

RRL Specification v0.4

Last Edited: 28 March 2003 by Hannes Pirker

The latest version of this paper should be retrieveable at the NECA-RRL-page

<http://www.oefai.at/NECA/RRL>

This is a NECA- draft of the RRL specification. Structures, elements and attributes are proposed and discussed according to the interfaces where they are required:

- ⑩ Scene Generation - Multimodal NLG
- ⑩ Multimodal NLG - Concept to Speech
- ⑩ Concept to Speech - Gesture Assignment
- ⑩ Gesture Assignment – Animation Generation

The current proposals are the results of intense discussions between the respective section authors.

At the end, a listing of the XML Schema is provided.

The current status of this document is of course preliminary. At a later stage we envisage providing a more formal presentation of XML elements and attributes, along the lines of the SABLE and VHML manuals.

The Scene Generation – Multimodal NLG interface

Authors: Marc Schröder, Paul Piwek and Martin Klesen

This section describes the structure of an RRL document in NECA at the interface between the Scene Generation module and the Multimodal Natural Language Generation module. The structure of the XML document is explained at three levels of organisation, going from the overall structure towards more local structure. At the end of the section, an example is given for the XML representation of a simple dialogue.

The scene description

Essentially, a scene consists of an initial common ground and a list of participants, the set of “acts” (i.e. dialogue acts and non-communicative acts), and information about their temporal ordering (see below).

In XML, we propose to specify this as follows (leaving out sub-structure for clarity; for illustration, different possible values of attributes are separated by vertical bars `|`):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<necaRRL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://neca.sysis.at/2002/04/necarrl"
  xsi:schemaLocation="http://neca.sysis.at/2002/04/necarrl
file:neca_eshowroom.0.4.xsd"
  version="0.4"
  xml:lang="en-US">
  <processingState>
    <state avail="sceneInfoAvailable"/>
  </processingState>

  <initialCommonGround id="d_1">
    ...drs ...
  </initialCommonGround>
  <participants>
    <person id="ritchie">
      <realname firstname="..."/>
      <gender type="male|female"/>
      <appearance character="Ritchie|Tina..."/>
      <voice> ...</voice>
      <personality agreeableness="..." politeness="..."/>
      <domainSpecificAttr role="seller|buyer"
        x-position="..." y-position="1..."/>
    </person>
    <person id="tina">
      <gender type="female"/>
      <domainSpecificAttr role="buyer" .../>
      ...
    </person>
  </participants>

  <temporalOrdering>
```

```

    <seq>
      <act id="v_1" />
      <act id="v_2" />
    </seq>
  </temporalOrdering>

  <setOfActs>
    <dialogueAct id="v_1">
      <...
    </dialogueAct>
    <dialogueAct id="v_2">
      ...
    </dialogueAct>
  </setOfActs>
</necaRRL>

```

The required syntax for a valid RRL document is specified in an XML Schema, consisting of a scenario-independent part specifying the structure of the document, and a scenario-dependent part specifying possible values for attributes carrying domain information. Notice the reference to the Schema `neca_eshowroom.xsd` in the opening tag of the root element `<necaRRL>`.

The `<temporalOrdering>`-element contains a specification of the temporal ordering among the individual dialogue acts and non-communicative acts. A simple sequence order is assumed for the moment. At later stages of the project, the temporal ordering could be extended to specify that two acts are to be carried out simultaneously, e.g. for backchanneling. In addition, the temporal ordering under the `<temporalOrdering>` element can contain under-specified temporal ordering to be complemented by the M-NLG, i.e. at the M-NLG input the temporal ordering might still be underspecified, but has to be fully specified at the M-NLG output.

Persons have a personality, defined using three factors out of the five-factor model of personality. In addition, they have a domain-specific element `<domainSpecificAttr>` which carries information about the persons relevant for the domain. In the eShowroom scenario, for example, the *role* of a person (buyer, seller) is relevant.

The dialogue act

A dialogue act has a type, a `reactionTo` reference to a previous act, a speaker, a list of addressees, an emotion element allowing to specify a set of emotions for both speaker and addressees, and some semantic content.

The XML form we propose for this is (filling in concrete example values for the attributes, in order to maintain readability):

```

<dialogueAct id="v_1">
  <domainSpecificAttr type="requestValue" />
  <speaker id="tina" />
  <addressee id="ritchie" />

```

```

    <emotion>
      <emotionExpressed type="joy" intensity="0.3" .../>
    </emotion>
    <semanticContent id="d_1">
      ...drs...
    </semanticContent>
  </dialogueAct>

```

Notice that only the simple data structures, namely the dialogue act's id and reactionTo, are realised as attribute/value pairs. More complex data structures, and data structures that may become more complex as time goes by, are realised as child elements.

Emotions are specified as a property of the dialogue acts, rather than a property of the speaker, in order to keep representation of the speaker simple (it does not change). In order to be able to specify possible differences between the speaker's real emotional state and the emotions to be used in her expression (which occur, e.g., when social rules do not allow to express feelings too frankly), <emotion> comprises a set of <emotionExpressed> and <emotionFelt>. The first specifies the expression to be used for uttering the dialogue act, the latter specifies the speaker's "real" feelings.

A list of addressees is obtained by simply allowing the <addressee> element to occur more than once.

Dialogue acts have domain-specific properties encoded in a <domainSpecificAttr> element. This element e.g. contains the dialogue act type which may differ between scenarios.

The DRS

A discourse representation structure is the type of data that occurs inside <initialCommonGround> and <semanticContent> elements, as well as recursively in some "conditions", e.g. <negation>.

A DRS has a first part which is an enumeration of referents, followed by a possibly complex list of "conditions" making statements about these referents.

```

<semanticContent id="d_1">
  <referent id="x_1"/>
  <referent id="x_2"/>
  ...
  <referent id="x_n"/>

  <!-- and now a list of conditions, in any order
       one or many of the following: -->

  <unaryCond id="c_1" pred="car|sportyCar|..."
    arg="x_1|...|x_n|leatherSeats|..." />

  <binaryCond id="c_2" pred="have|..."

```

```

    argOne="x_1|...|x_n|leatherSeats|..."
    argTwo="x_1|...|x_n|leatherSeats|..."/>

<negation id="c_3"> ...drs... </negation>

<conditional id="c_4">
  <argOne> ...drs... </argOne>
  <argTwo> ...drs... </argTwo>
</conditional>

<disjunction id="c_5">
  <argOne> ...drs... </argOne>
  <argTwo> ...drs... </argTwo>
</disjunction>

<dTopic id="c_6"
  arg="x_1|...|x_n|leatherSeats|..."/>

<free id="c_7" arg="x_1|...|x_n"/>

  <bridge id="c_8"> ...drs... </bridge>
</semanticContent>

```

Example Dialogue

In the following, we give an example of how these ideas translate into an XML document for a given dialogue. Of course, as we are at the stage in the NECA architecture where no words have yet been specified, the XML document at this stage consists only of the abstract dialogue act content specification.

A "target" dialogue

Ritchie (role: seller, personality traits: very agreeable, slightly extravert)

Genie (role: buyer, personality traits: moderately agreeable, very extravert)

M: "Hello! My name is Ritchie."

M: "What can I do for you?"

G: "Could you tell us something about this car?"

M: "This is a very sporty car."

M: "It can drive 100 miles per hour."

G: "How much gas does it consume?"

M: "It consumes 8 liters per 60 miles."

G: "Not bad!"

G: "Ok! We'll take it!"

M: "Congratulations! Exquisite taste!"

Informal semantic specification:

There will be an initial common ground. The value is a *drs*. Here it might contain just the car which is under discussion:

[x|car(x)]

Let us now have a quick glance at the semantic content and the dialogue act type of the utterances in the example dialogue:

M: "Hello! My name is Ritchie."

Type: greeting

Content: none

M: "What can I do for you?"

Type: openingquestion

Content: none

Remark: This part of the dialogue will be taken care of in the NLG mainly by using canned text fragments. Hence, no detailed description of the semantic contents of the utterances is

required. The dialogue act type is sufficient. These utterances occur only at very specific points in the dialogue (e.g., the first two moves).

G: "Could you tell us something about this car?"

Type: requestinfo
Content: [[d_topic(x)]]

M: "This is a very sporty car."

Type: inform
Content: [|very_sporty(x)]

M: "It can drive 100 miles per hour."

Type: inform
Content: [|maxspeed(x,100mph)]

G: "How much gas does it consume?"

Type: requestvalue
Content: [|consumption(x,y), free(y)]

M: "It consumes 8 liters per 60 miles."

Type: inform
Content: [|consumption(x,8lph)]

Remark: Here the feedback attribute of the dialogue act will point to the question which the assertion answers.

G: "Not bad!"

Type: positive_response
Content: none

G: "Ok! We'll take it!"

Type: initiate_closing_positive
Content: none

M: "Congratulations! Exquisite taste!"

Type: complete_positive_closing
Content: none

Formal XML Representation: example1_mminput.rrl

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<necaRRL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://neca.sysis.at/2002/04/necarrl"
  xsi:schemaLocation="http://neca.sysis.at/2002/04/necarrl
file:neca_eshowroom.0.4.xsd"
  version="0.4"
  xml:lang="en-US">

  <!-- the globally accessible schema should be placed on the neca
server:

xsi:noNamespeaceSchemaLocation="http://neca.sysis.at/RRL/03/neca_esho
wroom.xsd"
  -->

  <processingState>
    <state avail="sceneInfoAvailable"/>
  </processingState>

  <initialCommonGround id="d_1">
    <referent id="x_1"/>
    <unaryCond id="c_1" pred="car" arg="x_1"/>
  </initialCommonGround>
  <participants>
    <person id="ritchie">
      <realname firstname="Ritchie"/>
      <gender type="male"/>
      <appearance character="Ritchie"/>
      <voice name="us2"><prosody rate="-10%" pitch="-20%"/></voice>
      <personality agreeableness="0.8" politeness="polite"/>
      <domainSpecificAttr role="seller"
        x-position="200" y-position="100"/>
    </person>
    <person id="tina">
      <realname title="Miss" firstname="Tina"/>
      <gender type="female"/>
      <appearance character="Tina"/>
      <voice name="us1"><prosody rate="+10%" pitch="+20%"/></voice>
      <personality agreeableness="0.8" politeness="impolite"/>
      <domainSpecificAttr role="buyer"
        x-position="100" y-position="100"/>
    </person>
  </participants>

  <temporalOrdering>
    <seq>
      <act id="v_1"/>
      <act id="v_2"/>
    </seq>
  </temporalOrdering>

  <setOfActs>
    <dialogueAct id="v_1">
      <domainSpecificAttr type="requestValue"/>
      <speaker id="tina"/>
    </dialogueAct>
  </setOfActs>

```

```

    <addressee id="ritchie"/>
    <semanticContent id="d_1">
      <ternaryCond id="c_2" pred="attribute"
        argOne="x_1" argTwo="consumption" argThree="y"/>
      <free id="c_3" arg="y"/>
    </semanticContent>
  </dialogueAct>
  <dialogueAct id="v_2">
    <domainSpecificAttr type="inform"/>
    <speaker id="ritchie"/>
    <addressee id="tina"/>
    <semanticContent id="d_2">
      <ternaryCond id="c_3" pred="attribute"
        argOne="x_1" argTwo="consumption" argThree="lph8"/>
    </semanticContent>
  </dialogueAct>
</setOfActs>
</necaRRL>

```

The Multimodal NLG – Concept-to-Speech interface

Authors: Marc Schröder, Stefan Baumann

The information produced by the Multimodal NLG module is integrated into the RRL document. Specifically, information is added under the <dialogueAct> elements.

