taxonomy of communicative functions proposed by Isabella Poggi (Poggi et al., 2000). A communicative function is defined as a pair (meaning, signal) where the meaning corresponds to the communicative value the agent wants to communicate and the signal is the behaviour used to convey this meaning. The former ones are represented as a set of goals and beliefs the speaker has wants to communicate. Communicative functions are differentiated in four categories:

- 1) information about speaker's beliefs
- 2) information about speaker's intentions
- 3) information about speaker's affective state
- 4) metacognitive information about speaker's mental state

The APML tags correspond to the meaning of a given communicative function. The conversion from meaning to signals is done by looking up the definition of each tag into a library that contains a lexicon for looking up (meaning, signals) pairs.

Noot & Ruttkay (2003) developed a very complex representation language, Gestyle, based on several dictionaries. Each dictionary reflects an aspect of the style (e.g. cultural or professional characteristics or personality). They also define the association of meaning to signals. In this language the authors embed notions such as culture, personality, gender but

also physical information such as gesturing manner or tiredness. To create an agent with style one needs to select a set of values (e.g. an Italian extrovert professor). The proper set of mappings between meanings and signals is then instantiated. The authors modelled explicitly how factors such as culture and personality affect behaviours.

Languages for specifying the multimodal behaviour of Embodied Conversational Agents (ECA) have proposed a direct and rather deterministic one-step mapping from high-level specifications of dialog state or agent emotion onto low-level specifications of the multimodal behaviour to be displayed by the agent (e.g. facial expression, gestures, vocal utterance). The difference of abstraction between these two levels of specification makes a definition of such a complex mapping difficult. (Abrilian et al. 2003) proposed an intermediate level of specification based on combinations between modalities (e.g. redundancy, complementarity) applied to the case of deictic expressions. Algorithms for parsing such descriptions and generating the corresponding multimodal behaviour of 2D cartoon-like conversational agents are suggested. Some random selection has also been introduced in these algorithms in order to induce some "natural variations" in the agent's behaviour.

### **3.2. Facial expression coding**

Several languages or coding schemes have also been proposed to describe facial expressions. There are the FAPs and the BAPs as described by MPEG-4 at a low level but also the Action Units as defined by FACS, Facial Action Coding System (Ekman and Friesen, 1978). Expressions may be described by combinations of these low-level parameters, or may be recursively defined as the blending of already defined expressions. More details on these coding systems are provided in the Deliverable 2 related to WP5. We report here just a brief summary.

Facial expression may be coded using the Facial Action Coding System, FACS (Ekman and Friesen, 1978; Ekman et al., 2002), a framework to measure facial signals using minimal action units (AUs). FACS allows one to encode facial action units using a scale of five intensities. Behaviour changes along this scale are carefully described. When used to code videos, FACS coders are required to describe movements that are visible at the start frame of the videos. On the successive frames, coders should note only facial actions that were not present in the previous frames as well as facial actions that were already present but for which the intensity changes. Thus, timing information related to the appearance and disappearance of an expression are recorded.

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). MPEG-4 (Doenges et al 1997, Pandzic and Forchheimer 2002) defines specifications for the animation of face and body models within an MPEG-4 terminal. Two sets of parameters describe and animate the 3D facial model: a facial animation parameter set (FAPs) and facial definition parameters (FDPs). The FDPs define the shape of the model while FAPs define the facial actions. When the model has been characterized with FDPs, the animation is obtained by specifying for each frame the values of FAPs. FAPs represent a large set of basic facial actions including head, tongue, eye, and mouth movement. Their combination allow the representation of facial expressions.

Paradiso and L'Abbate (2001) have established an algebra to create facial expressions. The authors have elaborated operators that combine and manipulate facial expressions. Another definition language has been proposed by de Carolis et al (2004). In their language, an

expression may be defined at a high level (a facial expression is a combination of other facial expressions already pre-defined) or at a low level (a facial expression is a combination of facial parameters). The low level facial parameters correspond to the MPEG-4 Facial Animation Parameters (FAPs). The language allows one to create a large variety of facial expressions for any communicative functions as well as the subtleties that distinguish facial expressions. It also allows one to create a facial display dictionary which can easily be expanded.

### **3.3. Gesture coding**

Methods for the encoding of gestures can be classified on a continuum ranging from purely semantic representations (related only to the meaning of the gesture) to purely pragmatic formats (related only to the form of the gesture). Most existing representation languages for computational systems are founded upon annotation systems in psychology and sign language research; we will therefore briefly review some of these foundational works, followed by an outline of scripting languages used for ECA systems. A much more detailed review of existing gesture coding schemes can be found in IST-2000-26095 Deliverable D2.1 by Serenari et al. (2000).

McNeill (1992) provides a semantic classification of gesture function into iconic, metaphoric, deictic and beat categories. He also introduces a grid-like gesture space in front of the actor to localize gestures. His descriptions of gesture form though are holistic-imagistic and not readily adaptable to automatic processing.

Stokoe et al. (1976) introduces the concept of breaking gestural configurations down into formational parameters: location of the wrist, orientation of the wrist, and hand shape. The more recent HamNoSys notation framework (Prillwitz et al., 1989), originally developed for sign languages, follows this breakdown into formational parameters and provides a dictionary of the most frequently used configurations.

FORM2 by Martell (2002) is an annotation scheme that captures exact configuration and orientation of the arms and hands of a gesturer. The annotation scheme, 'FORM', focuses on detailed gesture pragmatics. A corpus of annotated video material is available.

MURML (Kransted et al., 2002) is an XML based description language used in ECA gesture synthesis that allows for detailed control of parallel and sequential components of gestures. Hartmann et al. (2002) describe a language for ECA animation that unites features of McNeill and HamNoSys.

Analogous to the FAP scheme described above, the MPEG-4 standard defines a set of Body Animation Parameters (BAPs) that control joint angles of a standardized humanoid skeleton. As a pure animation data format, a BAP file does not preserve any additional information about the type of gesture executed. Another comparable skeletal animation format that allows arbitrary hierarchies is Biovision's BVH specification.

## **4. The overall structure of a Gesticon entry**

```

<GesticonEntry key ="UID" identifier="STRING" />
  <verbatim/>
  <form/>
</GesticonEntry>

```

Attribute	Value(s)
key	UID
identifier	shrug frown .. .

Each GesticonEntry comes with a unique key and an identifier being a short key to the main information conveyed by the Gesticon entry.

The verbatim element contains a human readable description of the Gesticon entry.

In the Gesticon representation language, we concentrate on the signal. The pairing meaning-signal is done outside of the Gesticon in meaning-signal tables. This eases variation of behaviour. See for instance work by Allbeck et al. (2002), Beard et al. (2002), Ruttkay & Pelachaud Ruttkay (2002), Cassell et al. (2001). Moreover everything what is player-specific, such as visemes and information on faces, will be external to the Gesticon.

## 5. Representing the form of a Gesticon entry

We call a <form> element any communicative non-verbal behaviour. It comprises the subelements: gesture (arm movement), hand\_configuration, face, gaze, head, upper\_body and posture.

GesticonEntry	Description
<gesture>	Coordinated movement with arms and hands, including pointing, reaching, emphasizing (beating), depicting and signalling.
<hand_configuration>	Hand shape, fingers, orientation of the thumb.
<face>	Movement of facial muscles to form certain expressions.
<gaze>	Coordinated movement of the eyes, neck and head direction, indicating where the character is looking.
<head>	Movement of the head independent of eyes. Types include nodding, shaking, tossing and orienting to a given angle.
<upper_body>	Movement of the spine and shoulder.

<posture>	Movements of the body elements downward from the hip: pelvis, hip, legs including knee, toes and ankle.
-----------	---

Any combination of the subelements is also possible. The particular decomposition is motivated on the one hand by high-level considerations such as a) physiology (muscular contraction and joint articulation), and b) existing studies on communicative non-verbal behavior. On the other hand there are computational factors. For instance: the same hand configuration can be used with several arm movements. Gaze and head movement are separated in order to provide more flexibility for animation of gaze behavior, in particular allowing head movement while gazing at something. In our formal specification we use <upper\_body> for characterizing spine movement and shoulders. While in the posture element pelvis and legs are specified. We distinguish two types of identifiers, the one is a unique reference (UID) of the specific subelement, the other one is a shortcut to a human readable description of the behavior.

In the following sub-section we describe each subelement and introduce other elements that can be used across elements such as the entries to describe the timing and the trajectory of a movement.

## 5.1. The form element: general structure

Similar to GesticonEntry the subelements of form are assigned a unique identifier (UID) and a more verbose identifier the value of which is a string functioning as a human readable shortcut to the represented form.

The form element of a GesticonEntry must comprise at least one of the subelements gesture, hand\_configuration, face, gaze, head, upper\_body and posture. Any combination of the subelements is also possible.

For some form elements a rhythm attribute is defined. For the moment being the only possible value is repetition, indicating that the gesture shall be repeated when being part of an animation.

```
<form>
  <gesture id="UID" rhythm="STRING" identifier="STRING" />
  <hand_configuration id="UID" rhythm="STRING" identifier="STRING" />
  <face id="UID" rhythm="STRING" identifier="STRING" />
  <gaze id="UID" identifier="STRING" />
  <head id="UID" rhythm="STRING" identifier="STRING" />
  <upper_body id="UID" rhythm="STRING" identifier="STRING" />
  <posture id="UID" rhythm="STRING" identifier="STRING" />
</form>
```

Attribut	Value(s)
----------	----------

e	
id	UID
rhythm	repetition
identifier	STRING

## 5.2. Gesture entry

Gestures are complex, being composed by one or a sequence of basic gesture elements, each of which describing a basic hand-arm movement trajectory. A trajectory is defined by a sequence of key points where each point corresponds to a position of the wrist in 3D space<sup>2</sup>, and a hand configuration. Depending on the granularity of representation a gesture representation spans one or more phases. Via the phase attribute we encode gestural phases such as preparation, stroke, hold, retract, etc. In order to account for theories of hand-arm gesture where no phases are distinguished, e.g. (Martell, 2002), only the “start” and “end” labels are used. The hand description follows the ASL hand shape configuration description (Stokoe et al, 1976). It is composed of a hand shape, a thumb orientation, and fingers shapes. The static description is taken from MURML (Kransted et al., 2002; Kopp & Wachsmuth, 2004). For each of the subcomponents movement can be specified in order to allow for the representation of complex hand and finger movements. Such a complex representation is realized by means of a collection of gesture elements with different type attributes, and which are aligned via synchronization points.

The gesture element is the subelement of form where hand-arm gestures are specified. It comprises two subelements symmetrical and arm.

```
<form>
  <gesture>
    <symmetrical/>
    <arm/>
  </gesture>
</form>
```

## 5.3. The symmetrical element

The representation of symmetry is taken from MURML. According to MURML one arm is specified as the leading (dominant) arm whereas the other arm follows. The possible mirror symmetries are specified via the attribute symmetry, and are defined as configurations of the main body planes, with the following combinations: MS standing for sagittal, MT for

---

<sup>2</sup> In the current description we restrict arm movement to wrist movement using inverse kinematics to compute elbow articulation.

transversal, MF for frontal, MST for sagittal-transversal, MSF for sagittal-frontal, MTF for transversal-frontal and MSTF for sagittal-transversal-frontal.

<symmetrical dominant="STRING" symmetry="STRING">

Attribute	Value(s)
dominant	right_arm left_arm
symmetry	Sym SymMS SymMT SymMF SymMST SymMSF SymMTF SymMSTF

## 5.4. The arm element

In our approach the arm element is hierarchical. It is composed of one or a sequence of base\_gesture elements, each of which describes a basic hand-arm movement trajectory. A trajectory is defined by a sequence of key points where each point corresponds to a position of the wrist in 3D space<sup>3</sup>, and a hand configuration. The base\_gesture element can also be characterized by a phase attribute. The arm element contains one or more base\_gesture elements.

```
<arm id="ID" type="left" reversible="yes" >
  <base_gesture/>
  ...
  <base_gesture/>
</arm>
```

Attribute	Value(s)
id	ID
type	left right
reversible	yes no

As regards the attribute id, we distinguish throughout the Gesticon between two types of values: ID and UID. IDs are used local to a GesticonEntry, i.e. the id is unique only within a single GesticonEntry whereas id="UID" specifies a global unique id within the Gesticon. Each Gesticon\_entry has its UID. We employ ID in particular for relative timing in the form element.

---

<sup>3</sup> In the current description we restrict arm movement to wrist movement using inverse kinematics to compute elbow articulation.

The attribute type specifies which arm (left or right) is specified under the current arm element. The attribute reversible provides additional information to the gesture generation engine: reversible="yes" says that the arm-hand configuration specified can be used by the engine to automatically generate the gesture of the other arm.

## 5.5. The base\_gesture element

The base gesture element is the container for the movement element. Via the attribute phase of base\_gesture a complex gesture is divided into gestural phases such as preparation, stroke, hold, retract. In order to account for theories of hand-arm gesture where no phases are distinguished (e.g. Bielefeld GeWi; Martell (2002)), the phase attribute is optional.

```
<base_gesture id="ID" phase="STRING" >
  <movement/>
</base_gesture>
```

Attribute	Value(s)
Id	ID
phase	preparation stroke hold retract

## 5.6. The movement element

While the base\_gesture element is specific to gesture, the movement element is not specific to a certain modality. It is designed to describe the trajectory of any articulated element, i.e. finger, hand, upper body, legs, head.