Information provided by the Scene Generation module

The information relevant for the speech synthesis which is already provided by the Scene Generation module is the speaker, the dialogue act type, and the emotion with which the dialogue act is to be realised.

Below <dialogueAct> but before the individual <sentence>s composing the dialogue act, the following tags occur which carry relevant information for the speech synthesis module:

- ⑩ <speaker> is a required element containing only a reference attribute:
 - ⑩ name: The name of a speaker, as an identifier referencing to the more informative <person> element which only occurs once in the RRL document, in the initial participants list (required).
- ⑩ <emotion> is an optional element which, when present, specifies the emotion with which the dialogue act is to be uttered via <emotionExpressed>. The emotion is constant during the dialogue act. If a speaker has several things to express in a row, but with different emotional colouring, this can be realised as a sequence of dialogue acts from the same speaker. In addition to the the emotion type which is based on the OCC model, an alternative encoding in terms of the three dimensions activation, evaluation and power is provided. This encoding is used by the speech synthesis in order to control prosody. The <emotionExpressed> tag thus has the following attributes:

- ⑩ type: the emotion category, according to the appraisal model implemented in the affective reasoner ("joy", "distress", "satisfaction") (required).
- ⑩ intensity: the intensity of the emotion, on a scale from 0.0 to 1.0 (optional; if absent, a default intensity is assumed).
- ⑩ activation: a value on a scale from -1 to +1 (required)
- ⑩ evaluation: a value on a scale from -1 to +1 (required)
- ⑩ power: a value on a scale from -1 to +1 (required)

Temporal ordering of dialogue acts

The M-NLG outputs the (dialogue) acts in a fully specified temporal ordering:

```
<necaRRL>
...
<temporalOrdering>
  <seq>
    <act id="v_1"/>
    <act id="v_2"/>
  </seq>
</temporalOrdering>
<setOfActs>
...
</setOfActs>
</necaRRL>
```

At later stages in the project, this temporal ordering might allow for acts to happen simultaneously, using both <seq> and <par> elements¹.

Word-level information provided by the Multimodal NLG

To each dialogue act, the M-NLG adds a list of <sentence> elements (inspired from SSML). In each <sentence> element, the following child elements are admitted:

- ⑩ <text>The textual form of the sentence.
- ⑩ All the rest of the elements, as well as the <sentence>-element itself bear a required attribute "id" which contains an unique key in order to make every element identifiable. These "id"s are e.g. necessary for the coordination of speech and gestures (see below).
- ⑩ <word>: the individual words each correspond to a <word> tag, which has a number of attributes:
 - ⑩ pos ("part-of-speech", required);
 - ⑩ sampa (optional);
 - ⑩ id (required)

¹ <seq> and <par> are borrowed from SMIL 2.0 (Synchronized Multimedia Integration Language www.w3.org/TR/smil20)

- ⑩ <punct>: punctuation, with the attribute
 - ⑩ id (required)
- ⑩ <gesture>: meaning-carrying gestures and emotional interjections ("affect bursts"), with the following attributes:
 - ⑩ modality ("voice", "face", "body", optional);
 - ⑩ meaning (relatively general, optional);
 - ⑩ identifier (a specific name for a specific gesture or interjection, required).
 - ⑩ id (required)
 - ⑩ alignto, aligntype

The following example represents the sentence "Dieses Auto hat Ledersitze." ("This car has leather seats.") Let us assume that the correct pronunciation of the word "Ledersitze" cannot be automatically generated by the speech synthesis and must therefore be provided by the M-NLG. Let us also suppose that the M-NLG has determined the beginning of the sentence to be a good place for a interjection expressing the meaning "beautiful".

```
<sentence id="s1">
  <!-- <text>Dieses Auto hat Ledersitze.</text> →
  <gesture id="g1" modality="voice" identifier="beautiful"
    alignto="w1" aligntype="seq_before"/>
  <word id="w1" pos="PDAT">Dieses</word>
  <word id="w2" pos="NN"> Auto </word>
  <word id="w3" pos="VAFIN"> hat </word>
  <word id="w4" pos="NN" sampa="'le:-d6-,zI_ts@"/> Ledersitze </word>
  <punct id="p1">.</punct>
</sentence>
```

Sentence-level information provided by the Multimodal NLG

In addition to information on the word level, the text structure (including syntactic and information structure) must be encoded in a way compatible with the prosodic structure, which will be added by the speech synthesis component.

Since the prosodic structure might deviate from the syntactic structure and/or information structure leading to crossing edges in the XML document, we encode the prosodic structure non-hierarchically, i.e. boundaries of prosodic phrases are represented as empty elements. This solution is in line with current phonological theories.

The concept of *sentence-internal structural elements* seems to be useful for grouping everything that represents structure inside a sentence. Sentence-internal structural elements can occur inside <sentence> elements and, recursively, in sentence-internal structural elements. They encompass, currently,

- ⑩ <infoStruct>, the information structural partitioning of a sentence,

- ⑩ <synPhrase>, the syntactic structure, and
- ⑩ <infoStatus>, the informational status of individual referents.

The meaning of the sentence-internal structural elements and their attributes

- ⑩ <infoStruct> reflects the information structural partitioning of a sentence. Attribute:
 - ⑩ part, either "theme" or "rheme" (required).

The "theme" comprises the elements a sentence is about, often - but not necessarily - including elements that have been mentioned before in the discourse. There is a tendency of correlation between thematic elements and the subject on the one hand and sentence-initial position on the other.

The "rheme" represents a comment on the theme. It generally includes the most important information of the sentence, which is added to the knowledge base of the speaker and the hearer.

In order to cope with the problem of possible incompatibilities between syntax and information structure (i.e., crossing edges may occur) the division between theme and rheme within a sentence is not represented by a sequence of <theme> and <rheme> elements, but by embedding <rheme> into <theme> :

```
<sentence>
  <text>...</text>
  <infoStruct part="theme">
    ...
    <infoStruct part="rheme">
      ...
    </infoStruct>
  </infoStruct>
</sentence>
```

- ⑩ <synPhrase> represents a syntactic phrase. It has the following attributes:
 - ⑩ category (the syntactic phrase categories / nodes, for German following NEGRA (tags: e.g. *NP*, *VP*...), for English to be specified; required)
 - ⑩ function (the grammatical function, for German following NEGRA (tags: e.g. *SB*, *PD*, *OA*...), for English to be specified; required)
- ⑩ <infoStatus> informs about the informational status of a referent. Its attributes:
 - ⑩ type (the type of information status assigned to the enclosed elements; required). Possible values:
 - *referent-given* (i.e. an expression refers to a referent which is *identical with a referent* already given in the discourse),

- *concept-given* (i.e. an expression refers to a referent which *includes a concept* already given in the discourse),

- *contrast*.

"New" elements are not marked explicitly.

The attributes can be applied recursively and to elements of various sizes, e.g. the value "referent-given" might be assigned to a N or to a whole NP. A word or syntactic category can be assigned several tags at the same time, e.g. *xy-given* and *contrast*.

⑩ The <word> element introduced above is assigned an additional attribute reflecting grammatical function:

⑩ function (the grammatical function as for <synPhrase> tags; optional).

Assuming that in the above example, "Auto" is given information, "Dieses Auto" is a subject, and both "Dieses Auto" and "Ledersitze" are noun phrases, the example becomes:

```
<sentence id="s1">
<!-- <text>Dieses Auto hat Ledersitze.</text> →
  <gesture id="g1" modality="voice" identifier="beautiful"
    alignto="w1" aligntype="seq_before"/>
  <infoStruct part="theme">
    <synPhrase category="NP" function="SB">
      <word id="w1" pos="PDAT">Dieses</word>
      <infoStatus type="referent-given">
        <word id="w2" pos="NN"> Auto </word>
      </infoStatus >
    </synPhrase>
  <infoStruct part="rheme">
    <synPhrase phrase="VP" function="PD">
      <word id="w3" pos="VAFIN"> hat </word>
      <synPhrase phrase="NP" function="OA">
        <word id="w4" pos="NN"
          sampa="'le:-d6-,zI_ts@"/> Ledersitze </word>
      </synPhrase>
    </synPhrase>
  <punct id="p1">.</punct>
</infoStruct>
</infoStruct>
</sentence>
```

Gesture information within the RRL

Author: Hannes Pirker

Gestures can enter the RRL both at the M-NLG and at the Gesture-Assignment module (GA). While M-NLG is the place where “content related” gestures are specified, GA supplies “lower level” gestures such as blinking, gaze and breathing, but also fine-tunes the timing of gestures. “Gesture” in our terms encompasses to all all sorts of modalities such as posture, facial expression, affective bursts.

<gesture>: meaning-carrying gestures and emotional interjections ("affect bursts"), with the following attributes:

- ⑩ modality ("voice", "face", "body", optional);
- ⑩ meaning (relatively general, optional);
- ⑩ identifier (a specific name for a specific gesture or interjection. (required).
- ⑩ id : the identifier of a sppecific token (required)
- ⑩ alignto: the id of the entity to which the gesture is to be aligned Entities can be a <sentence>, a syntactic constituent or a <word> (required)
- ⑩ aligntype: specifying the positioning of the gesture in relation to the entity specified in the alignto attribute. Values can be “seq-before” , “seq-after” , “par_start” (gesture and alignto start at the same time), “par_end” (stop at the same time) etc.

<gesture> elements can appear virtually everywhere within a <dialogueAct>: They can be placed before or after sentences but are alsos allowed within sentences at the level same level as words.

Nevertheless, the actual position of a <gesture> element is completely arbitrary, as its physical positioning in respect to other entities is entirely captured by the alignto and aligntype attribute.

The Concept-to-Speech – Gesture Assignment interface

Authors: Marc Schröder

The types of information added by the concept-to-speech (CTS) module, to be used by Gesture Assignment, are of two kinds: prosodic sentence structure, and phonetic word substructure.

Prosodic sentence structure

Prosodic sentence structure is defined using the following element:

- ⑩ <prosBoundary> prosodic boundaries including pauses. This is an empty element representing a boundary at the place where the element occurs. Boundaries may occur at the same places as <word> elements, i.e. anywhere inside sentence-internal structural elements. Attributes:
 - ⑩ breakindex: A symbolic representation of boundary size, from 3 (minor boundary corresponding to the ToBI intermediate phrase ip)) ,through 4 (major boundary, corresponding to the ToBI intonation phrase (IP)),to 6 (paragraph-final boundary (required)).
 - ⑩ tone: A ToBI boundary tone. For breakindex=3, an ip boundary tone (required), and for breakindex>=4, an ip-IP boundary tone combination
 - ⑩ dur: The duration of the pause associated with the boundary, in milliseconds (optional). (if duration is omitted then duration=0 is assumed, i.e. no pause)

In addition to this, the <sentence> element is enhanced by an attribute “src” where an URI of the soundfile produced by the CTS is provided.

Phonetic substructure

Phonetic substructure of words is specified using the following elements:

- ⑩ <word>: The word element has attributes duplicating information also contained in its child elements, for ease of access:
 - ⑩ accent: present if one of the syllables in the word carries a ToBI accent.
- ⑩ <syllable> A <word> consists of a number of <syllable>s. They are required inside word elements. Attributes:
 - ⑩ stress: Indicates whether the syllable carries word stress; possible values are 0 (not stressed), 1 (primary stress), and 2 (secondary stress) (optional; stress=0 is assumed if omitted);
 - accent: present if the syllable carries a ToBI accent (e.g. "h*", "l*+h") (optional);
- ⑩ <ph> A <syllable> consists of a number of phonemes, each of which is characterised by the following attributes²:
 - ⑩ p: the phoneme symbol, following the sampa alphabet (required);
 - ⑩ dur: the duration, in milliseconds (required).

The Gesture Assignment – Animation Generation

Author: Hannes Pirker

² Note, that this <ph> element also can appear under <prosBoundary>s: This allows to distinguish whether a pause in speaking is accompanied with a breathing sound or with silence and allows for the phoneme to viseme procedure to derive the visemes in an uniform way.-

The output of the Gesture Assignment is the last piece of information included in the RRL. This output has to be rendered to a player-specific format (e.g. Shockwave Flash) by an additional component, namely the Animation Generator.

In addition to deciding on the choice and position of “lower-level” gestures, such as gaze and blinking, the GA mainly resolves the “symbolic”-alignment of gestures (encoded by the attributes “alignto” and “aligntype”) to strictly physical measures which are expressed using SMIL-2.0.

GA inserts a new element <animationSpec> at each <act>, where all the information supplied by this module is placed.

<animationSpec> is a repository of empty <sentence> and <gesture> elements, where the temporal relationships are specified using (a subset of) SMIL 2.0. Also the viseme specifications are added here.

General Open Questions

- ⑩ Should there be one Schema for the whole RRL, at all processing steps, or one Schema per interface? In many cases, there would be no problem writing it all into one big Schema: An element can be made optional, but if it is present, it must have certain properties. We have proposed a separation of the Schema into a domain-independent part (neca_rrl.xsd) providing most of the document structure, and a domain-specific part (e.g., neca_eshowroom.xsd) providing domain-specific concepts. Maybe it would make sense to have *language-specific* Schema parts as well? This could contain things like possible values for part-of-speech and syntactic categories.

A full document example: example1_animationinput.rrl

This is the the output for example1 at the end of the processing line.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<necaRRL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://neca.sysis.at/2002/04/necarrl"
        xsi:schemaLocation="http://neca.sysis.at/2002/04/necarrl
                            file:neca_eshowroom.0.4.xsd"
        version="0.4"
        xml:lang="en-US">

  <processingState>
    <state avail="sceneInfoAvailable"/>
    <state avail="mmInfoAvailable"/>
    <state avail="speechInfoAvailable"/>
    <state avail="gestureInfoAvailable"/>
  </processingState>

  <initialCommonGround id="d_1">
    <referent id="x_1"/>
    <unaryCond id="c_1" pred="car" arg="x_1"/>
  </initialCommonGround>
  <participants>
    <person id="ritchie">
      <realname firstname="Ritchie"/>
      <gender type="male"/>
      <appearance character="Ritchie"/>
      <voice name="us2"><prosody rate="-10%" pitch="-20%"/></voice>
      <personality agreeableness="0.8" politeness="polite"/>
      <domainSpecificAttr role="seller"
                          x-position="200" y-position="100"/>
    </person>
    <person id="tina">
      <realname title="Miss" firstname="Tina"/>
      <gender type="female"/>
      <appearance character="Tina"/>
      <voice name="us1"><prosody rate="+10%" pitch="+20%"/></voice>
      <personality agreeableness="0.8" politeness="impolite"/>
      <domainSpecificAttr role="buyer"
                          x-position="100" y-position="100"/>
    </person>
  </participants>

  <temporalOrdering>
    <seq>
      <act id="v_1"/>
      <act id="v_2"/>
    </seq>
  </temporalOrdering>

  <setOfActs>
    <dialogueAct id="v_1">
      <domainSpecificAttr type="requestValue"/>
      <speaker id="tina"/>
      <addressee id="ritchie"/>
      <semanticContent id="d_1">
```

```

    <ternaryCond id="c_2" pred="attribute"
      argOne="x_1" argTwo="consumption" argThree="y"/>
    <free id="c_3" arg="y"/>
  </semanticContent>
  <gesture meaning="takingcommand"
    modality="body" identifier="hips"
    id="g1" alignto="g2" aligntype="par" begin="200"/>
    <!-- this means that, this the gesture starts 200 ms
      after the start of the one it's aligned to -->
  <gesture meaning="establishcontact"
    modality="face" identifier="LookAddressee"
    id="g2" alignto="g3" aligntype="seq_before"/>
  <gesture meaning="deictic" modality="face"
    identifier="LookCar" id="g3" alignto="s1"
    aligntype="seq_before"/>
  <sentence id="s1" src="example1_s1.wav">
    <text>How much fuel does it consume?</text>
    <word id="w_1">How
      <syllable id="syl_1" stress="1">
        <ph dur="67" p="h"/>
        <ph dur="128" p="aU"/>
      </syllable>
    </word>
    <word id="w_2">much
      <syllable accent="H*" id="syl_2" stress="1">
        <ph dur="105" p="m"/>
        <ph dur="67" p="V"/>
        <ph dur="138" p="tS"/>
      </syllable>
    </word>
    <word id="w_3">fuel
      <syllable id="syl_3">
        <ph dur="79" p="f"/>
        <ph dur="59" p="j"/>
        <ph dur="104" p="u"/>
      </syllable>
      <syllable id="syl_4">
        <ph dur="33" p="@"/>
        <ph dur="57" p="l"/>
      </syllable>
    </word>
    <word id="w_4">does
      <syllable id="syl_5" stress="1">
        <ph dur="52" p="d"/>
        <ph dur="123" p="@U"/>
        <ph dur="72" p="z"/>
      </syllable>
    </word>
    <word id="w_5">it
      <syllable accent="H*" id="syl_6" stress="1">
        <ph dur="85" p="I"/>
        <ph dur="57" p="t"/>
      </syllable>
    </word>
    <word id="w_6">consume
      <syllable id="syl_7">
        <ph dur="91" p="k"/>
        <ph dur="27" p="@"/>

```

```

    <ph dur="63" p="n"/>
    <ph dur="94" p="s"/>
  </syllable>
  <syllable id="syl_8">
    <ph dur="101" p="u"/>
    <ph dur="94" p="m"/>
  </syllable>
</word>
<prosBoundary breakindex="4" dur="200" id="b_1" tone="L-L%"/>
<punct id="p_1">?</punct>
</sentence>
<animationSpec>
  <seq>

```

means

```

    <!-- all gestures are performed after another , their
    duration is their "intrinsic duration" whatever this

```

supplied

```

    ... Alternatively a value for "dur" could/SHOULD be

```

```

    -->

```

printed

```

    <!-- "alignto" "aligntype" "meaning" should be of NO
    importance at this place anymore, if they are still

```

```

    here, it's for human-readability/debugging only -->

```

```

    <gesture meaning="takingcommand"
      modality="body" identifier="hips"
      dur="9999"/>
    <gesture meaning="establishcontact"
      modality="face" identifier="LookAddressee"
      dur="9999"/>
    <gesture meaning="deictic" modality="face"
      identifier="LookCar"
      dur="9999"/>

```

```

  <par>
    <!-- after the gestures the speechfile is played -->
    <!-- Note: "src" is of type "xsd:anyURI" thus you
      can either use src="file:example1_s1.mp3" or
      src="example1.mp3"

```

```

    -->

```

```

    <audio src="file:example1_s1.mp3"/>

```

```

  <seq>
    <gesture modality="viseme" dur="67"
      identifier="v_h"/>
    <gesture modality="viseme" dur="128"
      identifier="v_aU"/>
    <gesture modality="viseme" dur="105"
      identifier="v_m"/>
    <gesture modality="viseme" dur="67"
      identifier="v_V"/>
    <gesture modality="viseme" dur="138"
      identifier="v_tS"/>
    <gesture modality="viseme" dur="79"
      identifier="v_f"/>
    <gesture modality="viseme" dur="59"
      identifier="v_j"/>
    <gesture modality="viseme" dur="104"

```