The movement element comprises two subelements timing and point. While the timing element can only occur once per movement element, several point elements can be specified for one movement element.

```
<movement trajectory_type="STRING" beat="yes">
  <timing/>
  <point/>
  ...
  <point/>
</movement/>
```

Attribute	Value(s)
trajectory_type	linear curve ellipse circle non

e	e
beat	yes

Trajectory\_type specifies the linear realization of the movement. Linear, curve, ellipse and circle are standard trajectories and widely used in animated character design.

Beat specifies that “beat” movements are to be generated when the gesture is aligned with speech. I.e. sub components of the gesture are to be aligned with accented syllables.

## 5.7. The timing element

The timing element is used in order to specify both the duration of a movement as well as its temporal alignment in relation to other movements within the gesticonEntry. A minimum, maximum and default duration can be specified. The default duration specifies the normal duration of a movement. The minimum and maximum durations specify the lower and upper bound according to which a movement can be shortened or prolonged. Shortening and lengthening of gestures is important when speech and gestures are aligned. Durations are specified in milliseconds. Fixed="yes" indicates that the duration cannot be modified by the animation engine.

For the specification of temporal synchronization of the different movements in a gesticonEntry, concepts from W3C’s Synchronized Multimedia Integration Language SMIL (<http://www.w3.org/TR/SMIL2/smil-timing.html>) have been adopted.

The temporal relationship between different components of the form element is defined as follows:

The top-level elements (i.e. gesture, hand\_configuration, face, etc.) are to be interpreted as parallel in timing, i.e. implicitly they all start at the same time. On the other hand, movement elements contained in each of these sub-elements are ordered sequentially. I.e. the different elements are (implicitly) embedded in the following structure of SMIL’s par(allel) and seq(ue ntial) tags:

```

<form>
  <par>
    <gesture>
      <seq>
        <arm>
          ...
        <seq>
          <movement/>
          <movement/>
        </seq>
      </arm>
    </arm/>
    ...
  </seq>
</gesture>

```

```

<hand_configuration>
  <seq> ... </seq>
</hand_configuration>
<gaze <seq> ... </seq> </gaze>
<head <seq> ... </seq> </head>
<upper_ <seq> ... </seq> </upper_body>
<posture <seq> ... </seq> </posture>
</par>
</form>

```

Should it be necessary to deviate from this pre-defined internal temporal arrangement of movements, the insertion of explicit <par> and <seq> elements is foreseen. Also grouping sub-elements of a movement under a node point blocks the sequential interpretation in favour of a parallel one.

The timing element thus has the following structure:

```

<timing
  min="DURATION_IN_MS"
  max="DURATION_IN_MS"
  default="DURATION_IN_MS" fixed="STRING"
  begin="BEGIN_TIME"
  end="END_TIME"
/>

```

Attribute	Value(s)
min	DURATION_IN_MS
max	DURATION_IN_MS
default	DURATION_IN_MS
fixed	yes
begin	time_based or event based start time
end	time_based or event based start time

The attributes begin and end allow for the specification of additional constraints on timing. With SMIL's event based timing model, temporal relationship between arbitrary elements in the gesticonEntry can be specified.

In the following example `begin="gest1.begin"` ties the begin of movement `hand2` to the begin-event of the movement with the id "gest1". I.e. movement `hand2` is to be started at the same time as movement `gest1`. In addition to the direct reference to events, SMIL also allows for additional arithmetic modifications within the attribute. In the examples `end="gest1.end+1s"` specifies, that movement `hand2` is to end one second after `gest1`. This event-based timing can be extended to all kinds of events, i.e. it can be used for synchronizing movements to events issued outside the `gesticon` as well.

```
<form>
  <gesture>
    <movement id="gest1"/>
    ...
  </gesture>
  <hand_configuration>
    <hand_shape>
      <movement id="hand2">
        <timing begin="gest1.begin" end="gest1.end+1s"/>
      </movement>
    </hand_shape>
  </hand_configuration>
</form>
```

## 5.8. The arm element and its subelement point

In the point elements the form features of a movement are specified. In hand-arm gestures the point element comprises the specification for the hand and the position of the wrist according to the main body planes frontal, transversal and sagittal. The attributes `LocFrontal`, `LocTransversal` and `LocSagittal` and their values are taken from MURML. While the attributes referring to the main body planes are fixed, alternative values may be defined by individual users of the `Gesticon` format. In this case they must be specified in the XML-Schema.

```
<arm>
  <base_gesture>
    <movement>
      <timing/>
      <point id="ID" LocFrontal="STRING" LocTransversal=STRING
LocSagittal="STRING">
        <hand/>
      </point>
    </movement>
  </base_gesture>
</arm>
```

Attribute	Value(s)
id	ID
LocFrontal	LocAboveHead LocHead LocForehead LocEyes LocNose LocMouth LocChin

	LocLowerChin LocNeck LocShoulder LocUpperChest LocLowerChest LocStomach LocBelowStomach LocHip LocBelowHip
LocTransversal	LocCCenter LocBackof LocCCenterRight  LocCCenterLeft LocCenterRight LocCenterLeft LocPeripheryRight LocPeripheryLeft LocExtremePeripheryRight LocExtremePeripheryLeft LocLeft LocRight
LocSagittal	LocContact LocNear LocNorm LocFar LocStreched

## 5.9. The hand element: general structure

For the hand element again a subelement movement – comprising the subelements duration and point – is defined.

```
<hand id="ID" >
  <movement>
    <timing/>
    <point/>
  </movement>
</hand>
```

## 5.10. The hand element and its subelement point

```
<hand id="ID" >
  <movement>
    <timing/>
    <point palm_orientation="STRING" finger_direction=""STRING >
      <hand_configuration/>
    </point>
  </movement>
</hand>
```

Attribute	Value(s)
palm_orientation	PalmL PalmLU PalmLD PalmR PalmRU PalmRD PalmU PalmD PalmA PalmAL PalmAR PalmT PalmTL PalmTR PalmAU PalmAD PalmTU PalmTD
finger_direction	DirL DirLU DirLD DirR DirRU DirRD DirU DirD DirA DirAL DirAR DirAU DirAD DirT DirTL DirTR DirTU DirTD

The values for palm\_orientation and finger\_direction are taken from MURML (cf. ExtendedFingerOrientation and RelativeRotation respectively, in (Kopp & Wachsmuth, 2004). An alternative set is taken from Greta specifications, namely:

- Palm\_orientation = PalmDefault, PalmUp, PalmDown, PalmAway, PalmTowards, PalmInwards, PalmOutwards, PalmNone
- Finger\_orientation = FBDefault, FBUp, FBDown, FBAway, FBTowards, FBInwards, FBOutwards, FBNone

Within MURML the spatial directions are:

- U: Up; vertical axis
- D: Down; vertical axis
- L: Left; horizontal axis
- R: Right, horizontal axis
- T: Toward, depth axis
- A: Away, depth axis

Within Greta, the spatial directions are:

- Up, Down: along the vertical axis
- Away, Towards: along the depth axis (perpendicular to the body direction)
- Inward, Outward: along the horizontal axis; inward refers to ‘right palm (resp. left) oriented toward left (resp. right)’ and outward refers to ‘right (resp. left) palm oriented toward right (resp. left).’

Again the use of individually defined values is possible as long as they are specified in the XML-Schema.

## 5.11. The hand\_configuration element

The hand\_configuration element comprises of the subelements hand\_shape, thumb\_orientation, and finger. The static description is taken from MURML. For each of the subelements movement can be specified in order to allow for the representation of complex hand and finger movements.

```
<hand>
  <movement>
    <timing/>
    <point>
      <hand_configuration id="" xlink="" identifier="" >
        <hand_shape>
          <movement/>
        </hand_shape>
        <thumb_orientation>
          <movement/>
        </thumb_orientation>
        <finger>
          <movement/>
        </finger>
      </hand_configuration>
    </point>
  </movement>
</hand>
```

```

    </finger>
    ...
    <finger/>
  </hand_configuration>
</point>
</movement>
</hand>

```

Attribute	Value(s)
id	ID
xlink	(*)
identifier	STRING

(\*) Hand configurations are typically stored as separate Gesticon entries. Their representations may also be stored outside of the Gesticon in a player-specific format. Thus the Xlink mechanism is used.

## 5.12. The point element under hand\_shape

```

<hand_shape>
  <movement>
    <timing/>
    <point shape="form_fist"/>
  </movement>
</hand_shape>

```

Attribute	Value(s)
Shape	form_fist form_open form_point1 form_point2 form_2apart form_openapart symbol_1_open symbol_1_closed symbol_2_open symbol_2_closed symbol_3_open symbol_3_closed

The shape values represent HamNoSys' basic hand shapes. Figure 1 illustrates some of the hand shapes and related value labels. The labels presented are those used in the Greta system (cf Greta Gesture manual).



**Fig.1:** thumb away    thumb closed    thumb default

### 5.13. The thumb\_orientation element

```
<thumb_orientation type="">
  <movement>
    <timing/>
    <point thumb=""thumb_default/>
  </movement>
</thumb_orientation>
```

Attribute	Value(s)
thumb	thumb_default thumb_away thumb_over open_default open_thumbout open_thumbin closed_default closed_straight closed_inside closed_tight

### 5.14. The finger element

```
<finger type="">
  <movement>
    <timing/>
    <point bend=""/>
  </movement>
</finger>
```

Attribute	Value(s)
type	t,i,m,r,s
bend	straight curved ANGLE S

The attribute type specifies which finger is addressed (thumb, index, middle, ring, small finger). In the point element the finger bend is specified, i.e. whether the finger is curved or stretched. This can be realized via labels (straight, curved) or via the specification of angles. The choice which representation to employ is left user-defined.

### 5.15. Example of arm gestures

An example of the benefits of using this hierarchical representation is the differentiation of continuous and discontinuous movement. For instance, let's consider the gestures describing a square in space. In the continuous case the dynamism of the trajectories is constant and can be described by only one base gesture. Whereas in the discontinuous case several base gestures are specified for simulating an accentuated angular movement.

**Example of a continuous movement:**

```

<base_gesture id="ID" phase="stroke">
  <movement trajectory_type="linear" >
    <timing/>
    <point wrist_position=""/>
    <point wrist_position=""/>
    <point wrist_position=""/>
    <point wrist_position=""/>
  </movement>
</base_gesture>

```

### Example of a discontinuous movement:

```

<base_gesture id="ID" phase="stroke">
  <movement trajectory_type="linear" >
    <timing/>
    <point wrist_position=""/>
  </movement>
</base_gesture>
<base_gesture id="ID" phase="stroke">
  <movement trajectory_type="linear" >
    <timing/>
    <point wrist_position=""/>
  </movement>
</base_gesture>
<base_gesture id="ID" phase="stroke">
  <movement trajectory_type="linear" >
    <timing/>
    <point wrist_position=""/>
  </movement>
</base_gesture>
<base_gesture id="ID" phase="stroke">
  <movement trajectory_type="linear" >
    <timing/>
    <point wrist_position=""/>
  </movement>
</base_gesture>

```

## 5.16. Face entry

While it is quite common to describe hand-arm movements through a rather small set of parameters (Stokoe et al, 1976), the description of facial expression is a different case. Within behavior specification languages it is often the case that facial expression are described by a set of labels such as smile, raise eyebrow, open mouth etc. It is our point of view that such description counters encoding variability. While FACS does allow variability of surface expression to be described, it is not widely used in the graphics community. Facial expression in graphics models is commonly described by sets of low-level parameters, e.g. MPEG-4, or via muscle contraction. Thus to be independent of individual facial models, we propose to describe facial expression via sets of face elements, with each set being a placeholder for various model dependent facial descriptions.

In addition, a face element contains temporal information. This element is specified either as subelement of the face element when timing is equal to all facebases, or as subelement of

individual facebases when variable timing is required. This way the representations are able to account for the two major approaches to facial display of emotions:

1. the more static and traditional one, where emotion displays are switched on and off as a whole. The full expression of emotion appears simultaneously on the face. So far, most ECAs systems have implemented such an approach.
2. the more dynamic approach where emotion display is set on gradually. See for instance (Wehrle et al., 2000) and (Kaiser & Wehrle, 2006).

```
<face id="UID" identifier="" >
  <facebase id="ID" identifier="" factor="INTEGER" >
    <temporal_information>
      <attack>
        <timing/>
      </attack>
      <decay>
        <timing/>
      </decay>
      <sustain>
        <timing/>
      </sustain>
      <release>
        <timing/>
      </release>
    </temporal_information>
  </facebase>
</face >
```

The factor attribute is used in order to specify an intensity value for controlling the amount of muscular contraction.