```

        identifier="v_u"/>
    <gesture modality="viseme" dur="33"
        identifier="v_@"/>
    <gesture modality="viseme" dur="57"
        identifier="v_l"/>
    <gesture modality="viseme" dur="52"
        identifier="v_d"/>
    <gesture modality="viseme" dur="123"
        identifier="v_@U"/>
    <gesture modality="viseme" dur="72"
        identifier="v_z"/>
    <gesture modality="viseme" dur="85"
        identifier="v_I"/>
    <gesture modality="viseme" dur="57"
        identifier="v_t"/>
    <gesture modality="viseme" dur="91"
        identifier="v_k"/>
    <gesture modality="viseme" dur="27"
        identifier="v_@"/>
    <gesture modality="viseme" dur="63"
        identifier="v_n"/>
    <gesture modality="viseme" dur="94"
        identifier="v_s"/>
    <gesture modality="viseme" dur="101"
        identifier="v_u"/>
    <gesture modality="viseme" dur="94"
        identifier="v_m"/>
    </seq>
</par>
</seq>
</animationSpec>
</dialogueAct>
<dialogueAct id="v_2">
    <domainSpecificAttr type="inform"/>
    <speaker id="ritchie"/>
    <addressee id="tina"/>
    <semanticContent id="d_2">
        <ternaryCond id="c_3" pred="attribute"
            argOne="x_1" argTwo="consumption" argThree="lph8"/>
    </semanticContent>
    <!-- start LookAddressee at the same time as "s2" -->
    <gesture meaning="establishcontact" modality="face"
        identifier="LookAddressee" id="g4" alignto="s2"
        aligntype="par_start"/>
    <sentence id="s2" src="example1_s2.wav">
        <text>It consumes eight liters per one hundred
kilometers.</text>
        <word id="w_7">It
            <syllable accent="H*" id="syl_9" stress="1">
                <ph dur="80" p="I"/>
                <ph dur="63" p="t"/>
            </syllable>
        </word>
        <word id="w_8">consumes
            <syllable id="syl_10">
                <ph dur="91" p="k"/>
                <ph dur="27" p="@"/>
                <ph dur="63" p="n"/>
            </syllable>
        </word>
    </sentence>

```

```

    <ph dur="94" p="s" />
  </syllable>
  <syllable id="syl_11">
    <ph dur="79" p="u" />
    <ph dur="84" p="m" />
    <ph dur="52" p="z" />
  </syllable>
</word>
<word id="w_9">eight
  <syllable accent="H*" id="syl_12" stress="1">
    <ph dur="209" p="EI" />
    <ph dur="80" p="t" />
  </syllable>
</word>
<word id="w_10">liters
  <syllable accent="!H*" id="syl_13" stress="1">
    <ph dur="72" p="l" />
    <ph dur="128" p="i" />
    <ph dur="75" p="t" />
  </syllable>
  <syllable id="syl_14">
    <ph dur="68" p="r=" />
    <ph dur="66" p="z" />
  </syllable>
</word>
<word id="w_11">per
  <syllable id="syl_15" stress="1">
    <ph dur="94" p="p" />
    <ph dur="59" p="r=" />
  </syllable>
</word>
<word id="w_12">one
  <syllable id="syl_16" stress="1">
    <ph dur="51" p="w" />
    <ph dur="32" p="V" />
    <ph dur="63" p="n" />
  </syllable>
</word>
<word id="w_13">hundred
  <syllable id="syl_17" stress="1">
    <ph dur="61" p="h" />
    <ph dur="45" p="V" />
    <ph dur="39" p="n" />
    <ph dur="34" p="d" />
  </syllable>
  <syllable id="syl_18">
    <ph dur="34" p="r" />
    <ph dur="27" p="@ " />
    <ph dur="16" p="d" />
  </syllable>
</word>
<word id="w_14">kilometers
  <syllable id="syl_19">
    <ph dur="91" p="k" />
    <ph dur="27" p="@ " />
    <ph dur="82" p="l" />
  </syllable>
  <syllable accent="L+H*" id="syl_20" stress="1">

```

```

        <ph dur="108" p="A"/>
        <ph dur="87" p="m"/>
    </syllable>
    <syllable id="syl_21">
        <ph dur="67" p="@"/>
        <ph dur="65" p="t"/>
    </syllable>
    <syllable id="syl_22">
        <ph dur="149" p="r="/>
        <ph dur="156" p="z"/>
    </syllable>
</word>
<prosBoundary breakindex="4" dur="200" id="b_2" tone="L-L%"/>
<punct id="p_2">.</punct>
</sentence>
<!-- smile all the time while speaking -->
<gesture meaning="happyness" modality="face"
    identifier="Smile"
    id="g5" />
<animationSpec>
    <seq>
        <!-- gestures are performed in parallel -->
        <gesture meaning="establishcontact" modality="face"
            identifier="LookAddressee"
            dur="9999"/>
        <!-- speechfile is played and seq-of-visemes are
produced in parallel -->
        <par>
            <audio src="example1_s2.mp3"/>
            <seq>
                <gesture modality="viseme" dur="80"
                    identifier="v_I"/>
                <gesture modality="viseme" dur="63"
                    identifier="v_t"/>
                <gesture modality="viseme" dur="91"
                    identifier="v_k"/>
                <gesture modality="viseme" dur="27"
                    identifier="v_@"/>
                <gesture modality="viseme" dur="63"
                    identifier="v_n"/>
                <gesture modality="viseme" dur="94"
                    identifier="v_s"/>
                <gesture modality="viseme" dur="79"
                    identifier="v_u"/>
                <gesture modality="viseme" dur="84"
                    identifier="v_m"/>
                <gesture modality="viseme" dur="52"
                    identifier="v_z"/>
                <gesture modality="viseme" dur="209"
                    identifier="v_EI"/>
                <gesture modality="viseme" dur="80"
                    identifier="v_t"/>
                <gesture modality="viseme" dur="72"
                    identifier="v_l"/>
                <gesture modality="viseme" dur="128"
                    identifier="v_i"/>
                <gesture modality="viseme" dur="75"
                    identifier="v_t"/>
            </seq>
        </par>
    </seq>

```

```

    <gesture modality="viseme" dur="68"
            identifier="v_r" />
    <gesture modality="viseme" dur="66"
            identifier="v_z" />
    <gesture modality="viseme" dur="94"
            identifier="v_p" />
    <gesture modality="viseme" dur="59"
            identifier="v_r" />
    <gesture modality="viseme" dur="51"
            identifier="v_w" />
    <gesture modality="viseme" dur="32"
            identifier="v_V" />
    <gesture modality="viseme" dur="63"
            identifier="v_n" />
    <gesture modality="viseme" dur="61"
            identifier="v_h" />
    <gesture modality="viseme" dur="45"
            identifier="v_V" />
    <gesture modality="viseme" dur="39"
            identifier="v_n" />
    <gesture modality="viseme" dur="34"
            identifier="v_d" />
    <gesture modality="viseme" dur="34"
            identifier="v_r" />
    <gesture modality="viseme" dur="27"
            identifier="v@" />
    <gesture modality="viseme" dur="16"
            identifier="v_d" />
    <gesture modality="viseme" dur="91"
            identifier="v_k" />
    <gesture modality="viseme" dur="27"
            identifier="v@" />
    <gesture modality="viseme" dur="82"
            identifier="v_l" />
    <gesture modality="viseme" dur="108"
            identifier="v_A" />
    <gesture modality="viseme" dur="87"
            identifier="v_m" />
    <gesture modality="viseme" dur="67"
            identifier="v@" />
    <gesture modality="viseme" dur="65"
            identifier="v_t" />
    <gesture modality="viseme" dur="149"
            identifier="v_r" />
    <gesture modality="viseme" dur="156"
            identifier="v_z" />
  </seq>
</par>
<gesture meaning="happyness" modality="face"
        identifier="Smile"
        alignto="s2" alightype="seq_after"
        dur="9999" />
</seq>
</animationSpec>

</dialogueAct>
</setOfActs>

```

</necaRRL>

The XML Schema

The Schema is separated into a domain-independent part defining the overall structure of an RRL document, and a scenario-dependent part defining relevant concepts and properties for the domain. In addition there is an scenario independent schema where “SMIL2.0-lookalike” elements are defined.

The latest versions of these XML-schemata can be retrieved at the NECA-RRL-page:

<http://www.oefai.at/NECA/RRL>

The domain-independent Schema: neca_rrl.0.4.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsd:schema xmlns="http://neca.sysis.at/2002/04/necarrl"
            targetNamespace="http://neca.sysis.at/2002/04/necarrl"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ssml="http://www.w3.org/2001/10/synthesis"
            xmlns:mary="http://mary.dfki.de/2002/MaryXML"
            elementFormDefault="qualified"
            >

    <xsd:annotation>
    <xsd:documentation>
        This is the schema for the application-independent part of the
necaRRL
        Copyright (C) 2002, 2003 The NECA Consortium. All rights
reserved.
    </xsd:documentation>
    </xsd:annotation>
    <xsd:annotation>
        <xsd:documentation>
            This Schema uses parts of the XML Schema
            draft, SSML 1.0 Core Schema (20021129), for which the
            copyright notice is reproduced here:
            Copyright 1998-2002 W3C (MIT,
            INRIA, Keio), All Rights Reserved. Permission to use,
            copy, modify and distribute the SSML core schema and its
            accompanying documentation for any purpose and without
            fee is hereby granted in perpetuity, provided that the
            above copyright notice and this paragraph appear in all
            copies. The copyright holders make no representation
            about the suitability of the schema for any purpose.
            It is provided "as is" without expressed or implied
            warranty.</xsd:documentation>
        </xsd:documentation>
    </xsd:annotation>

    <xsd:annotation>
    <xsd:documentation>Importing dependent
namespaces</xsd:documentation>
    </xsd:annotation>
```

```

<xsd:import namespace="http://www.w3.org/XML/1998/namespace"
            schemaLocation="http://www.w3.org/2001/xml.xsd"/>

<!-- schema of speech synthesis markup language - SSML -->
<xsd:import namespace="http://www.w3.org/2001/10/synthesis"
            schemaLocation="http://www.w3.org/TR/2002/WD-speech-
synthesis-20021202/synthesis.xsd"/>

<!-- schemaLocation="http://www.w3.org/TR/speech-
synthesis/synthesis.xsd" -->

<!-- schema of markup language - MaryXML -->
<!-- ATTENTION: schema should be placed to the neca server ??? -->
<xsd:import namespace="http://mary.dfki.de/2002/MaryXML"
            schemaLocation="file:MaryXML.xsd"/>

<!-- include schema-parts which define SMIL-2.0 "lookalike"
elements such as <par> and <seq> -->
<!-- ATTENTION: schema should be placed to the neca server ??? -->

<!-- hp 4.2.02 use "redefine" instead of "include" because
neca_smil.xsd has NO namespace
(when using "include" the embedded schema has to have the same
namespace as the embedding one) -->
<xsd:redefine schemaLocation="file:neca_smil.xsd"/>

<!-- The <necaRRL> root element. -->
<xsd:element name="necaRRL">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="processingState"
type="processingType"/>
      <xsd:element name="initialCommonGround" type="drsType"
minOccurs="0"/>
      <xsd:element name="participants"
type="participantsType"/>
      <xsd:element name="temporalOrdering"
type="orderingActsType"/>

      <xsd:element name="setOfActs" type="setOfActsType"
minOccurs="0"/>

    </xsd:sequence>
    <xsd:attribute name="version" use="required" fixed="0.4">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN"/>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute ref="xml:lang"/>

    <!-- attribute "nextProcessingStep" is only kept for
compatibility reasons to v0.3 until neca-komponents are adapted to
rrl0.4 -->
    <xsd:attribute name="nextProcessingStep" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="sceneGenerator"/>
          <xsd:enumeration value="mmGenerator"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

```

```

        <xsd:enumeration value="speechGenerator" />
        <xsd:enumeration value="gestureGenerator" />
        <xsd:enumeration value="animationGenerator" />
    </xsd:restriction>
</xsd:simpleType>
</xsd:attribute>

</xsd:complexType>
</xsd:element>

<!-- The <proprocessingState> element -->
<xsd:complexType name="processingType">
  <xsd:sequence minOccurs="1" maxOccurs="unbounded">
    <xsd:element name="state">
      <xsd:complexType>
        <xsd:attribute name="avail" use="required">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="sceneInfoAvailable" />
              <xsd:enumeration value="mmInfoAvailable" />
              <xsd:enumeration value="speechInfoAvailable" />
              <xsd:enumeration value="gestureInfoAvailable" />
              <xsd:enumeration
value="animationInfoAvailable" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<!-- The <participants> element -->
<xsd:complexType name="participantsType">
  <xsd:sequence>
    <xsd:element name="domainSpecificAttr"
      type="domainSpecificParticipantsType"
minOccurs="0" />
    <!-- One or more <person> elements -->
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="person" type="personType" />
    </xsd:sequence>
  </xsd:sequence>
</xsd:complexType>

<!-- The <person> element. -->
<xsd:complexType name="personType">
  <!-- A person has an *id* attribute, and a number of
    child elements, which may occur in any order. -->
  <xsd:all>
    <xsd:element name="realname" type="realnameType" />
    <xsd:element name="gender" type="genderType" />
    <!-- The visual appearance of the person, by reference
      to a player character -->
    <xsd:element name="appearance">
      <xsd:complexType>

```

```

        <xsd:attribute name="character" use="required"
type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<!-- The voice to use for this person, and optionally the
    default prosody. They are specified using a subset of
    SSML. -->

    <xsd:element name="voice" type="voiceType"/>
    <xsd:element name="personality" type="personalityType"/>
    <xsd:element name="domainSpecificAttr"
        type="domainSpecificPersonAttrType"/>
</xsd:all>
    <xsd:attribute name="id" type="xsd:ID"/>
</xsd:complexType>

<!-- The <realname> element. -->
<xsd:complexType name="realnameType">
    <xsd:attribute name="title" use="optional" type="xsd:string"/>
    <xsd:attribute name="firstname" use="optional"
type="xsd:string"/>
    <xsd:attribute name="middlename" use="optional"
type="xsd:string"/>
    <xsd:attribute name="lastname" use="optional" type="xsd:string"/>
</xsd:complexType>

<!-- The <gender> element. -->
<xsd:complexType name="genderType">
    <xsd:attribute name="type" use="required">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="male"/>
                <xsd:enumeration value="female"/>
                <xsd:enumeration value="neutral"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>

<!-- The <voice> element. -->
<xsd:complexType name="voiceType">
    <xsd:sequence>
        <xsd:element name="prosody" minOccurs="0" maxOccurs="1">
            <xsd:complexType>
                <xsd:attribute name="pitch" use="optional"
type="ssml:pitch.datatype"/>
                <xsd:attribute name="range" use="optional"
type="ssml:range.datatype"/>
                <xsd:attribute name="rate" use="optional"
type="ssml:rate.datatype"/>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- SSML extracts used by <prosody>. -->

```

```

<!-- hp.22/11/2002: I <import> these types from the ssml-schema
instead of copying them -->

<!-- The <personality> element. -->
<xsd:complexType name="personalityType">
  <!-- A <personality> element has attributes
representing personality factors -->
  <xsd:attribute name="extroversion" use="optional"
type="zeroToOneScaleType"/>
  <xsd:attribute name="agreeableness" use="optional"
type="zeroToOneScaleType"/>
  <xsd:attribute name="neuroticism" use="optional"
type="zeroToOneScaleType"/>
  <!-- Until we know better what politeness actually is
(a personality trait? an emotion? an attitude?
something else?), we treat it in the simplest way
possible, namely as a binary personality trait. -->
  <xsd:attribute name="politeness" use="optional">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="polite"/>
        <xsd:enumeration value="impolite"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

<!-- Possible values of the personality scales -->
<xsd:simpleType name="zeroToOneScaleType">
  <!-- Decimal numbers between 0.0 and 1.0 (inclusive)
with maximally 3 digits after the decimal point. -->
  <xsd:restriction base="xsd:decimal">
    <xsd:minInclusive value="0.0"/>
    <xsd:maxInclusive value="1.0"/>
    <xsd:fractionDigits value="3"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- NEW Version: use "smil-related" types <seq> and <par> for
orderingActs -->
<xsd:complexType name="orderingActsType">
  <xsd:annotation>
    <xsd:documentation>
      Temporal ordering: a list /seq/ and /par/ elements,
imposing an order
      on /act/-elements. This can be underspecified in the mnlg-
input, but
      has to be fully specified in the mnlg-output. Pay
attention: this type
      is not entirely "clean" : it allows all substitutions for
the abstract
      element "allowed-within-container".
    </xsd:documentation>
  </xsd:annotation>
  <!-- maxOccurs="unbounded" means, that e.g. several <seq>-
elements

```

```

may appear at top-level: this is not conformant with SMIL, but is
useful for specifying partially underspecified tempoal ordering
-->
<xsd:choice maxOccurs="unbounded">
  <xsd:group ref="smil_structure"/>
</xsd:choice>
</xsd:complexType>

<xsd:annotation>
  <xsd:documentation>
    The ordering of /act/-references under
/orderingReferentsType/
    This element serves for possibly underspecified temporal
ordering
    in M-NLG input.
    /sequence/ element may contain, in any order
    recursively, /sequence/, /parallel/ and /act/ elements.
  </xsd:documentation>
</xsd:annotation>
<xsd:annotation>
  <xsd:documentation>
    an /act/ element has one attribute which is a reference
    to a dialogue act or a non-communicative act.
    The attribute value is syntactically not restrained,
    but should of course reference one of the acts defined in
the
    document.
  </xsd:documentation>
</xsd:annotation>

<!-- <act> - allowed within smil_tags due to
substitutionGroup="awc" -->
<xsd:element name="act" substitutionGroup="awc">
  <xsd:complexType>
    <xsd:attribute name="id" type="xsd:IDREF" use="required"/>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="setOfActsType">
  <xsd:annotation>
    <xsd:documentation>
      An UNordered enumeration of /dialogueAct/s and/or
/nonCommunicativeActs/
    </xsd:documentation>
  </xsd:annotation>
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="dialogueAct"
type="dialogueActType"/>
    <xsd:element name="nonCommunicativeAct"
type="nonCommunicativeActType"/>
  </xsd:choice>
</xsd:complexType>

<!-- ***** dialogue acts and non-communicative acts
***** -->
<!-- The <dialogueAct> element. -->

```

```

<xsd:complexType name="dialogueActType">
  <xsd:sequence>
    <xsd:element name="domainSpecificAttr"
      type="domainSpecificDialogueActAttrType"/>
    <xsd:element name="speaker">
      <xsd:complexType>
        <xsd:attribute name="id" type="xsd:IDREF"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="addressee" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:attribute name="id" type="xsd:IDREF"/>
      </xsd:complexType>
    </xsd:element>
    <!-- Emotion is a set and optional. -->
    <xsd:element name="emotion" type="emotionType"
      minOccurs="0" maxOccurs="1"/>
    <!-- There can be more than one addressee. -->
    <!-- Semantic content is optional -->
    <xsd:element name="semanticContent" type="drsType"
      minOccurs="0" maxOccurs="1"/>

    <!-- Any number of sentences and gestures -->
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <!-- gewagter versuch: hier ssml:sentence erlauben?? -->
      <!-- <xsd:element ref="ssml:sentence"/> -->
      <!-- hp: hat funktioniert, ist aber nicht wirklich
geschickt. Stattdessen wird
"ssml:allowed-within-sentence" IN <sentence> verwendet -->
      <xsd:element name="sentence" type="sentenceType"/>
      <xsd:element name="gesture" type="gestureMnlgType"/>
    </xsd:choice>

    <xsd:element name="animationSpec" type="animationSpecType"
      minOccurs="0" maxOccurs="1"/>

  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID"
use="required"/>
  <xsd:attribute name="reactionTo" type="xsd:IDREF"
use="optional"/>
</xsd:complexType>

<!-- The <emotion> element. is optional -->
<!-- this includes a set of emotionExpressed and emotionFelt -->
<xsd:complexType name="emotionType">
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="emotionExpressed">
        <xsd:complexType>
          <xsd:attributeGroup ref="emotionAttributes"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="emotionFelt">
        <xsd:complexType>
          <xsd:attributeGroup ref="emotionAttributes"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