Temporal\_information contains subelements representing the shape of the envelope of a signal. Commonly used temporal variations are attack-decay-sustain-release and onset-apex-offset. In other words, the temporal variation of a facial expression can be represented by

1. a trapezoid. After Paul Ekman (1989) the temporal phases are:
  - a. onset: time of appearance of an expression
  - b. apex: time an expression is maintained
  - c. offset: time of disappearance of an expression
2. ADSR that stands for:
  - a. attack: time to reach the expression of its maximal intensity
  - b. decay: time during which the intensity of the expression lightly decreases
  - c. sustain: time during which the expression is maintained
  - d. release: time the expression takes to disappear

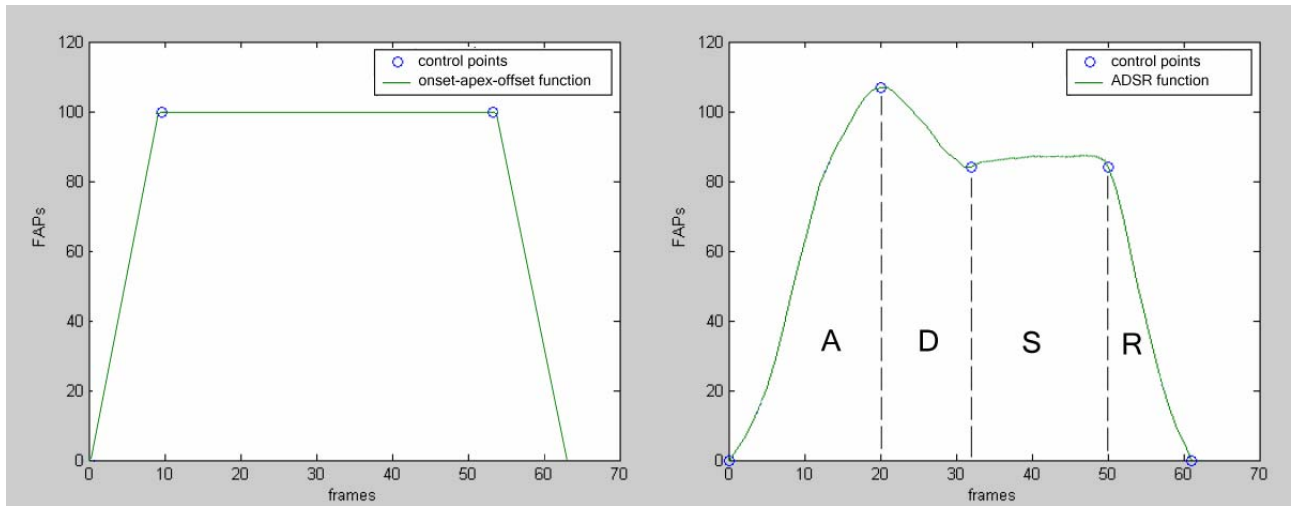


Fig: left: onset-apex-offset course; right: attach-delay-sustain-release course.

In addition to the core attributes (identifier and factor) of face\_base, we also provide a set of labels (attributes and values) for a high-level description of the individual face\_bases in order to make the otherwise opaque information in the face\_base. Our list of labels is one of many possible ways to separate the face into areas. It is meant as a guideline, but leaves users free to define their own labels if they wish to do so. Technically speaking the following attributes and values are optional. The following list of facial area is a refined composition from Ekman (Ekman, 1999). The list of values is taken from FACS, description of facial expression (Ekman and Friesen, 1978; Ekman et al., 2002). We do emphasize that the values in this list are open and can be changed. They are listed there as indicators.

Attribute	Value(s)
forehead_eyebrows	inner_raise (AU1), inner_raise_left (AU1L), inner_raise_right (AU1R), inner_raise (AU1), outer_raise_left (AU2L), outer_raise_right (AU2R), outer_raise (AU2), frown (AU4), raise (AU1+2), inner_raise_central (AU1+4), raise_central (AU1+2+4)
eye_lids	upper_lid_raiser (AU5), lower_lid_raiser (AU7), lid_droop (AU41), slit (AU42), eyes_closed (AU43), squint (AU44), blink (AU45), wink (AU46), wink_left (AU46L), wink_right (AU46R)
nose_cheeks	Cheek_raiser (AU6), nose_wrinkler (AU9), cheek_blow (AU33), cheek_puff (AU34), cheek_suck (AU35), nostril_dilator (AU38), nostril_compressor (AU39)
mouth	upper_lip_raiser (AU10), nasolabial_furrow_deepener (AU11), nasolabial_furrow_deepener_left (AU11L), nasolabial_furrow_deepener_right (AU11R), lip_corner_puller (AU12), lip_corner_puller_left (AU12L), lip_corner_puller_right (AU12R), sharp_lip_puller (AU13), sharp_lip_puller_left (AU13L), sharp_lip_puller_right (AU13R), dimpler (AU14), dimpler_left (AU14L), dimpler_right (AU14R), lip_corner_depressor (AU15), lip_corner_depressor_left (AU15L), lip_corner_depressor_right

	(AU15R), lower_lip_depressor (AU16), lip_pucker (AU18), lip_stretcher (AU20), lip_funneler (AU22), lip_tightener (AU23), lip_presser (AU24), lip_suck (AU28), lip_bite (AU32), smile (AU11+12)
Chin_jaw	chin_raiser (AU17), lip_parts (AU25), jaw_drop (AU26), mouth_stretch (AU27), jaw_thrust (AU29), jaw_sideways (AU30), jaw_clencher (AU31)
Tongue	Tongue_show (AU19), tongue_bulge (AU36), lip_wipe (AU37)

## EXAMPLES



Figure: Inner raise eyebrow AU1;



frown AU4

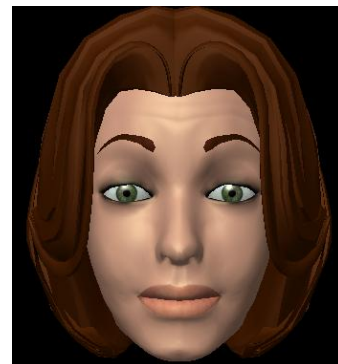
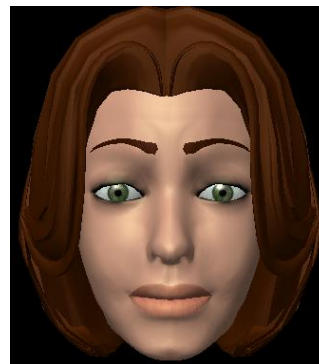


Figure: Raise eyebrow AU1+2;



Raise-Central eyebrow AU1+2+4

## 5.17. Gaze entry

Gaze is another example for a complex modality: comprising (i) only eye direction, (ii) neck, head and eyes showing one direction or (iii) neck, head and eyes showing individual directions. Other than in FACS or MPEG-4 where gaze direction is absolute (e.g. defined by angle values), we propose a mechanism for relative reference: gaze direction such as `look_at`, `look_away_from` are specified relative to a target, the value of which is generated outside of the Gesticon either in the real world or the virtual environment. To further specify `look_away_from` a modifier `gaze_dir` may be used, where a spatial region is defined such as `left`, `right`, `up_left`, etc.

```
<gaze id="" rhythm="" identifier=""
  head="STRING"
```

```

neck="STRING"
look_at="TARGET"
look_away_from="TARGET"
gaze_dir="STRING" />

```

Attribute	Value(s)
head	yes
neck	yes
gaze_dir	left right up down up_left up_right down_left down_right
look_at	TARGET
look_away_from	TARGET

The gaze mechanism is illustrated by the following examples:

1. `<gaze head="yes" neck="yes" look_at="TARGET" />`  
 Head and eye movement are in parallel (head="yes"), and the agent is looking at a non-moving target (look\_at="TARGET"). The head movement starts at the basis of the neck. As the actual value of the variable TARGET is generated outside of the Gesticon, thus some mechanism is required to bind the variable.
2. `<gaze head=yes neck=yes look_away_from="TARGET" gaze_dir="right" />`  
 Head and eye movement are in parallel, the agent is looking away from TARGET by directing the gaze to the right of the TARGET (gaze\_dir="right").

## 5.18. The head element

While the neck and head direction are specified via the gaze entry, head and neck movement are given by means of a movement trajectory and timing. A movement may start at the base of the head or of the neck. So head and neck movement have the same type of representation. Head and neck trajectory is specified along three dimensions (horizontal, vertical, tilt).

```

<head id="UID" rhythm="" identifier="" repetition="number_of_repetitions"/>
  <movement trajectory_type="STRING" >
    <timing/>
    <point id="ID" horizontal="STRING" vertical="STRING" tilt="STRING"/>
  </movement>
</head>

```

Attribute	Value(s)
trajectory_type	linear curve
horizontal	left left_central central right_central right
vertical	up up_central central down_central down

tilt	left left_central central right_central right
fixed	yes no

Example of a head nod:

```
<head id="UID" rhythm="repetition" identifier="" repetition="number_of_repetitions"/>
  <movement trajectory_type="linear" >
    <timing/>
    <point id="ID" vertical="up_central" />
  </movement>
  <movement trajectory_type="linear" >
    <timing/>
    <point id="ID" vertical="down_central" />
  </movement>
  <movement trajectory_type="linear" >
    <timing/>
    <point id="ID" vertical="central" />
  </movement>
</head>
```

## 5.19. The upper\_body element

The upper\_body element comprises the specification for the neck, shoulders, spine and global movements of the torso. The manner attribute of spine specifies the posture of the upper body: upright and collapsed are marked in comparison to neutral. Neutral should also be used to specify the default of a specific characters posture.

As the upper\_body element comprises movements which can well be executed independent from each other, the default sequential ordering of movements has to be overridden by inserting an explicit <par> element.

```
<upper_body id="UID" rhythm="STRING" identifier="STRING" />
  <par>
    <neck id="ID" rhythm="" identifier="" repetition="number_of_repetitions"/>
      <movement trajectory_type="STRING" >
        <timing/>
        <point id="ID" horizontal="STRING" vertical="STRING" tilt="STRING"/>
      </movement>
    </neck>
    <shoulder intensity="" type="STRING"/>
      <movement>
        <timing/>
        <point id="ID" intensity="STRING" dir="STRING"/>
      </movement>
    </shoulder>
    <spine manner="STRING" intensity="" />
  </torso>
  <movement trajectory_type="STRING" >
    <timing/>
```

```

    <point id="ID" forward_backward="STRING" left_right="STRING" tilt="STRING"/>
  </movement>
</torso>
<par>
</upper_body>

```

Attribute	Value(s)
type	left right both
dir	down up backward forward neutral
intensity	low high
manner	upright collapsed neutral
trajectory_type	linear curve
vertical	up up_central central down_central down
horizontal	left left_central central right_central right
tilt	left left_central central right_central right
leaning	forward backward

As already stated in section 5.18, a movement may start at the base of the head or of the neck. So head and neck movement have the same type of representation.

The type attribute of the shoulder element specifies which shoulder is to be animated.

## 5.20. The posture element

The posture element involves the body elements downward from the hip. Pelvis, hip, legs, including knee, toes and ankle, are characterized by movement trajectory and duration. The specification of legs is analogous to the arms.

```

<posture id="UID" identifier="STRING" type="STRING" />
  <leg id="ID" type="STRING" >
    <timing/>
    <ankle_rotation>
      <movement trajectory_type="" >
        <timing/>
        <point flexion="" twisting="" />
      </movement>
    </ankle_rotation>
    <toes>
      <movement trajectory_type="" >
        <timing/>

```

```

    <point toebend="" />
  </movement>
</toes>
<knee>
  <movement trajectory_type="" >
    <timing/>
    <point flexion="" twisting="" />
  </movement>
</knee>
</leg>
<hip>
  <movement trajectory_type="">
    <timing/>
    <point flexion="" twisting="" abduct="" />
  </movement>
</hip>
<pelvis>
  <movement trajectory_type="">
    <timing/>
    <point tilt="" twisting="" rolling="" />
  </movement>
</pelvis>
</posture>

```

Attribute	Value(s)
type (as attribute of posture)	stand sit crouch
type (as attribute of leg)	left right
ankle_rotation	analoguous to palm orientation
toe_direction	analoguous to finger_direction
trajectory_type	linear curve
flexion (~ x-axis)	DEGREE
tilt (~ y-axis)	DEGREE
twisting	DEGREE
rolling	DEGREE
abduct	DEGREE
toebend	neutral bend flexion

Note: for the time being we do not make any provisions for modeling individual toes.



## 6. Emotion Annotation and Representation Language

This report proposes a syntax for an XML-based language for representing and annotating emotions in technological contexts. In contrast to existing markup languages, where emotion is often represented in an ad-hoc way as part of a specific language, we propose a language aiming to be usable in a wide range of use cases, including corpus annotation as well as systems capable of recognising or generating emotions. We describe the scientific basis of our choice of emotion representations and the use case analysis through which we have determined the required expressive power of the language. We illustrate core properties of the proposed language using examples from various use case scenarios.

### 6.1. Introduction

Representing emotional states in technological environments is necessarily based on some representation format. Ideally, such an Emotion Annotation and Representation Language (EARL) should be standardised to allow for data exchange, re-use of resources, and to enable system components to work together smoothly.

As there is no agreed model of emotion, creating such a unified representation format is difficult. In addition, the requirements coming from different use cases vary considerably. We have nevertheless formulated a first suggestion, leaving much freedom to the user to “plug in” their preferred emotion representation. The possibility to map one representation to another will make the format usable in heterogeneous environments where no single emotion representation can be used.

### 6.2. Different Descriptive Schemes for Emotions

A unified theory or model of emotional states currently does not exist (Scherer et al., 2005). Out of the range of existing types of descriptions, we focus on three that may be relevant when annotating corpora, or that may be used in different components of an emotion-oriented technological system.

Categorical representations are the simplest and most wide-spread, using a word to describe an emotional state. Such category sets have been proposed on different grounds, including evolutionarily basic emotion categories (Ekman, 1999); most frequent everyday emotions (Douglas-Cowie et al., 2005); application-specific emotion sets (Steidl et al., 2005); or categories describing other affective states, such as moods or interpersonal stances (Scherer, 2000).