```

```

        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>

<!-- ***** Common attributes for emotion-types. ***** -->
<xsd:attributeGroup name="emotionAttributes">
    <!-- The person to which this specification belongs -->
    <!-- hp 3.3.2003: isn't "person" redundant ???? --->
    <xsd:attribute name="person" use="required" type="xsd:ID"/>
    <xsd:attribute name="type" use="required">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <!-- These are based on OCC -->
                <!-- In the long run, probably only a subset of these
will be used -->
                <xsd:enumeration value="joy"/>
                <xsd:enumeration value="distress"/>
                <xsd:enumeration value="happy-for"/>
                <xsd:enumeration value="gloating"/>
                <xsd:enumeration value="resentment"/>
                <xsd:enumeration value="sorry-for"/>
                <xsd:enumeration value="hope"/>
                <xsd:enumeration value="fear"/>
                <xsd:enumeration value="satisfaction"/>
                <xsd:enumeration value="relief"/>
                <xsd:enumeration value="fears-confirmed"/>
                <xsd:enumeration value="disappointment"/>
                <xsd:enumeration value="pride"/>
                <xsd:enumeration value="admiration"/>
                <xsd:enumeration value="shame"/>
                <xsd:enumeration value="reproach"/>
                <xsd:enumeration value="liking"/>
                <xsd:enumeration value="disliking"/>
                <xsd:enumeration value="gratitude"/>
                <xsd:enumeration value="anger"/>
                <xsd:enumeration value="gratification"/>
                <xsd:enumeration value="remorse"/>
                <!-- These are added by Elliot -->
                <xsd:enumeration value="love"/>
                <xsd:enumeration value="hate"/>
                <!-- add a "undef" for now -->
                <xsd:enumeration value="undef"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="intensity" type="zeroToOneScaleType"
use="required"/>
    <!-- activation/evaluation/power is an alternative encoding for
the
        emotion-categories, which is used by the emotiona; speech-
synthesis -->
    <xsd:attribute name="activation" type="minusOneToOneScaleType"
use="required"/>
    <xsd:attribute name="evaluation" type="minusOneToOneScaleType"
use="required"/>
    <xsd:attribute name="power" type="minusOneToOneScaleType"
use="required"/>
    <!-- Possible object to which emotion is referring to -->

```

```

    <xsd:attribute name="object"    use="optional"
type="xsd:string"/>
  </xsd:attributeGroup>

  <!-- minusOneToOneScaleType: Possible values on emotion dimensions
-->
  <xsd:simpleType name="minusOneToOneScaleType">
    <!-- Decimal numbers between -1.0 and 1.0 (inclusive)
        with maximally 3 digits after the decimal point. -->
    <xsd:restriction base="xsd:decimal">
      <xsd:minInclusive value="-1.0"/>
      <xsd:maxInclusive value="1.0"/>
      <xsd:fractionDigits value="3"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- The <nonCommunicativeAct> element. -->
  <xsd:complexType name="nonCommunicativeActType">
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <!-- currently nothing specified here. -->
  </xsd:complexType>

  <!-- ***** discourse representation structures
***** -->
  <!-- The content of <initialCommonGround> and <semanticContent>
        elements. -->
  <xsd:complexType name="drsType">
    <xsd:sequence>
      <!-- Zero to many <referent> elements -->
      <xsd:element name="referent" minOccurs="0"
maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:attribute name="id" type="referentType"/>
        </xsd:complexType>
      </xsd:element>
      <!-- followed by one or many conditions out of the following: -
->
      <xsd:choice maxOccurs="unbounded">
        <xsd:element name="unaryCond">
          <xsd:complexType>
            <xsd:attribute name="id" type="xsd:string"/>
            <xsd:attribute name="pred" type="unaryPredicateType"/>
            <xsd:attribute name="arg" type="termType"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="binaryCond">
          <xsd:complexType>
            <xsd:attribute name="id" type="xsd:string"/>
            <xsd:attribute name="pred" type="binaryPredicateType"/>
            <xsd:attribute name="argOne" type="termType"/>
            <xsd:attribute name="argTwo" type="termType"/>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="ternaryCond">
          <xsd:complexType>
            <xsd:attribute name="id" type="xsd:string"/>

```

```

        <xsd:attribute name="pred" type="ternaryPredicateType"/>
        <xsd:attribute name="argOne" type="termType"/>
        <xsd:attribute name="argTwo" type="termType"/>
        <xsd:attribute name="argThree" type="termType"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="conditional">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="argOne" type="drsType"/>
            <xsd:element name="argTwo" type="drsType"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="negation" type="drsType"/>
<xsd:element name="dTopic">
    <xsd:complexType>
        <xsd:attribute name="id" type="xsd:string"/>
        <xsd:attribute name="arg" type="termType"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="free">
    <xsd:complexType>
        <xsd:attribute name="id" type="xsd:string"/>
        <xsd:attribute name="arg" type="referentType"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="bridge" type="drsType"/>
</xsd:choice>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- a term -->
<xsd:simpleType name="termType">
    <xsd:union memberTypes="drtConstantType referentType"/>
</xsd:simpleType>

<!-- a referent -->
<xsd:simpleType name="referentType">
    <!-- restrict referent names to "x", "y" or "z",
         optionally followed by "_" and a number: -->
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[xyz](_\d+)?"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- ***** The <sentence> element. ***** -->
<xsd:complexType name="sentenceType" mixed="true">
    <xsd:sequence>
        <!-- First, the textual representation of the entire sentence.
-->
        <xsd:element name="text" type="xsd:string" minOccurs="0"/>

        <!-- Then, its structure. -->
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:group ref="sentenceInternalStructuralElements"/>

```

```

        <xsd:group ref="wordLevelElements"/>
        <!-- allowing all elements also allowed in <ssml:sentence>
-->
        <xsd:group ref="ssml:allowed-within-sentence"/>
        <!-- allowing all elements also allowed in
<mary:sentence> -->
        <xsd:group ref="mary:allowed-within-sentence"/>
        </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" use="optional"/>
    <xsd:attribute name="src" type="xsd:anyURI"/>
    <!-- Also allow align-info at the sentence level -->
    <xsd:attributeGroup ref="alignAttr"/>
</xsd:complexType>

<!-- ***** Sentence-internal structural elements. ***** -
->
<xsd:group name="sentenceInternalStructuralElements">
    <xsd:choice>
        <xsd:element name="infoStruct" type="infoStructType"/>
        <xsd:element name="synPhrase" type="synPhraseType"/>
        <xsd:element name="infoStatus" type="infoStatusType"/>
    </xsd:choice>
</xsd:group>

<!-- The <infoStruct> element. -->
<xsd:complexType name="infoStructType">
    <xsd:choice maxOccurs="unbounded">
        <xsd:group ref="sentenceInternalStructuralElements"/>
        <xsd:group ref="wordLevelElements"/>
    </xsd:choice>
    <xsd:attribute name="id" type="xsd:ID" use="optional"/>
    <xsd:attribute name="part">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <!-- as <theme> is always comprising the whole sentence
                     it can be omitted in the encoding -->
                <xsd:enumeration value="theme"/>
                <xsd:enumeration value="rheme"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>

<!-- The <synPhrase> element. -->
<xsd:complexType name="synPhraseType">
    <xsd:choice maxOccurs="unbounded">
        <xsd:group ref="sentenceInternalStructuralElements"/>
        <xsd:group ref="wordLevelElements"/>
    </xsd:choice>
    <!-- The syntactic phrase category -->
    <xsd:attribute name="id" type="xsd:ID" use="optional"/>
    <xsd:attribute name="category" type="syntacticCategoryType"
        use="required"/>
    <xsd:attribute name="function" type="grammaticalFunctionType"
        use="required"/>
</xsd:complexType>

```

```

<!-- The <infoStatus> element. -->
<xsd:complexType name="infoStatusType">
  <xsd:choice maxOccurs="unbounded">
    <xsd:group ref="sentenceInternalStructuralElements"/>
    <xsd:group ref="wordLevelElements"/>
  </xsd:choice>
  <xsd:attribute name="id" type="xsd:ID" use="optional"/>
  <xsd:attribute name="type" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <!-- The possible values for the type attribute: -->
        <xsd:enumeration value="referent-given"/>
        <xsd:enumeration value="concept-given"/>
        <xsd:enumeration value="constrast"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>

<!-- ***** Word-level elements. ***** -->
<xsd:group name="wordLevelElements">
  <xsd:choice>
    <xsd:element name="gesture" type="gestureMnlgType"/>
    <xsd:element name="word" type="wordType"/>
    <xsd:element name="punct" type="punctType"/>
    <xsd:element name="prosBoundary" type="prosBoundaryType"/>
  </xsd:choice>
</xsd:group>

<!-- <gesture> element: is part of substitutionGroup for SMIL-
content -->
<!--
  NOTE: there are TWO different <gesture> elements:
  1) <gesture> of type gestureMnlgType -> to be used within the
m-nlg    contains alignto/aligntype attributes
  2)      of type gestureAnimType -> to be used in the
animationSpec: contains smil attributes
-->
<xsd:element name="gesture" type="gestureAnimType"
substitutionGroup="awc"/>

<!-- <audio> element: is part of substitutionGroup for SMIL-content
-->
<xsd:element name="audio" substitutionGroup="awc">
  <xsd:complexType>
    <xsd:attribute name="src" type="xsd:anyURI"/>
    <xsd:attributeGroup ref="smil_timingAttributes"/>
  </xsd:complexType>
</xsd:element>

<!-- The <word> element. -->
<xsd:complexType name="wordType" mixed="true">
  <xsd:sequence>
    <xsd:element name="syllable" minOccurs="0"
maxOccurs="unbounded"
type="syllableType"/>
  </xsd:sequence>