Dimensional descriptions capture essential properties of emotional states, such as arousal (active/passive) and valence (negative/positive) (Cowie et al., 2000). Emotion dimensions can be used to describe general emotional tendencies, including low-intensity emotions.

Appraisal representations (Ellsworth & Scherer, 2003) characterise emotional states in terms of the detailed evaluations of eliciting conditions, such as their familiarity, intrinsic pleasantness, or relevance to one’s goals. Such detail can be used to characterise the cause or object of an emotion as it arises from the context, or to predict emotions in AI systems (Krenn et al., 2002; Aylett, 2004).

### 6.3. Use Cases and Requirements for an Emotion Annotation and Representation Language

In order to ensure that the expressive power of the representation language will make it suitable for a broad range of future applications, the design process for EARL was initiated by performing a collection of use cases among members of HUMAINE. This list of use cases for emotional representations comprises i) manual annotation of emotional content of (multimodal) databases, ii) affect recognition systems and iii) affective generation systems such as speech synthesizers or embodied conversational agents (ECAs). On the basis of these use cases and the survey of theoretical models of emotions, a first list of requirements for EARL was compiled, which subsequently underwent discussion and refinement by a considerable number of HUMAINE participants.

Among the different use cases, the annotation of databases poses the most refined and extended list of requirements, which also covers the requirements raised in systems for recognition or generation.

In the simplest case, text is marked up with categorical labels only. More complex use cases comprise time-varying encoding of emotion dimensions (Cowie et al., 2000), independent annotation of multiple modalities, or the specification of relations between emotions occurring simultaneously (e.g. blending, masking) (Douglas-Cowie et al., 2005).

EARL is thus requested to provide means for encoding the following types of information.

*Emotion descriptor.* No single set of labels can be prescribed, because there is no agreement – neither in theory nor in application systems – on the types of emotion descriptors to use, and even less on the exact labels that should be used. EARL has to provide means for using different sets of categorical labels as well as emotion dimensions and appraisal-based descriptors of emotion.

*Intensity* of an emotion, to be expressed in terms of numeric values or discrete labels.

*Regulation types*, which encode a person's attempt to regulate the expression of her emotions (e.g., simulate, hide, amplify).

*Scope* of an emotion label, which should be definable by linking it to a time span, a media object, a bit of text, a certain modality etc.

*Combination* of multiple emotions appearing simultaneously. Both the co-occurrence of emotions as well as the type of *relation* between these emotions (e.g. dominant vs. secondary emotion, masking, blending) should be specified.

*Probability* expresses the labeller's degree of confidence with the emotion label provided.

In addition to these information types included in the list of requirements, a number of additional items were discussed. Roughly these can be grouped into information about the person (i.e. demographic data but also personality traits), the social environment (e.g., social register, intended audience), communicative goals, and physical environment (e.g. constraints on movements due to physical restrictions). Though the general usefulness of many of these information types is undisputed, they are intentionally not part of the currently proposed EARL specification. If needed, they have to be specified in domain-specific coding schemes that embed EARL. It was decided to draw the line rather strictly and concentrate on the encoding of emotions in the first place, in order to ensure a small but workable representation

core to start with. The main rationale to justify this restrictive approach was to first provide a simple language for encoding emotional states properly, and to leave out the factors that may have led to the actual expression of this state. Thus, EARL only encodes the fact that a person is, e.g., trying to hide certain feelings, but not the fact that this is due to a specific reason such as social context. Clearly, more discussion is needed to refine the limits of what should be part of EARL.

## 6.4. Proposed Realisation in XML

We propose an extendible, XML-based language to annotate and represent emotions, which can easily be integrated into other markup languages, which allows for the mapping between different emotion representations, and which can easily be adapted to specific applications.

Our proposal shares certain properties with existing languages such as APML (deCarolis et al., 2004), RRL (Krenn et al., 2002), and EmoTV coding scheme (Douglas-Cowie et al., 2005), but was re-designed from scratch to account for the requirements compiled from theory and use cases. We used XML Schema Definition (XSD) to specify the EARL grammar, which allows us to define abstract datatypes and extend or restrict these to specify a particular set of emotion categories, dimensions or appraisals.

The following sections will present some core features of the proposed language, using illustrations from various types of data annotation. A structured specification table is given in Appendix A; the actual XML Schema, formally defining the language, is included as Appendix B.

### 1.1.1 Simple emotions

In EARL, emotion tags can be simple or complex. A simple `<emotion>` uses attributes to specify the category, dimensions and/or appraisals of one emotional state. Emotion tags can enclose text, link to other XML nodes, or specify a time span using start and end times to define their scope.

One design principle for EARL was that **simple cases should look simple**. For example, annotating text with a simple “pleasure” emotion results in a simple structure:

```
<emotion category="pleasure">Hello!</emotion>
```

Annotating the facial expression in a picture file `face12.jpg` with the category “pleasure” is simply:

```
<emotion xlink:href="face12.jpg" category="pleasure"/>
```

This “stand-off” annotation, using a reference attribute, can be used to refer to external files or to XML nodes in the same or a different annotation document in order to define the scope of the represented emotion.

In uni-modal or multi-modal clips, such as speech or video recordings, a start and end time can be used to determine the scope:

```
<emotion start="0.4" end="1.3" category="pleasure"/>
```

Besides categories, it is also possible to describe a simple emotion using emotion dimensions or appraisals:

```
<emotion xlink:href="face12.jpg" arousal="-0.2" valence="0.5" power="0.2"/>
```

```
<emotion xlink:href="face12.jpg" suddenness="-0.8" intrinsic_pleasantness="0.7" goal_conduciveness="0.3"
  relevance_self_concerns="0.7"/>
```

EARL is designed to give users full control over the sets of categories, dimensions and/or appraisals to be used in a specific application or annotation context (see below).

Information can be added to describe various additional properties of the emotion: an emotion *intensity*; a *probability* value, which can be used to reflect the (human or machine) labeller's confidence in the emotion annotation; a number of *regulation* attributes, to indicate attempts to *suppress*, *amplify*, *attenuate* or *simulate* the expression of an emotion; and a *modality*, if the annotation is to be restricted to one modality.

For example, an annotation of a face showing obviously simulated pleasure of high intensity:

```
<emotion xlink:href="face12.jpg" category="pleasure" simulate="1.0" intensity="0.9"/>
```

In order to clarify that it is the face modality in which a pleasure emotion is detected with moderate probability, we can write:

```
<emotion xlink:href="face12.jpg" category="pleasure" modality="face" probability="0.5"/>
```

In combination, these attributes allow for a detailed description of individual emotions that do not vary in time.

## 1.1.2 Complex emotions

A `<complex-emotion>` describes one state composed of several aspects, for example because two emotions co-occur, or because of a regulation attempt, where one emotion is masked by the simulation of another one.

For example, to express that an expression could be either pleasure or friendliness, one could annotate:

```
<complex-emotion xlink:href="face12.jpg">
  <emotion category="pleasure" probability="0.5"/>
  <emotion category="friendliness" probability="0.5"/>
</complex-emotion>
```

The co-occurrence of a major emotion of “pleasure” with a minor emotion of “worry” can be represented as follows.

```
<complex-emotion xlink:href="face12.jpg">
  <emotion category="pleasure" intensity="0.7"/>
  <emotion category="worry" intensity="0.5"/>
</complex-emotion>
```

Simulated pleasure masking suppressed annoyance would be represented:

```
<complex-emotion xlink:href="face12.jpg">
  <emotion category="pleasure" simulate="0.8"/>
  <emotion category="annoyance" suppress="0.5"/>
</complex-emotion>
```

The numeric values for “simulate” and “suppress” indicate the amount of regulation going on, on a scale from 0 (no regulation) to 1 (strongest regulation possible). The above example corresponds to strong indications of simulation while the suppression is only halfway successful.

If different emotions are to be annotated for different modalities in a multi-modal clip, there are in principle two choices. On the one hand, it is possible to describe them as different aspects of one complex emotion, and thus share the same scope:

```

<complex-emotion xlink:href="clip23.avi">
  <emotion category="pleasure" modality="face"/>
  <emotion category="worry" modality="voice"/>
</complex-emotion>

```

When scope is defined in terms of start and end times, it seems unlikely that different emotional expressions start and end at exactly the same moments in different modalities.

Such expressions in the different modalities can be described as separate events, each with their own temporal scope:

```

<emotion start="0" end="1.9" category="pleasure" modality="face"/>
<emotion start="0.4" end="1.3" category="worry" modality="voice"/>

```

Both options for modality-specific annotation co-exist, and it remains to be seen which is more useful.

It may be important to note explicitly that complex patterns of overlapping emotions can easily be represented using multiple, independent `<emotion>` or `<complex-emotion>` tags. Consider for example the case that during an expression of pleasure, a complex expression that could be worry or boredom appears, they co-exist for some time, and after the pleasure expression ends, the worry-or-boredom expression continues:

```

<emotion start="0" end="1.9" category="pleasure"/>
<complex-emotion start="1.4" end="3.3">
  <emotion category="worry" probability="0.5"/>
  <emotion category="boredom" probability="0.5"/>
</complex-emotion>

```

Insofar, the constraint of a `<complex-emotion>` referring to a single scope is not a limitation in expressivity, but simply means that several elements need to be used if more complex patterns are to be described.

### 1.1.3 Annotating time-varying signals

Two modes are previewed for describing emotions that vary over time. They correspond to types of annotation tools used for labelling emotional database. The Anvil (Kipp, 2004) approach consists in assigning a (possibly complex) label to a time span in which a property is conceptualised as constant. This can be described with the start and end attributes presented above.

The Feeltrace (Cowie et al., 2000) approach consists in tracing a small number of dimensions continuously over time. In EARL, we propose to specify such time-varying attributes using embedded `<samples>` tags.

For example, a curve annotated with Feeltrace describing a shift from a neutral state to an active negative state would be realised using two `<samples>` elements, one for each dimension:

```

<emotion start="2" end="2.7">
  <samples value="arousal" rate="10">
    0 .1 .25 .4 .55 .6 .65 .66
  </samples>
  <samples value="valence" rate="10">
    0 -.1 -.2 -.25 -.3 -.4 -.4 -.45
  </samples>
</emotion>

```

The output of more recent descendents of Feeltrace, which can be used to annotate various regulations or appraisals, can be represented in the same way. A sudden drop in the appraisal “consonant with expectation” can be described:

```
<emotion start="2" end="2.7">
  <samples value="consonant_with_expectation" rate="10">
    .9 .9 .7 .4 .1 -.3 -.7 -.75
  </samples>
</emotion>
```

An emotion of anger being increasingly suppressed over time can similarly be described:

```
<emotion start="2" end="2.7" category="anger">
  <samples value="suppress" rate="10">
    .1 .2 .3 .4 .4 .5 .6 .6
  </samples>
</emotion>
```

This relatively simple set of XML elements addresses many of the collected requirements.

## 6.5. A family of EARL Dialects: XML Schema design

Our suggested solution to the dilemma that no agreed emotion representation exists is to clearly separate the definition of an EARL document's structure from the concrete emotion labels allowed, in a modular design. Each concrete EARL dialect is defined by combining a base XML schema, which defines the structure, and three XML schema "plugins", containing the definitions for the sets of emotion categories, dimensions and appraisal tags, respectively. Different alternatives for each of these plugins exist, defining different sets of category labels, dimensions and appraisals. Figure 1 illustrates the idea.

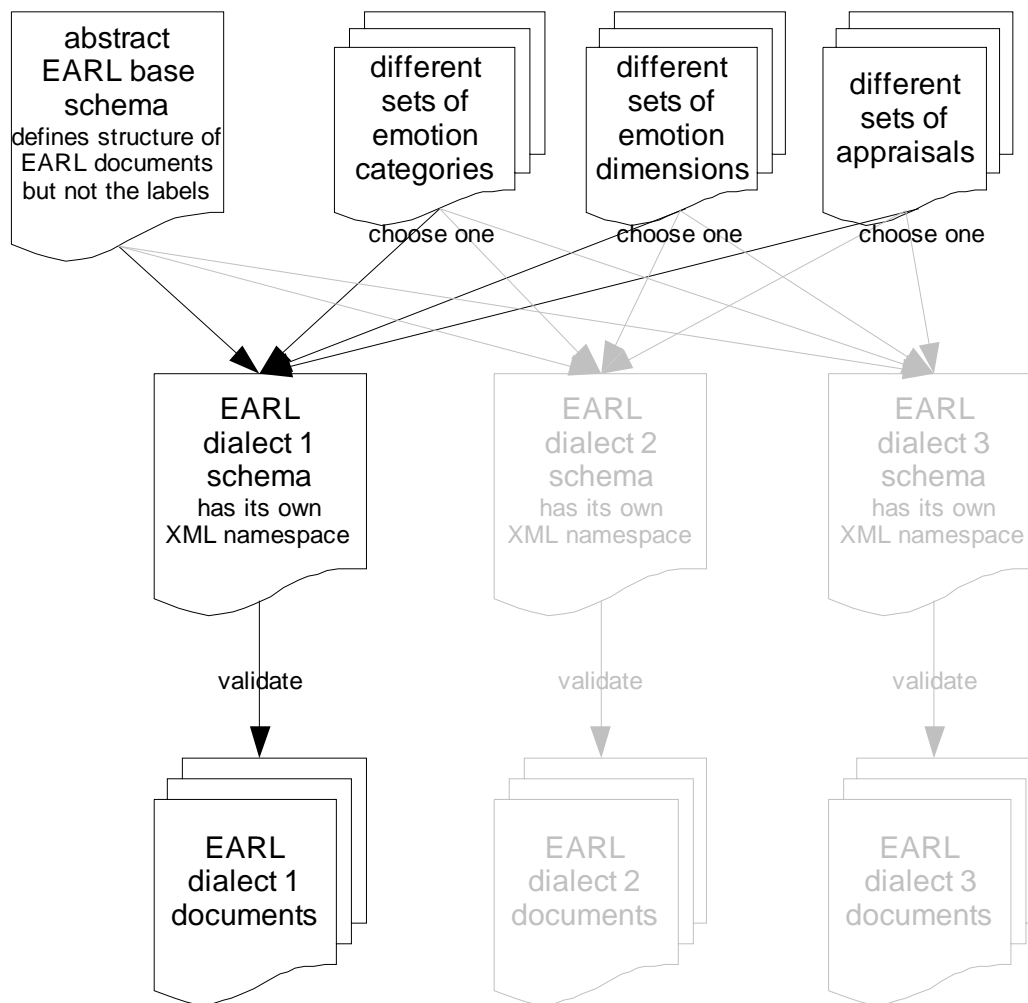


Figure 1: Illustration of how different sets of emotion descriptors are combined to form a concrete EARL dialect

For example, to allow emotions to be described by a core set of 27 categories describing everyday emotions in combination with two emotion dimensions, the EARL dialect would combine the base schema with the corresponding plugins for the 27 categories and the two dimensions, and the “empty set” plugin for appraisals. Another EARL dialect, describing emotions in terms of four application-specific categories, would combine the base schema with an application-specific category plugin and two “empty set” plugins for dimensions and appraisals.