```

```

<!-- Word attributes: -->
<xsd:attribute name="id" type="xsd:ID" use="optional"/>
<!-- Textual representation -->
<!-- Not used any more!
<xsd:attribute name="text" type="xsd:string" use="required"/>
-->
<!-- Part-of-speech -->
<xsd:attribute name="pos" type="posType" use="optional"/>
<!-- Phonemic transcription -->
<xsd:attribute name="sampa" type="xsd:string" use="optional"/>
<!-- Grammatical function -->
<xsd:attribute name="function" type="grammaticalFunctionType"
    use="optional"/>
<!-- ToBI accent -->
<xsd:attribute name="accent" type="xsd:string" use="optional"/>
</xsd:complexType>

<!-- The <punct> (punctuation) element. -->
<xsd:complexType name="punctType" mixed="true">
  <xsd:attribute name="id" type="xsd:ID" use="optional"/>
  <!-- not used anymore
  <xsd:attribute name="text" type="xsd:string"/>
  -->
</xsd:complexType>

<!-- attributeGroup for <gesture> -->
<xsd:attributeGroup name="gestureAttr">
  <xsd:attribute name="modality" use="optional">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="voice"/>
        <xsd:enumeration value="face"/>
        <xsd:enumeration value="body"/>
        <xsd:enumeration value="viseme"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <!-- "meaning" is intended to be used by GA: when (e.g. timing-)
restrictions imply that a gesture
  is not suitable, GA may select another one with identical
meaning -->
  <xsd:attribute name="meaning" type="xsd:string"
use="optional"/>
  <xsd:attribute name="identifier" type="xsd:string"
use="required"/>
</xsd:attributeGroup>

<xsd:attributeGroup name="alignAttr">
  <!-- align-attributes: make them "required" later? -->
  <xsd:attribute name="alignto" type="xsd:IDREF"
use="optional"/>
  <!-- alternatively alignstart and alignend can be used in order
to align to a RANGE of tags -->
  <xsd:attribute name="alignstart" type="xsd:IDREF"
use="optional"/>
  <xsd:attribute name="alignend" type="xsd:IDREF"
use="optional"/>

```

```

    <!-- aligntype specifies the temporal relation between gesture
and the "alignto"-element -->
    <xsd:attribute name="aligntype" type="alignType"
use="optional"/>
    <!-- begin, end, dur -->
</xsd:attributeGroup>

<!-- The <gesture> element.produced by M-NLG etc. -->
<xsd:complexType name="gestureMnlgType">
  <xsd:attribute name="id" type="xsd:ID" use="optional"/>
  <xsd:attributeGroup ref="gestureAttr"/>
  <xsd:attributeGroup ref="alignAttr"/>
</xsd:complexType>

<!-- The <gesture> element used in under <animationSpec>:
alignAttr is missing here
"id" is IDREF instead of ID : it is only included for debugging
purposes -->
<xsd:complexType name="gestureAnimType">
  <xsd:attribute name="id" type="xsd:IDREF" use="optional"/>
  <xsd:attributeGroup ref="gestureAttr"/>
  <xsd:attributeGroup ref="smil_timingAttributes"/>
</xsd:complexType>

<!-- animation type: holds all the information for the animation-
timing
within a dialogueAct: it now allows smil-lookalikes <par> and
<seq> with embedded <gestures> -->
<xsd:complexType name="animationSpecType">
  <xsd:choice>
    <xsd:group ref="smil_structure"/>
  </xsd:choice>
</xsd:complexType>

<!-- the aligntype-attribute -->
<!-- what's about sequences of <gestures> like in the current
eshowroom? Then alignwith must be another gesture -->
<xsd:simpleType name="alignType">
  <xsd:restriction base="xsd:string">
    <xsd:annotation>
      <xsd:documentation>
        How are gesture G and "alignto" element X coupled to
each other?
        PARALELLISM
        par_start : G and X start at the same time
        par_end : G and X stops at the same time

        par_adjust_to_fit (formerly: par_stretch)
        : G's duration is forced to be the same as X's : they
start and stop at the same time

        atstress: G is aligned to the STRESSED position of X

        SEQUENTIALITY
        seq_before : G is performed before X --> precede
        seq_after : G is performed after X --> succede
      </xsd:documentation>
    </xsd:annotation>
  </xsd:restriction>
</xsd:simpleType>

```

```

    <!-- parallelism -->
    <!-- par_start -->
    <xsd:enumeration value="par" />
    <xsd:enumeration value="par_start" />
    <xsd:enumeration value="par_end" />
    <xsd:enumeration value="par_stretch" />
    <!-- a better name for par_stretch ??? -->
    <xsd:enumeration value="par_adjust_to_fit" />
    <!-- sequentiality -->
    <xsd:enumeration value="seq_before" />
    <xsd:enumeration value="seq_after" />
    <!-- or do you prefer the following tags instead ?? -->
    <xsd:enumeration value="preceede" />
    <xsd:enumeration value="succeede" />

    <xsd:enumeration value="atstress" />
    <!-- "none" could be used when a <gesture> is alone in a
speech-act (or is the first one in a sequence of <gesture>s -->
    <xsd:enumeration value="none" />
  </xsd:restriction>
</xsd:simpleType>

<!-- The <prosBoundary> element. -->
<xsd:complexType name="prosBoundaryType">
  <xsd:sequence>
    <!-- prosBoundary may contain PHONEMES - typically "_" but also
breathing might be specified -->
    <xsd:element ref="ph" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID" use="optional" />
  <xsd:attribute name="tone" type="xsd:string" use="required" />
  <xsd:attribute name="breakindex" type="xsd:positiveInteger"
    use="required" />
  <xsd:attribute name="dur" type="xsd:nonNegativeInteger"
default="0" use="optional" />
</xsd:complexType>

<!-- phoneme-element -->
<xsd:element name="ph" type="phType" />

<!-- ***** Elements below word level ***** -->
<!-- The <syllable> element. -->
<xsd:complexType name="syllableType">
  <xsd:sequence>
    <!-- The phonemes composing the syllable -->
    <xsd:element ref="ph" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:ID" use="optional" />
  <xsd:attribute name="stress" default="0">
    <xsd:simpleType>
      <xsd:restriction base="xsd:nonNegativeInteger">
        <xsd:maxInclusive value="2" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="accent" type="xsd:string" use="optional" />
</xsd:complexType>

```

```

<!-- The <ph> (phoneme) element. -->
<xsd:complexType name="phType">
  <!-- The phoneme symbol -->
  <xsd:attribute name="p" type="xsd:string" use="required"/>
  <!-- The phoneme duration, in milliseconds -->
  <xsd:attribute name="dur" type="xsd:positiveInteger"
use="required"/>
  <xsd:attribute name="id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<!-- ***** Possible values for syntactic tags ***** -->
<!-- Part of speech tags -->
<xsd:simpleType name="posType">
  <xsd:restriction base="xsd:string">
    <!-- add the list of possible values here -->
  </xsd:restriction>
</xsd:simpleType>

<!-- Syntactic phrase categories -->
<xsd:simpleType name="syntacticCategoryType">
  <xsd:restriction base="xsd:string">
    <!-- add the list of possible values here -->
  </xsd:restriction>
</xsd:simpleType>

<!-- Grammatical functions -->
<xsd:simpleType name="grammaticalFunctionType">
  <xsd:restriction base="xsd:string">
    <!-- add the list of possible values here -->
  </xsd:restriction>
</xsd:simpleType>

</xsd:schema>

```

The domain dependent schema neca_socialite.0.4.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsd:schema targetNamespace="http://neca.sysis.at/2002/04/necarr1"
  xmlns="http://neca.sysis.at/2002/04/necarr1"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  >

  <!-- include the domain-independent part of the Schema -->
  <xsd:include schemaLocation="file:neca_rrl.0.4.xsd"/>

  <!-- hp.12.11.2002: created on basis of "neca_eshowroom.0.3.xsd" -
  -->

  <xsd:annotation>
    <xsd:documentation>
      This is the schema for the scenario dependent part of the
      necaRRL
      Copyright (C) 2002, 2003 The NECA Consortium. All rights
      reserved.
    </xsd:documentation>
  </xsd:annotation>

  <!-- *****
  -->
  <!-- ***** scenario-specific content definitions *****
  -->
  <!-- *****
  -->

  <!-- as long as we do not know how/if to specify "overlap" it is
  of type "string" -->
  <xsd:complexType name="domainSpecificParticipantsType">
    <xsd:attribute name="bodyoverlap" type="xsd:string"
    use="optional"/>
  </xsd:complexType>

  <!-- A <person>'s <domainSpecificAttr> element.
  It has a *role* attribute -->
  <xsd:complexType name="domainSpecificPersonAttrType">
    <xsd:attribute name="role" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="storyteller"/>
          <xsd:enumeration value="listener"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <!-- A <person>'s X position relative to window.screenLeft -->
    <!-- <xsd:attribute name="x-position"
    type="xsd:positiveInteger" use="optional"/> -->
    <!-- A <person>'s Y position relative to window.screenTop -->
    <!-- <xsd:attribute name="y-position" type="xsd:positiveInteger"
    use="optional"/> -->
  </xsd:complexType>
```

```

    <!-- A <dialogueAct>'s <domainSpecificAttr> element.
         Here, it has a *type* attribute. -->

    <!-- OPENQUESTION: where and how to put speech-relevant subtypes
         by Martine and Stefan -->

    <!--
<xsd:restriction base="xsd:string">
  <xsd:enumeration value="WQ"/>
  <xsd:enumeration value="inititiate"/>
  <xsd:enumeration value="unfinished"/>
</xsd:restriction>

greeting (initiate, answer)
question (WQ, default)
assertion (unfinished, default)
-->