Even though EARL will provide users with the freedom to define their own emotion descriptor plugins, a default set of categories, dimensions and appraisals will be proposed, which can be used if there are no strong reasons for doing otherwise.

## 6.6. Inventories of categories, dimensions and appraisals

Even though the EARL design is “pluggable” in the sense just described, some degree of normation should be attempted by proposing inventories of categories, dimensions and

appraisals that can be used by default, i.e. when no application-specific requirements demand different inventories.

Such default inventories should be in line with contemporary theory as much as possible; work in HUMAINE WP3 appears most promising in view of emotion-oriented computing.

### 1.1.4 Categories

Douglas-Cowie, Cox et al. (see HUMAINE deliverable D5f) propose a list of 48 emotion categories consolidated from various sources. The table is reproduced as Table 1, and the 48 categories are proposed as the default category set for EARL.

<b>Negative &amp; forceful</b> Anger Annoyance Contempt Disgust Irritation	<b>Positive &amp; lively</b> Amusement Delight Elation Excitement Happiness Joy Pleasure
<b>Negative &amp; not in control</b> Anxiety Embarrassment Fear Helplessness Powerlessness Worry	<b>Caring</b> Affection Empathy Friendliness Love
<b>Negative thoughts</b> Doubt Envy Frustration Guilt Shame	<b>Positive thoughts</b> Courage Hope Pride Satisfaction Trust
<b>Negative &amp; passive</b> Boredom Despair Disappointment Hurt Sadness	<b>Quiet positive</b> Calm Content Relaxed Relieved Serene
<b>Agitation</b> Shock Stress Tension	<b>Reactive</b> Interest Politeness Surprise

Table 1: Consolidated list of emotion categories as proposed in HUMAINE deliverable D5f.

### 1.1.5 Dimensions

Typically, when emotions are described in terms of emotion dimensions, the literature has most frequently found two or three dimensions of decreasing relevance in terms of the proportion of, e.g., similarity ratings explained (see e.g. (Schröder, 2004) for an overview).

The most important dimension is related to a valenced evaluation in terms of positive vs. negative, pleasurable vs. unpleasurable. The second dimension is usually related to the overall state of activity or arousal, from active to passive. The third dimension, which is slightly less frequently used, is related to the degree of control or social power that an individual has in a situation, i.e. high vs. low control or, when the focus is on the social relationship, dominant vs. submissive.

Naming conventions vary. We suggest using the following names, for reasons of clarity, limited ambiguity and wide-spread use.

arousal  
valence  
power

### **1.1.6 Appraisals**

The following list is a flattened representation tentatively formulated based on Klaus Scherer's work (Scherer, 1984). Feedback from users is appreciated to determine the use of this list, and possible needs for adjustment.

suddenness  
familiarity  
predictability  
intrinsic\_pleasantness  
relevance\_self\_concerns  
relevance\_relationship\_concerns  
relevance\_social\_order\_concerns  
goal\_outcome\_probability  
consonant\_with\_expectation  
goal\_conduciveness  
goal\_urgency  
cause\_agent\_self  
cause\_agent\_other  
cause\_motive\_intentional  
event\_controllability  
agent\_power  
goal\_adjustment\_possible  
standards\_compatibility\_external  
standards\_compatibility\_internal

## 6.7. Examples of applying EARL in different use cases

In the following, a number of different use cases are mentioned as examples for the use of EARL representations. These use cases are illustrations from various potential application areas, and are not intended in any way to be exhaustive. Other use cases certainly exist, and in the use cases mentioned, many more uses of EARL can be conceived.

### 1.1.7 Use case 0: Annotating text

Text can be annotated in three ways:

a) By directly enclosing the text:

```
<text xmlns:earl="http://emotion-research.net/earl/040/default">
This is a text with some
<earl:emotion category="excitement" intensity="0.3">quite exciting
</earl:emotion>
examples.
</text>
```

b) By referring to an element in the local document:

```
<text xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:earl="http://emotion-research.net/earl/040/default">
<earl:emotion category="excitement" intensity="0.3" xlink:href="#sent1"/>
<sentence id="sent1">
This is a text with some quite exciting examples.
</sentence>
</text>
```

c) Stand-off annotation, referring to a different document:

text34.xml:

```
<text>
<sentence id="sent1">
This is a text with some quite exciting examples.
</sentence>
</text>
```

emotion34.xml:

```
<earl xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://emotion-
research.net/earl/040/default">
<emotion category="excitement" intensity="0.3"
xlink:href="text34.xml#sent1"/>
</earl>
```

### 1.1.8 Use case 1a: Annotating multi-level audio-visual databases with ANVIL

For labelling time stretches with global or modality-specific emotion, a number of setups differing in complexity can be distinguished. For readability, top-level <earl> element and namespace definition are omitted here where these are not ambiguous.

#### a) Simplest case: One global label

```
<emotion category="pleasure" start="0.5" end="1.02"/>
```

#### b) Global and modality-specific labelling

The global label is complemented with, or replaced by, one or several modality-specific labels:

```
<emotion category="pleasure" probability="0.4" start="0.5" end="1.02"/>
<emotion modality="voice" category="pleasure" probability="0.9" start="0.5"
end="1.02"/>
<emotion modality="face" category="neutral" probability="0.5" start="0"
end="2"/>
<emotion modality="text" probability="0.4" start="0.5" end="1.02"
arousal="-0.5" valence="0.1"/>
```

#### c) Two labels (co-occurring emotions)

A "major" emotion can be annotated using a higher intensity than a "minor" emotion.

```
<complex-emotion start="0.5" end="1.02">
  <emotion category="pleasure" intensity="0.7"/>
  <emotion category="worry" intensity="0.5"/>
</complex-emotion>
```

#### d) One emotion simulated, the other suppressed

```
<complex-emotion start="0.5" end="1.02">
  <emotion category="pleasure" simulate="1.0"/>
  <emotion category="worry" suppress="1.0"/>
</complex-emotion>
```

### 1.1.9 Use case 1b: Continuous annotation of emotional tone with Feeltrace

A series of sample points is represented, usually as a global assessment:

```
<emotion start="0.387" end="0.416">
  <samples rate="333" values="arousal">
    -0.00256 -0.00256 -0.00256 -0.00256 -0.00256 -0.00256
    -0.00256 -0.00256 -0.00256 -0.00256
  </samples>
  <samples rate="333" values="valence">
    0.023 0.023 0.0281 0.0281 0.0281 0.0332 0.0332 0.0383 0.0383 0.0383
  </samples>
</emotion>
```

### 1.1.10 Use case 2a: Emotion recognition/classification of multi-modal input

#### a) annotation directly with EARL

Let us assume that an emotion recognition algorithm analyses a given multimodal video clip "clip123.avi". The algorithm analyses individual modalities separately and provides probabilities for different emotion classes. Let us assume for example that the set of classes used in the classifier is: "positive", "neutral", "negative", defined in an EARL dialect with its own namespace, e.g. <http://emotion-research.net/earl/040/posneuneg>.

Probabilities are encoded in the "probability" attribute.

Then we would have, for example:

```
<complex-emotion xmlns="http://emotion-research.net/earl/040/posneuneg"
  xlink:href="clip123.avi">
  <complex-emotion modality="biosignal">
    <emotion category="positive" probability="0.1"/>
    <emotion category="neutral" probability="0.4"/>
    <emotion category="negative" probability="0.05"/>
  </complex-emotion>
  <complex-emotion modality="face">
    <emotion category="positive" probability="0.6"/>
    <emotion category="neutral" probability="0.1"/>
    <emotion category="negative" probability="0.3"/>
  </complex-emotion>
  <complex-emotion modality="voice">
    <emotion category="positive" probability="0.1"/>
    <emotion category="neutral" probability="0.3"/>
    <emotion category="negative" probability="0.1"/>
  </complex-emotion>
</complex-emotion>
```

In this example, we are trying to recognise from biosignal, face and voice; there is relatively clear evidence in favor of an emotion from the face, probably a positive one; and inconclusive evidence from the biosignal and the voice. In a further "merging" step, these individual results could then be combined to, assuming global averaging:

```
<complex-emotion xmlns="http://emotion-research.net/earl/040/posneuneg"
  xlink:href="clip123.avi">
  <emotion category="positive" probability="0.267"/>
  <emotion category="neutral" probability="0.267"/>
  <emotion category="negative" probability="0.15"/>
</complex-emotion>
```

So the emotion is either neutral or positive.

## b) Using EARL within an EMMA document

EMMA is a W3C working draft for an "Extensible MultiModal Annotation markup language", <http://www.w3.org/TR/emma>. The general purpose of EMMA is to represent information automatically extracted from a user's input by an interpretation component.

EARL can easily be integrated into EMMA via EMMA's extensible content representation mechanism.

```
<emma:emma version="1.0"
  xmlns:emma:="http://www.w3.org/2003/04/emma"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:earl="http://emotion-research.net/earl/040/posneuneg">
  <emma:interpretation id="int1">
    <earl:complex-emotion>
      <earl:complex-emotion modality="biosignal">
        <earl:emotion category="positive" probability="0.1"/>
        <earl:emotion category="neutral" probability="0.4"/>
        <earl:emotion category="negative" probability="0.05"/>
      </earl:complex-emotion>
      <earl:complex-emotion modality="face">
        <earl:emotion category="positive" probability="0.6"/>
        <earl:emotion category="neutral" probability="0.1"/>
        <earl:emotion category="negative" probability="0.3"/>
      </earl:complex-emotion>
      <earl:complex-emotion modality="voice">
        <earl:emotion category="positive" probability="0.1"/>
        <earl:emotion category="neutral" probability="0.3"/>
        <earl:emotion category="negative" probability="0.1"/>
      </earl:complex-emotion>
    </earl:complex-emotion>
  </emma:interpretation>
</emma:emma>
```

The result of the "merging" step could also be represented using EARL within EMMA, e.g. as follows:

```
<emma:emma version="1.0"
  xmlns:emma:="http://www.w3.org/2003/04/emma"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:earl="http://emotion-research.net/earl/040/posneuneg">
  <emma:one-of id="r1">
    <emma:interpretation id="int1" probability="0.267">
      <earl:emotion category="positive"/>
    </emma:interpretation>
    <emma:interpretation id="int2" probability="0.267">
      <earl:emotion category="neutral"/>
    </emma:interpretation>
    <emma:interpretation id="int3" probability="0.15">
      <earl:emotion category="negative"/>
    </emma:interpretation>
  </emma:one-of>
</emma:emma>
```

### 1.1.11 Use case 2b: Emotion generation in ECAs

In the generation of an ECA dialogue act, first an appraisal component predicts a suitable emotion for the expression to be generated. The emotion is *about* something, which is

encoded in some semantic representation of the world.

In this example, we specify the emotion in a modified version of the rich representation language (RRL) defined in the NECA project, and the emotion refers to an object from the semantic representation based on DRS (discourse representation structure).

Situation: The simulated salesperson tells the customer that the car is environmentally friendly. When predicting the customer's emotional response, the customer appraisal model evaluates this as an unexpected but very pleasant news caused by the other agent.

```
<rml xmlns:earl="http://emotion-research.net/earl/040/default">
...
<dialogueAct id="d_5">
  <semanticContent>
    <unaryCond id="c_11" arg="car_1" pred="nonpolluting"/>
    ...
  </semanticContent>
  ...
</dialogueAct>
<dialogueAct id="d_6">
  ...
  <earl:emotion predictability="0.1"
    intrinsic_pleasantness="0.9" cause_agent_other="0.9"/>
  ...
</dialogueAct>
...
</rml>
```

After text generation, the emotion would be applied to the generated sentence. In addition, let us assume that a mapping to a set of emotion categories and dimensions occurs:

```
<rml xmlns:earl="http://emotion-research.net/earl/040/default">
...
<dialogueAct id="d_6">
  ...
  <sentence id="s_6">Now that's nice to hear!</sentence>
  <earl:emotion xlink:href="#s_6" category="delight" intensity="0.9"
    predictability="0.1" intrinsic_pleasantness="0.9"
    cause_agent_other="0.9" arousal="0.7" valence="0.8"/>
</dialogueAct>
...
</rml>
```

## 6.8. Mapping emotion representations

The reason why EARL previews the use of different emotion representations is that no preferred representation has yet emerged for all types of use. Instead, the most profitable representation to use depends on the application. Still, it may be necessary to convert between different emotion representations, e.g. to enable components in a multi-modal generation system to work together even though they use different emotion representations (Krenn et al., 2002).

For that reason, EARL will be complemented with a mechanism for mapping between emotion representations. From a scientific point of view, it will not always be possible to define such mappings. For example, the mapping between categories and dimensions will only work in one direction. Emotion categories, understood as short labels for complex states, can be located on emotion dimensions representing core properties; but a position in emotion dimension space is ambiguous with respect to many of the specific properties of emotion categories, and can thus only be mapped to generic super-categories. Guidelines for defining scientifically meaningful mappings will be provided.

## 6.9. Outlook

We have presented the expressive power of the EARL specification as it is currently conceived. Some specifications are still suboptimal, such as the representation of the start and end times. Other aspects may be missing but will be required by users, such as the annotation of the object of an emotion or the situational context. The current design choices can be questioned, e.g. more clarity could be gained by replacing the current flat list of attributes for categories, dimensions and appraisals with a substructure of elements. On the other hand, this would increase the annotation overhead, especially for simple annotations, which in practice may be the most frequently used. An iterative procedure of comment and improvement is needed before this language is likely to stabilise into a form suitable for a broad range of applications.

We are investigating opportunities for promoting the standardisation of the EARL as a recommended representation format for emotional states in technological applications.

## 7. REFERENCES

- Abrilian, S., Martin, J.-C. & Buisine, S. (2003). Algorithms for controlling cooperation between output modalities in 2D Embodied Conversational Agents. Proceedings of the Fifth International Conference on Multimodal Interfaces (ICMI'2003), Vancouver, British Columbia, Canada, November 5-7, ACM Press, 293-296.
- Allbeck, J., Badler, N. (2002). Toward Representing Agent Behaviors Modified by Personality and Emotion, "Workshop Embodied conversational agents - let's specify and evaluate them!" at AAMAS 2002, Bologna, Italy.
- Arafa, Y., Kamyab, K., Kshirsagar, S., Guye-Vuilleme, A., Thalmann, N. (2002). "Two Approaches to Scripting Character Animation", Proc. of the AAMAS Workshop on "Embodied conversational agents -- Let's specify and evaluate them!" Bologna.
- Aylett, R.S. (2004). Agents and affect: why embodied agents need affective systems Invited paper, 3rd Hellenic Conference on AI, Samos, May 2004 Springer Verlag LNAI 3025 pp496-504.
- Beard S., Reid D. (2002). MetaFace and VHML: A First Implementation of the Virtual Human Markup Language, in Marriott A. et al. (eds.), Embodied Conversational Agents: Let's Specify and Compare Them!, Workshop Notes, Autonomous Agents & Multiagent Systems 2002, University of Bologna, Bologna, Italy.
- Cassell, J., Vilhjálmsson, H., and Bickmore T. (2001). BEAT : the Behavior Expression Animation Toolkit, Proc. ACM SIGGRAPH 2001, Los Angeles, August 12-17, 477-486.
- Cowie, R., Douglas-Cowie, E., Savvidou, S., McMahon, E., Sawey, M., & Schröder, M. (2000). 'FEELTRACE': An instrument for recording perceived emotion in real time, *ISCA Workshop on Speech and Emotion, Northern Ireland* , p. 19-24.
- DeCarolis B., Pelachaud C., Poggi I, and Steedman M. (2004). APMML, a mark-up language for believable behavior generation. In H. Prendinger and M. Ishizuka, editors, *Life-like Characters. Tools, Affective Functions and Applications*, 65--85. Springer.
- Doenges, P., Lavagetto, F., Ostermann, J., Pandzic, I.S., and Petajan, E.. (1997). "MPEG-4: Audio/video and synthetic graphics/audio for mixed media." *Image Communications Journal*, 5(4).
- Douglas-Cowie, E., L. Devillers, J-C. Martin, R. Cowie, S. Savvidou, S. Abrilian, and C. Cox (2005). Multimodal Databases of Everyday Emotion: Facing up to Complexity. In *Proc. InterSpeech*, Lisbon, September 2005.
- Ekman P., Friesen W.: *Facial Action Coding System*, Consulting Psychologist Press, Palo Alto, CA, 1978.
- Ekman, P. (1989), "The argument and evidence about universals in facial expressions of emotion", in H. Wagner and A. Manstead, eds., *Handbook of Social Psychophysiology*, Wiley, Chichester, New-York.
- Ekman, P. (1999). Basic emotions. In Tim Dalgleish and Mick J. Power (Ed.), *Handbook of Cognition & Emotion* (pp. 301–320). New York: John Wiley.

Ekman, P., Friesen, W.V., & Hager, J.C. (2002). THE FACIAL ACTION CODING SYSTEM Second edition. Salt Lake City: Research Nexus eBook. London: Weidenfeld & Nicolson.

Elliott, R. and Glauert, J.R.W. and Jennings, V. and Kennaway, J.R.. (2004). "An Overview of the SiGML Notation and SiGML Signing Software System", In Fourth International Conference on Language Resources and Evaluation, LREC 2004, Edited by Streiter, O. and Vettori, C., Lisbon, Portugal, pp. 98-104,.

Ellsworth, P.C., & Scherer, K. (2003). Appraisal processes in emotion. In Davidson R.J. et al. (Ed.), *Handbook of Affective Sciences* (pp. 572-595). Oxford New York: Oxford University Press.

Hartmann, B., Mancini, M., and Pelachaud, C. (2002). Formational parameters and adaptive prototype instantiation for MPEG-4 compliant gesture synthesis. In *Computer Animation'02*, Geneva, Switzerland. IEEE Computer Society Press.

Kaiser, S. and Wehrle, T. (2006). Modeling appraisal theory of emotion and facial expression. In N. Magnenat-Thalmann, D. Pai, A. Paiva, & E Wu (Eds.), *Proceedings of the 19th International Conference on Computer Animation and Social Agents (CASA 2006)* (pp. 121-130). Computer Graphics Society (CGS).

Kipp, M. (2004). *Gesture Generation by Imitation - From Human Behavior to Computer Character Animation*. Boca Raton, Florida: Dissertation.com.

Kopp S., Jung, B., Lessmann, N., Wachsmuth, I. (2003) Max--A Multimodal Assistant in Virtual Reality Construction. KI 4/03: 11-17.

Kopp S., Wachsmuth I. (2004): Synthesizing Multimodal Utterances for Conversational Agents. *Computer Animation and Virtual Worlds*, 15(1): 39-52.

Kransted A., Kopp S., Wachsmuth I. (2002). MURML: A Multimodal Utterance Representation Markup Language for Conversational Agents, in Marriott A. et al. (eds.), *Embodied Conversational Agents: Let's Specify and Compare Them! Workshop Notes, Autonomous Agents & Multiagent Systems 2002*, University of Bologna, Bologna, Italy.

Krenn B. (2005). Representational Lego for ECAs, Background paper for a presentation held at the FP6 NoE HUMAINE Workshop on Emotion and Interaction. Paris, 10-11 March.

Krenn B., Pirker H. (2004). Defining the Gesticon: Language and Gesture Coordination for Interacting Embodied Agents, in Proc. of the AISB-2004 Symposium on Language, Speech and Gesture for Expressive Characters, University of Leeds, UK, 107-115.

Krenn, B., Pirker, H., Grice, M., Piwek, P., Deemter, K.v., Schröder, M., Klesen, M., & Gstrein, E. (2002). Generation of multimodal dialogue for net environments. *Proceedings of Konvens*. Saarbrücken, Germany.

Martell, C. (2002). FORM: An Extensible, Kinematically-based Gesture Annotation Scheme in *Proceedings of the 2002 International Conference on Language Resources and Evaluation*, Las Palmas, Canary Island.

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

Mori K., Jatowt A. and Ishizuka M. (2003). Enhancing Conversational Flexibility in Multimodal Interactions with Embodied Lifelike Agents, Proc. Int'l Conf. on Intelligent User Interfaces (IUI 2003) (ACM), pp. 270--272, Miami, Florida, USA

Noot H., Ruttkay Z. (2003). GESTYLE: Doing the same in different styles, in Rist T. et al. (eds.), *Intelligent Virtual Agents, 4th International Workshop (IVA 2003)*, Sept. 15-17, Kloster Irrsee, Springer, Berlin/Heidelberg/New York.

Ostermann J., (1998). Animation of synthetic faces in MPEG-4. In *Computer Animation'98*, pages 49–51, Philadelphia, USA, June

Pandzic, I.S and Forchheimer, R. (editors). 2002. *MPEG-4 Facial Animation - The standard, implementations and applications*, pp. 291-296, John Wiley & Sons.

Paradiso, A. and L'Abbate, M. (2001). A Model for the Generation and Combination of Emotional Expressions In: Pelachaud, C., Poggi, I. (Eds.) *Multimodal Communication and Context in Embodied Agents Workshop Proceedings of the fifth International Conference on Autonomous Agents (AA'01)*, May 2001, Montreal, Canada (pp. 65 - 70) Montreal: ACM Press.

Piwek P., Krenn B., Schröder M., Grice M., Baumann S., Pirker H. (2002). RRL: A Rich Representation Language for the Description of Agent Behaviour in NECA, In *Proceedings of the Workshop Embodied conversational agents - let's specify and evaluate them!*, held in conjunction with AAMAS-02, July 16 2002, Bologna, Italy.

Poggi, C. Pelachaud, and F. de Rosis. (2000). "Eye communication in a conversational 3D synthetic agent." *Special Issue on Behavior Planning for Life-Like Characters and Avatars of AI Communications*.

Poggi, I. (2002) Symbolic gestures. The case of the Italian gestuary. *Gesture*, 2(1):71–98.

Posner, R., Kruger, R., Noll, T., and Serenari, M. (2002). *The Berlin Dictionary of Everyday Gestures*. Version 9 2002. Technical report, Research Center for Semiotics, TU Berlin.

Prendinger H., Descamps S., Ishizuka M. (2004). MPML: A markup language for controlling the behavior of life-like characters, *Journal of Visual Languages and Computing*, 15(2):183-203.

Prillwitz S., Leven R., Zienert H., Hanke T., and Henning J. (1989). HamNoSys. Version 2.0: Hamburg notation system for sign languages: An introductory guide. In *International Studies on Sign Language and Communication of the Deaf*, volume 5. Signum Press, Hamburg, Germany.

Ruttkay, Z., Pelachaud, C. (2000). Exercises of style for virtual humans, *Animating Expressive Characters for Social Interactions*, Symposium of the AISB'02 Convention, Londres, April 2002.

Zs. Ruttkay, V. van Moppes, and H. Noot. The jovial, the reserved and the robot. In proceedings of the AAMAS03 Ws on Embodied Conversational Characters as Individuals, Melbourne, Australia, 2003

Serenari, M., Dybkjaer, L., Heid, U., Kipp, M., Reithinger, N.(2000). Survey of existing gesture, facial expression, and cross-modality coding schemes. IST-2000-26095 Deliverable D2.1, Project NITE.

Scherer, K.R. (1984). On the nature and function of emotion: a component process approach. In Klaus R. Scherer and Paul Ekman (Ed.), *Approaches to emotion* (pp. 293–317). Hillsdale, NJ: Erlbaum.

Scherer, K.R. (2000). Psychological models of emotion. In J. C. Borod (Ed.), *The Neuropsychology of Emotion* (pp. 137–162). New York: Oxford University Press.

Scherer, K. et al. (2005). Proposal for exemplars and work towards them: Theory of emotions. HUMAINE deliverable D3e, <http://emotion-research.net/deliverabl>

Schröder, M. (2004). Speech and emotion research: an overview of research frameworks and a dimensional approach to emotional speech synthesis (Ph.D thesis). Vol. 7 of Phonus, Research Report of the Institute of Phonetics, Saarland University. Online at: <http://www.dfki.de/~schroed>.

Steidl, S., Levit, M., Batliner, A., Nöth, E., & Niemann, H. (2005). "Of all things the measure is man" - automatic classification of emotions and inter-labeler consistency. *ICASSP 2005, International Conference on Acoustics, Speech, and Signal Processing, March 19-23, 2005, Philadelphia, U.S.A., Proceedings* (pp. 317--320).

Stokoe W. C., Casterline D. C. and Croneberg C. G. (1976). A dictionary of American sign language on linguistic principles. Linstok Press

Tsutsui, T., Saeyor, S. and Ishizuka, M. (2000). MPML: A Multimodal Presentation Markup Language with Character Agent Control Functions, Proc.(CD-ROM) WebNet World Conf. on the WWW and Internet, San Antonio, Texas, USA, (2000.10.30-2000.11.4) [available online at <http://www.miv.t.u-tokyo.ac.jp/papers/santiWebNet2000.pdf> ]

Wehrle, T., Kaiser, S., Schmidt, S. & Scherer, K. R (2000). Studying the dynamics of emotional expression using synthesized facial muscle movements. *Journal of Personality and Social Psychology*, 78 (1), 105-119.

# Appendix A: Specification of the HUMAINE Emotion Annotation and Representation Language (EARL)

## Attributes common to all emotion elements

*Attributes shared by all EARL emotion elements (simple or complex). They cover: an optional, unique identifier for each emotion element; the scope of the annotation (i.e., what it refers to); and its modality. Scope is meaningful on highest-level emotion elements only, not on emotion elements embedded in a complex emotion.*

Attribute	Type	Use	Documentation
id	xs:ID	optional	unique identifier of the emotion.tag
xlink:href	xs:string	optional	the scope of the emotion element, as a reference (e.g. an XML element ID in the same or another document, or a file name)
start	xs:float	optional	the scope of the emotion element in terms of time – the time where the emotion starts. Usually specified in seconds from the start of the current clip.
end	xs:float	optional	the time where the emotion finishes.
modality	modalityType, i.e. one of: <i>face, voice, body, biosignal, text</i>	optional	indicates the modality through which the emotion is expressed (when labelling) or through which the emotion is to be expressed (when generating).

## Samples Element <samples>

*The sampling mechanism specifies a list of values taking into account a rate (number of samples per second)*

Attribute	Type	Use	Documentation
rate	xs:float	optional	number of samples per second
values	xs:string	optional	it specifies on what the sampling mechanism is applied. For example, dimensions or intensity or regulation, etc.

## Simple Emotion Element `<emotion>`

A “simple” (atomic) emotion can be described in terms of categories, dimensions and/or appraisals. The specification in terms of a category can be complemented by an optional intensity. It can enclose an optional list of `<samples>` elements. This list of samples elements can be optionally followed by arbitrary XML structures, in particular by plain text. If such a sub-structure (apart from `<samples>` elements) is present, then this enclosed structure is the scope of the emotion; if the `<emotion>` element is empty or contains only `<samples>` elements and white space, then a scope for the emotion must be specified using either the `xlink:href` reference or start and end times.

Attribute	Type	Use	Documentation
category	categoryType	optional	describe an emotional state in terms of an emotion category. See below for details on the definition of valid category values.
intensity	xsd:float	optional	specifies the intensity of the emotion, between 0 and 1.
[attributes denoting emotion dimensions]	xs:float	optional	describe an emotional state in terms of emotion dimensions. See below for details on the definition of valid dimension attributes. Valid values for dimensions typically range from -1 to 1, even though users may decide to use only the values from 0 to 1 if they consider a dimension to be unipolar rather than bipolar.
[attributes denoting appraisals]	xs:float	optional	describe an emotional state in terms of the appraisals bringing about the emotion. See below for details on the definition of valid appraisal attributes. Valid values for appraisals typically range from -1 to 1, even though users may decide to use only the values from 0 to 1 if they consider an appraisal scale to be unipolar rather than bipolar.
probability	xs:float	optional	the probability that the emotion specification is adequate. In annotation, this corresponds to labeller confidence / classification probability; in generation, to the probability of the state being realised.
[attributes denoting regulation] simulate suppress amplify attenuate	xs:float	optional	indicates that the emotion is not expressed in the same way as it is perceived; some sort of control is exercised by the person, to fulfil some socially motivated expression target.

## Sub-elements

Element	Type	Use	Minimum	Maximum	Documentation
samples	sampleType	optional	0	unbounded	

## <emotion> Examples

- Annotating a text:  

```
<emotion category="pleasure">Hello!</emotion>
```
- Annotating a photo of a face:  

```
<emotion xlink:href="face12.jpg" category="pleasure"/>
```
- Annotating a time span:  

```
<emotion start="0.4" end="1.3" category="pleasure"/>
```
- Emotion dimensions:  

```
<emotion xlink:href="face12.jpg" arousal="-0.2" valence="0.5" power="0.2"/>
```
- Appraisals  

```
<emotion xlink:href="face12.jpg" suddenness="-0.8" intrinsic_pleasantness="0.7" goal_conduciveness="0.3" relevance_self_concerns="0.7" />
```
- Combining labels:  

```
<emotion category="pleasure" intensity="0.9" simulate="0.6" modality="face" probability="0.5">Hello!</emotion>
```
- Annotating time-varying signals (Feeltrace: arousal+valence):  

```
<emotion start="2" end="2.7">  
  <samples value="arousal" rate="10">  
    0 .1 .25 .4 .55 .6 .65 .66  
  </samples>  
  <samples value="valence" rate="10">  
    0 -.1 -.2 -.25 -.3 -.4 -.4 -.45  
  </samples>  
</emotion>
```

## Complex Emotion Element <complex-emotion>

*It describes an emotion which is complex in the sense that it is composed of several ( $\geq 2$ ), atomic or complex, emotion descriptions. Still, it refers to a single "entity" (reference or start/end time). Only the top level <complex-emotion> element should have a ref or start/end attributes, to reflect the fact that they are jointly attached to a single entity. Typical cases are the co-occurrence of two or more emotions, possibly varying in intensity, and regulation attempts, where one emotion may be masked by the simulation of another one.*

### Sub-elements

Element	Type	Use	Minimum	Maximum	Documentation
emotion	emotionType	optional	0	unbounded	
complex-emotion	complexEmotionType	optional	0	unbounded	

### <complex-emotion> Examples

- Ambiguity coded through probability:

```
<complex-emotion xlink:href="face12.jpg">
  <emotion category="pleasure" probability="0.5">
  <emotion category="friendliness" probability="0.5">
</complex-emotion>
```
- Major/minor emotion coded through intensity:

```
<complex-emotion xlink:href="face12.jpg">
  <emotion category="pleasure" intensity="0.7" />
  <emotion category="worry" intensity="0.5" />
</complex-emotion>
```
- A simulated emotion masking a suppressed one:

```
<complex-emotion xlink:href="face12.jpg">
  <emotion category="pleasure" simulate="1.0" />
  <emotion category="worry" suppress="1.0" />
</complex-emotion>
```
- Different emotions in different modalities:

```
<complex-emotion xlink:href="face12.jpg">
  <emotion category="pleasure" modality="face" />
  <emotion category="worry" modality="voice" />
</complex-emotion>
```

## EARL root element <earl>

When EARL annotation is used as a standalone document, the document root node <earl> is used. It groups <emotion> and <complex-emotion> elements, and refers to a namespace.

### Sub-elements

Element	Type	Use	Minimum	Maximum	Documentation
emotion	emotionType	optional	0	unbounded	
complex-emotion	complexEmotionType	optional	0	unbounded	

### <earl> Examples

- A stand-off annotation of a speech synthesis markup (SSML) document:

The ssml document, doc123.ssml: (note that we need to add “id” attributes to SSML sentence tags to allow for cross-referencing)

```
<speak xmlns="http://www.w3.org/2001/10/synthesis" [...]>
  <voice gender="female">
    <s id="s1">Why are you so angry today?</s>
  </voice>
  <voice gender="male">
    <s id="s2">I am not angry!!</s>
  </voice>
</speak>
```

The stand-off annotation document, doc123.earl:

```
<earl xmlns="http://emotion-research.net/earl/040/default" [...]>
  <emotion xlink:href="doc123.ssml#s1" category="empathy"/>
  <emotion xlink:href="doc123.ssml#s2" category="anger"/>
</earl>
```

# Appendix B: EARL XML Schema

In the following, the full text of EARL schema files is listed. Naming conventions are proposed allowing for the consistent integration of schemas into EARL dialects.

## B.1 The core schema defining the EARL structure

The core EARL schema, defining the overall structure of an EARL document, is named:

earl-base-<version-number>.xsd

The version number for this release is 0.4.0, so that the file below must be called

earl-base-0.4.0.xsd

```
<?xml version="1.0"?>
<!-- Emotion annotation and representation language -->
<!-- Version 0.2, by Myriam Lamolle, 13 October 2005 -->
<!-- Version 0.3, by Marc Schroeder, 27 October 2005 -->
<!-- Version 0.4, by Myriam Lamolle, 25 June 2006 -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            elementFormDefault="qualified" attributeFormDefault="unqualified">

    <xsd:import namespace="http://www.w3.org/1999/xlink"
        schemaLocation="http://www.loc.gov/standards/mets/xlink.xsd"/>

    <!-- ***** -->
    <!-- EMOTION DEFINITION -->
    <!-- ***** -->
    <!-- ##### -->
    <!-- REGULATION ATTRIBUTE GROUP DEFINITION -->
    <xsd:attributeGroup name="regulationAttributeGroup">
        <xsd:annotation>
            <xsd:documentation>A number of regulation attributes to indicate
            attempts to suppress, amplify, attenuate or simulate the expression of an
            emotion</xsd:documentation>
        </xsd:annotation>
        <xsd:attribute name="suppress" type="xsd:float" use="optional"/>
        <xsd:attribute name="simulate" type="xsd:float" use="optional"/>
        <xsd:attribute name="amplify" type="xsd:float" use="optional"/>
        <xsd:attribute name="attenuate" type="xsd:float" use="optional"/>
    </xsd:attributeGroup>
    <!-- ##### -->
    <!-- MODALITY TYPE DEFINITION -->
    <xsd:simpleType name="modalityType">
        <xsd:annotation>
```

```

    <xsd:documentation>The modality type specifies the allowed values of
emotion modality. It indicates through which the emotion is expressed (when
labelling) or through which the emotion is to be expressed
generating</xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="voice"/>
    <xsd:enumeration value="text"/>
    <xsd:enumeration value="face"/>
    <xsd:enumeration value="body"/>
    <xsd:enumeration value="biosignal"/>
  </xsd:restriction>
</xsd:simpleType>
<!-- ##### -->
<!-- START and END time ATTRIBUTE GROUP for complexEmotion and emotion
-->
  <xsd:attributeGroup name="emotionTimeAttributeGroup">
    <xsd:annotation>
      <xsd:documentation>This attribute group specifies the common
attribute used by emotion and complex emotion. Start and end represent the
start and the end time of emotion.</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="start" type="xsd:float" use="optional"/>
    <xsd:attribute name="end" type="xsd:float" use="optional"/>
  </xsd:attributeGroup>
<!-- ##### -->
<!-- SAMPLING DEFINITION -->
  <xsd:simpleType name="listValueSampleType">
    <xsd:annotation>
      <xsd:documentation>We propose a sampling mechanism, assuming that
points occur at fixed time intervals determined by a "sampling rate" (= the
number of samples per second).</xsd:documentation>
    </xsd:annotation>
    <xsd:list itemType="xsd:decimal"/>
  </xsd:simpleType>
  <xsd:complexType name="sampleType">
    <xsd:annotation>
      <xsd:documentation>The sample type specifies the list of values
corresponding to a sampling .</xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base="listValueSampleType">
        <xsd:attribute name="rate" type="xsd:positiveInteger"/>
        <xsd:attribute name="values" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
<!-- ##### -->
<!-- EMOTION TYPE DEFINITION -->
  <xsd:complexType name="emotionType" mixed="true">

```

```

    <xsd:annotation>
      <xsd:documentation>The emotion type contains different attributes to
describe an emotion. An emotion have (or not) a sequence of
sampling</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="samples" type="sampleType" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:choice maxOccurs="unbounded" minOccurs="0">
        <xsd:any namespace="##other" processContents="lax"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" />
    <xsd:attribute ref="xlink:href" use="optional"/>
    <xsd:attribute name="modality" type="modalityType" use="optional"/>
    <xsd:attributeGroup ref="emotionTimeAttributeGroup"/>
    <xsd:attribute name="category" type="categoryType"/>
    <xsd:attribute name="probability" type="xsd:float" use="optional"/>
    <xsd:attributeGroup ref="dimensionAttributeGroup"/>
    <xsd:attributeGroup ref="appraisalAttributeGroup"/>
    <xsd:attributeGroup ref="regulationAttributeGroup"/>
    <xsd:attribute name="intensity" type="xsd:float" use="optional"/>
  </xsd:complexType>

<!-- ##### -->
<!-- COMPLEX EMOTION DEFINITION -->
<xsd:complexType name="complexEmotionType" mixed="true">
  <xsd:annotation>
    <xsd:documentation>A complex emotion describes an emotion which is
complex in the sense that is composed of several complex emotion
descriptions. Still, it refers to a single "entity" (reference or start/end
time). Only the tag "complex-emotion" element should have a ref or
start/end attributes, to reflect the fact that they are joined single
entity. Typical cases are the co-occurrence of two or more emotion,
possibly varying in intensity; attempts, where on emotion may be masked by
the simulation of another one; or anotation of different
modalities.</xsd:documentation>
  </xsd:annotation>
  <xsd:choice minOccurs="2" maxOccurs="unbounded">
    <xsd:element name="emotion" type="emotionType" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="complex-emotion" type="complexEmotionType"
minOccurs="0" maxOccurs="unbounded"/>
  </xsd:choice>
  <xsd:attribute name="id" type="xsd:ID" />
  <xsd:attribute ref="xlink:href" use="optional"/>
  <xsd:attributeGroup ref="emotionTimeAttributeGroup"/>
  <xsd:attribute name="modality" type="modalityType" use="optional"/>
</xsd:complexType>

<!-- ##### -->

```

```

<!-- ##### -->
<!-- element EMOTION -->
<xsd:element name="emotion" type="emotionType" />
<!-- element COMPLEX_EMOTION -->
<xsd:element name="complex-emotion" type="complexEmotionType" />

<!-- EARL type Definition -->
<xsd:complexType name="earlType" mixed="true">
  <xsd:sequence minOccurs="1" maxOccurs="unbounded">
    <xsd:element name="complex-emotion" type="complexEmotionType"
minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="emotion" type="emotionType" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<!-- root element definition -->
<xsd:element name="earl" type="earlType"/>
</xsd:schema>

```

## B.2 Schemas defining default sets of categories, dimensions or appraisals

### B.2.1 Categories

The following Schema file defines the set of 48 emotion categories introduced in Section 1.1.4 so that these can be used as values of the “category” attribute of an <emotion> element. File naming conventions:

```
earl-categories-<domain>.xsd
```

As this is the default category set, the name of this file is:

```
earl-categories-default.xsd
```

Note that there is no version number in the name, i.e. the format of defining content is not expected to change between versions of EARL.

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- CATEGORY TYPE DEFINITION -->
  <xsd:simpleType name="categoryType">
    <xsd:annotation>
      <xsd:documentation>
        A category set consisting of 27 everyday emotion words, distilled
        from 55 words list of 2004 summer school. In order to be able to
        annotate explicitly the absence of an emotion, we add the category
        "neutral".
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:NMTOKEN">
      <!-- Negative and forceful -->
      <xsd:enumeration value="anger"/>
      <xsd:enumeration value="annoyance"/>
      <xsd:enumeration value="contempt"/>
      <xsd:enumeration value="disgust"/>
      <xsd:enumeration value="irritation"/>
      <!-- Negative and not in control -->
      <xsd:enumeration value="anxiety"/>
      <xsd:enumeration value="embarrassment"/>
      <xsd:enumeration value="fear"/>
      <xsd:enumeration value="helplessness"/>
      <xsd:enumeration value="powerlessness"/>
      <xsd:enumeration value="worry"/>
      <!-- Negative thoughts -->
      <xsd:enumeration value="doubt"/>
      <xsd:enumeration value="envy"/>
      <xsd:enumeration value="frustration"/>
      <xsd:enumeration value="guilt"/>
      <xsd:enumeration value="shame"/>
      <!-- Negative and passive -->
      <xsd:enumeration value="boredom"/>
      <xsd:enumeration value="despair"/>
      <xsd:enumeration value="disappointment"/>
      <xsd:enumeration value="hurt"/>
      <xsd:enumeration value="sadness"/>
      <!-- Agitation -->
      <xsd:enumeration value="shock"/>
      <xsd:enumeration value="stress"/>
      <xsd:enumeration value="tension"/>
      <!-- Positive and lively -->
      <xsd:enumeration value="amusement"/>
      <xsd:enumeration value="delight"/>
      <xsd:enumeration value="elation"/>
      <xsd:enumeration value="excitement"/>
      <xsd:enumeration value="happiness"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>

```

```

<xsd:enumeration value="joy"/>
<xsd:enumeration value="pleasure"/>
<!-- Caring -->
<xsd:enumeration value="affection"/>
<xsd:enumeration value="empathy"/>
<xsd:enumeration value="friendliness"/>
<xsd:enumeration value="love"/>
<!-- Positive thoughts -->
<xsd:enumeration value="courage"/>
<xsd:enumeration value="hope"/>
<xsd:enumeration value="pride"/>
<xsd:enumeration value="satisfaction"/>
<xsd:enumeration value="trust"/>
<!-- Quiet positive -->
<xsd:enumeration value="calm"/>
<xsd:enumeration value="content"/>
<xsd:enumeration value="relaxed"/>
<xsd:enumeration value="relieved"/>
<xsd:enumeration value="serene"/>
<!-- Reactive -->
<xsd:enumeration value="interest"/>
<xsd:enumeration value="politeness"/>
<xsd:enumeration value="surprise"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

## B.2.2 Dimensions

The following Schema file defines the set of 3 emotion dimensions introduced in Section 1.1.5 so that these can be used as attributes representing emotion dimensions in an <emotion> element. File naming conventions:

```
earl-dimensions-<domain>.xsd
```

As this is the default set of emotion dimensions, the name of this file is:

```
earl-dimensions-default.xsd
```

Note that there is no version number in the name, i.e. the format of defining content is not expected to change between versions of EARL.

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- ##### -->
  <!-- DIMENSION ATTRIBUTE GROUP DEFINITION -->
  <xsd:attributeGroup name="dimensionAttributeGroup">
    <xsd:annotation>
    <xsd:documentation>
      Emotional dimensions: naming conventions vary. We suggest the
      following names, for reasons of clarity, limited ambiguity
      wide-spread use
    </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="arousal" type="xsd:float" use="optional"/>
    <xsd:attribute name="valence" type="xsd:float" use="optional"/>
    <xsd:attribute name="power" type="xsd:float" use="optional"/>
  </xsd:attributeGroup>
</xsd:schema>

```

### **B.2.3 Appraisals**

The following Schema file defines the set of appraisals introduced in Section 1.1.6 so that these can be used as attributes representing appraisals in an <emotion> element. File naming conventions:

earl-appraisals-<domain>.xsd

As this is the default appraisal set, the name of this file is:

earl-appraisals-default.xsd

Note that there is no version number in the name, i.e. the format of defining content is not expected to change between versions of EARL.

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- APPRAISALS TYPE DEFINITION -->
  <xsd:attributeGroup name="appraisalAttributeGroup">
    <xsd:annotation>
      <xsd:documentation>The appraisal type specifies the allowed values of
appraisal emotion. The following list is flattened representation which we
have tentatively formulated based on Klaus Scherer</xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="suddenness" type="xsd:float" use="optional"/>
    <xsd:attribute name="familiarity" type="xsd:float" use="optional"/>
    <xsd:attribute name="predictability" type="xsd:float" use="optional"/>
    <xsd:attribute name="intrinsic_pleasantness" type="xsd:float"
use="optional"/>
    <xsd:attribute name="relevance_self_concerns" type="xsd:float"
use="optional"/>
    <xsd:attribute name="relevance_relationship_concerns" type="xsd:float"
use="optional"/>
    <xsd:attribute name="relevance_social_order_concerns" type="xsd:float"
use="optional"/>
    <xsd:attribute name="goal_outcome_probability" type="xsd:float"
use="optional"/>
    <xsd:attribute name="consonant_with_expectation" type="xsd:float"
use="optional"/>
    <xsd:attribute name="goal_conduciveness" type="xsd:float"
use="optional"/>
    <xsd:attribute name="goal_urgency" type="xsd:float" use="optional"/>
    <xsd:attribute name="cause_agent_self" type="xsd:float"
use="optional"/>
    <xsd:attribute name="cause_agent_other" type="xsd:float"
use="optional"/>
    <xsd:attribute name="cause_motive_intentional" type="xsd:float"
use="optional"/>
    <xsd:attribute name="event_controllability" type="xsd:float"
use="optional"/>
    <xsd:attribute name="agent_power" type="xsd:float" use="optional"/>
    <xsd:attribute name="goal_adjustment_possible" type="xsd:float"
use="optional"/>
    <xsd:attribute name="standards_compatibility_external" type="xsd:float"
use="optional"/>
    <xsd:attribute name="standards_compatibility_internal" type="xsd:float"
use="optional"/>
  </xsd:attributeGroup>
</xsd:schema>

```

### **B.3 Schema defining the default EARL dialect by binding together the EARL structure and the three default sets of content concepts**

As depicted in Figure 1, a concrete EARL dialect is defined by combining the base EARL schema, defining the EARL structure, with content specifications for categories, dimensions and appraisals.

Naming conventions:

File name: earl-`<domain>`-`<version>`.xsd

Namespace: `http://emotion-research.net/earl/<version-without-dots>/<domain>`

Consequently, the default EARL dialect for version 0.4.0 is called:

File name: earl-default-0.4.0.xsd

Namespace: `http://emotion-research.net/earl/040/default`

The following Schema file, earl-default-0.4.0.xsd, defines the default EARL dialect in this sense.

```
<xsd:schema targetNamespace="http://emotion-research.net/earl/040/default"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="earl-base-0.4.0.xsd"/>
  <xsd:include schemaLocation="earl-categories-default.xsd"/>
  <xsd:include schemaLocation="earl-dimensions-default.xsd"/>
  <xsd:include schemaLocation="earl-appraisals-default.xsd"/>
</xsd:schema>
```

## B.4 Defining other EARL dialects

### B.4.1 Defining new category sets

For each new set of categories to be used, a category definition file earl-categories-`<domain>`.xsd must be created. Here are some examples.

- earl-categories-none.xsd – the empty set, in case no categories are to be used.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- CATEGORY TYPE DEFINITION -->
  <xsd:simpleType name="categoryType">
    <xsd:annotation>
      <xsd:documentation>no categories </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value=""/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

- earl-categories-aibo.xsd – an application-specific set of eleven emotion categories

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- CATEGORY TYPE DEFINITION -->
  <xsd:simpleType name="categoryType">
    <xsd:annotation>
      <xsd:documentation>
        A category set consisting of 10 emotion classes (plus neutral)
        as used in the annotation of the AIBO database, described in:
        Batliner et al. (2004). "You stupid tin box" - Children interacting
        with the AIBO robot. LREC 2004, pp. 171-174.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="joyful"/>
      <xsd:enumeration value="surprised"/>
      <xsd:enumeration value="neutral"/>
      <xsd:enumeration value="emphatic"/>
      <xsd:enumeration value="helpless"/>
      <xsd:enumeration value="touchy"/>
      <xsd:enumeration value="angry"/>
      <xsd:enumeration value="motherese"/>
      <xsd:enumeration value="bored"/>
      <xsd:enumeration value="reprimanding"/>
      <xsd:enumeration value="rest"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>

```

## B.4.2 Defining new sets of emotion dimensions

For each new set of emotion dimensions to be used, a dimension definition file `earl-dimensions-<domain>.xsd` must be created. Here are some examples.

- `earl-dimensions-none.xsd` – the empty set, in case no dimensions are to be used.

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- ##### -->
  <!-- DIMENSION ATTRIBUTE GROUP DEFINITION -->
  <xsd:attributeGroup name="dimensionAttributeGroup">
    <xsd:annotation>
      <xsd:documentation>no emotion dimensions
    </xsd:documentation>
    </xsd:annotation>
  </xsd:attributeGroup>
</xsd:schema>

```

- earl-dimensions-aibo.xsd – two application-specific emotion dimensions.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- ##### -->
  <!-- DIMENSION ATTRIBUTE GROUP DEFINITION -->
  <xsd:attributeGroup name="dimensionAttributeGroup">
    <xsd:annotation>
      <xsd:documentation>
        Two dimensions as found in an analysis of the AIBO database,
        described in:
        Batliner et al. (2005). Private Emotions vs. Social Interaction -
        towards New Dimensions in Research on Emotion.
        Proc. Workshop on Adapting the Interaction Style to Affective
        Factors, User Modelling 2005
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="valence" type="xsd:float" use="optional"/>
    <xsd:attribute name="interaction" type="xsd:float" use="optional"/>
  </xsd:attributeGroup>
</xsd:schema>
```

### B.4.3 Defining new appraisal sets

For each new set of appraisals to be used, an appraisal definition file earl-appraisals-  
<domain>.xsd must be created. Here are some examples.

- earl-appraisals-none.xsd – the empty set, in case no appraisals are to be used.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- ##### -->
  <!-- DIMENSION ATTRIBUTE GROUP DEFINITION -->
  <xsd:attributeGroup name="appraisalAttributeGroup">
    <xsd:annotation>
      <xsd:documentation>no appraisals</xsd:documentation>
    </xsd:annotation>
  </xsd:attributeGroup>
</xsd:schema>
```

- earl-appraisals-pleasantcontrol.xsd – a set of just two appraisals, pleasantness and control.

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- APPRAISALS TYPE DEFINITION -->
  <xsd:attributeGroup name="appraisalAttributeGroup">
    <xsd:annotation>
      <xsd:documentation>Two appraisals, pleasantness and control
    </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="pleasantness" type="xsd:float" use="optional"/>
    <xsd:attribute name="control" type="xsd:float" use="optional"/>
  </xsd:attributeGroup>
</xsd:schema>

```

## B.4.4 Defining new EARL dialects

A new EARL dialect is created by combining a set of categories, a set of dimensions and a set of appraisals with the base EARL structure. This is done in a simple file

earl-`<domain>-version.xsd`

defining namespace

`http://emotion-research.net/earl/<version-without-dots>/<domain>`.

Some examples follow.

- earl-aibolabels-0.4.0.xsd – only the aibo category labels, no dimensions or appraisals

```

<xsd:schema
  targetNamespace="http://emotion-research.net/earl/040/aibolabels"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="earl-base-0.4.0.xsd"/>
  <xsd:include schemaLocation="earl-categories-aibo.xsd"/>
  <xsd:include schemaLocation="earl-dimensions-none.xsd"/>
  <xsd:include schemaLocation="earl-appraisals-none.xsd"/>
</xsd:schema>

```

- earl-defaultdimensions-0.4.0.xsd – only the three default dimensions, no categories or appraisals

```

<xsd:schema
  targetNamespace="http://emotion-research.net/earl/040/defaultdimensions"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="earl-base-0.4.0.xsd"/>
  <xsd:include schemaLocation="earl-categories-none.xsd"/>
  <xsd:include schemaLocation="earl-dimensions-default.xsd"/>
  <xsd:include schemaLocation="earl-appraisals-none.xsd"/>
</xsd:schema>

```