    <xsd:complexType name="domainSpecificDialogueActAttrType">
      <!-- as long as not defined: speaker-orientation is of type
"string" -->
      <xsd:attribute name="speaker-orientation" type="xsd:string"
default="front" use="optional"/>
      <xsd:attribute name="type" use="optional">
        <!-- "type" is the name for the "macro structure" used in
Sysis's XML-based generator -->
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="requestJudgeAction"/>
            <xsd:enumeration value="reportSympathyLevel"/>
            <xsd:enumeration value="backchannel"/>
            <xsd:enumeration value="continueStory"/>
            <xsd:enumeration value="elaborateStory"/>
            <xsd:enumeration value="familiarityLevelAction"/>
            <xsd:enumeration value="familiarityLevelPerson"/>
            <xsd:enumeration value="futureAction"/>
            <xsd:enumeration value="goodBye"/>
            <xsd:enumeration value="greeting"/>
            <xsd:enumeration value="initiateStory"/>
            <xsd:enumeration value="judgeAction"/>
            <xsd:enumeration value="opening"/>
            <xsd:enumeration value="prepareGoodBye"/>
            <xsd:enumeration value="reportMood"/>
            <xsd:enumeration value="requestBackchannel"/>
            <xsd:enumeration value="requestReportMood"/>
            <xsd:enumeration value="requestSympathyLevel"/>
            <xsd:enumeration value="response"/>
            <xsd:enumeration value="specifyTopic"/>
            <!-- These are NOT found in the specification, only in
Brig's example : is it a bug or a frature ??? -->
            <xsd:enumeration value="disagree"/>
            <xsd:enumeration value="requestMoreDetail"/>
            <xsd:enumeration value="specifyRelationship"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>

```

```

    <xsd:attribute name="subtype" use="optional" default="none">
      <!-- "subtype" holds the "refining info" for the "macro
structure" used in Sysis's XML-based generator -->
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="none"/>
          <xsd:enumeration value="positive"/>
          <xsd:enumeration value="negative"/>
          <xsd:enumeration value="indifferent"/>
          <xsd:enumeration value="neutral"/>
          <xsd:enumeration value="question"/>
          <xsd:enumeration value="reportLow"/>
          <xsd:enumeration value="reportMedium"/>
          <xsd:enumeration value="reportNo"/>
          <xsd:enumeration value="reportYes"/>
          <xsd:enumeration value="request"/>
          <xsd:enumeration value="response"/>
          <xsd:enumeration value="self"/>
          <xsd:enumeration value="other"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="speechtype" use="optional" default="none">
      <!-- "speechtype" is a classification demanded by Stefan
and Martine in order to faciliatate the generation of
appropriate intonation -->
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="none"/>
          <xsd:enumeration value="greeting(initiate)"/>
          <xsd:enumeration value="greeting(answer)"/>
          <xsd:enumeration value="question(WQ)"/>
          <xsd:enumeration value="question(default)"/>
          <xsd:enumeration value="assertion(unfinished)"/>
          <xsd:enumeration value="assertion(default)"/>
          <!-- allowing "default" for now -->
          <xsd:enumeration value="default"/>

          <!--
          <xsd:enumeration value="none"/>
          <xsd:enumeration value="wq"/>
          <xsd:enumeration value="initiate"/>
          <xsd:enumeration value="answer"/>
          <xsd:enumeration value="unfinished"/>
          -->
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>

  <!-- *** The semantic "vocabulary" *** -->

  <!-- hp. 13.11.2002: though DRSS and semantic-content are not used
in
Socialite right now, some of the types still have to be defined here,
otherwise validation of the neca_rrl-scheme will fail because of
missing types -->

```

```

<!-- drt constants -->
<xsd:simpleType name="drtConstantType">
  <xsd:restriction base="xsd:string">
    <!-- Car attributes -->
    <xsd:enumeration value="price"/>
    <xsd:enumeration value="consumption"/>
    <xsd:enumeration value="etc." />
  </xsd:restriction>
</xsd:simpleType>

<!-- unary predicates -->
<xsd:simpleType name="unaryPredicateType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="car"/>
    <xsd:enumeration value="cheap"/>
    <xsd:enumeration value="etc." />
  </xsd:restriction>
</xsd:simpleType>

<!-- binary predicates -->
<xsd:simpleType name="binaryPredicateType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="have"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- ternary predicates -->
<xsd:simpleType name="ternaryPredicateType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="attribute"/>
  </xsd:restriction>
</xsd:simpleType>

</xsd:schema>

```

The domain dependent schema *neca_eshowroom.0.4.xsd*

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://neca.sysis.at/2002/04/necarr1"
            targetNamespace="http://neca.sysis.at/2002/04/necarr1"
            elementFormDefault="qualified"
            >

  <!-- Include the domain-independent part of the Schema -->
  <xsd:redefine schemaLocation="file:neca_rr1.0.4.xsd"/>

  <xsd:annotation>
    <xsd:documentation>
      This is the schema for the scenario dependent part of the
      necaRRL
      Copyright (C) 2002, 2003 The NECA Consortium. All rights
      reserved.
    </xsd:documentation>
  </xsd:annotation>

  <!-- *****
  -->
  <!-- ***** scenario-specific content definitions *****
  -->
  <!-- *****
  -->

  <!-- at the moment domainSpecificParticipantsType" is not used in
  eshowroom -->
  <xsd:complexType name="domainSpecificParticipantsType">
    </xsd:complexType>

  <!-- A <person>'s <domainSpecificAttr> element.
  It has a *role* attribute -->
  <xsd:complexType name="domainSpecificPersonAttrType">
    <xsd:attribute name="role" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="seller"/>
          <xsd:enumeration value="buyer"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <!-- A <person>'s X position relative to window.screenLeft -->
    <xsd:attribute name="x-position" type="xsd:positiveInteger"
    use="required"/>
    <!-- A <person>'s Y position relative to window.screenTop -->
    <xsd:attribute name="y-position" type="xsd:positiveInteger"
    use="required"/>
  </xsd:complexType>

  <!-- A <dialogueAct>'s <domainSpecificAttr> element.
  Here, it has a *type* attribute. -->
  <xsd:complexType name="domainSpecificDialogueActAttrType">
```

```

<xsd:attribute name="type">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="openingComplaint" />
      <xsd:enumeration value="openingComplaintResponse" />
      <xsd:enumeration value="greeting" />
      <xsd:enumeration value="openingQuestion" />
      <xsd:enumeration value="openingResponse" />
      <xsd:enumeration value="requestInfo" />
      <xsd:enumeration value="inform" />
      <xsd:enumeration value="positiveEvaluation" />
      <xsd:enumeration value="negativeEvaluation" />
      <xsd:enumeration value="requestIf" />
      <xsd:enumeration value="requestValue" />
      <xsd:enumeration value="positiveResponse" />
      <xsd:enumeration value="negativeResponse" />
      <xsd:enumeration value="initiateClosingPositive" />
      <xsd:enumeration value="completeClosingPositive" />
      <xsd:enumeration value="initiateClosingNegative" />
      <xsd:enumeration value="completeClosingNegative" />
      <xsd:enumeration value="refuseAnswer" />
      <xsd:enumeration value="refuseAnswerResponse" />
      <xsd:enumeration value="confirm" />
      <xsd:enumeration value="disconfirm" />
      <xsd:enumeration value="agree" />
      <xsd:enumeration value="disagree" />
      <xsd:enumeration value="feedback" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
</xsd:complexType>

<!-- *** The semantic "vocabulary" *** -->
<!-- drt constants -->
<xsd:simpleType name="drtConstantType">
  <xsd:restriction base="xsd:string">
    <!-- Car attributes -->
    <xsd:enumeration value="price" />
    <xsd:enumeration value="consumption" />
    <xsd:enumeration value="luggage_compartment" />
    <xsd:enumeration value="interior" />
    <xsd:enumeration value="horsepower" />
    <xsd:enumeration value="recyclable_materials" />
    <xsd:enumeration value="airbags" />
    <xsd:enumeration value="anti-lock_brakes" />
    <xsd:enumeration value="broad_tires" />
    <xsd:enumeration value="maxspeed" />
    <xsd:enumeration value="power_windows" />
    <xsd:enumeration value="leather_seats" />
    <xsd:enumeration value="catalytic_converter" />
    <!-- Car attribute values -->
    <xsd:enumeration value="true" />
    <xsd:enumeration value="false" />
    <xsd:enumeration value="hp80" />
    <xsd:enumeration value="mph110" />
    <xsd:enumeration value="eur25000" />
    <xsd:enumeration value="spacious" />
    <xsd:enumeration value="lph8" />
  </xsd:restriction>
</xsd:simpleType>

```

```

    </xsd:restriction>
</xsd:simpleType>

<!-- unary predicates -->
<xsd:simpleType name="unaryPredicateType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="car" />
    <xsd:enumeration value="cheap" />
    <xsd:enumeration value="economical" />
    <xsd:enumeration value="comfortable" />
    <xsd:enumeration value="prestigious" />
    <xsd:enumeration value="sporty" />
    <xsd:enumeration value="nonpolluting" />
    <xsd:enumeration value="safe" />
    <xsd:enumeration value="spacious" />
  </xsd:restriction>
</xsd:simpleType>

<!-- binary predicates -->
<xsd:simpleType name="binaryPredicateType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="have" />
  </xsd:restriction>
</xsd:simpleType>

<!-- ternary predicates -->
<xsd:simpleType name="ternaryPredicateType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="attribute" />
  </xsd:restriction>
</xsd:simpleType>

</xsd:schema>

```

The auxiliary schema *neca_smil.xsd*

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:annotation>
    <xsd:documentation>
      hp. 25.11.2002 The purpose of this scheme is to define
      'smil-related' elements (i.e. 'par' and 'seq' to be used within
      neca. This might be replaced by a REAL integration of smil20
one
      day.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="struct" abstract="true"/>
  <xsd:annotation>
    <xsd:documentation>The 'smil_structure'
      group uses this abstract element. Elements with 'struct' as
      their substitution class are then alternatives for
      'structrure'. Currently these are 'seq' and
'par'</xsd:documentation>
  </xsd:annotation>

  <xsd:group name="smil_structure">
    <xsd:sequence>
      <xsd:element ref="struct"/>
    </xsd:sequence>
  </xsd:group>

  <xsd:element name="awc" abstract="true">
    <xsd:annotation>
      <xsd:documentation>The 'allowed-within-container'
        group uses this abstract element. Elements with awc as
        their substitution class are then alternatives for
        'allowed-within-container'.
        These are the 'leafs' allowed in the smil_structure
elements</xsd:documentation>
    </xsd:annotation>
  </xsd:element>

  <xsd:group name="allowed-within-container">
    <xsd:choice>
      <xsd:element ref="awc"/>
    </xsd:choice>
  </xsd:group>

  <!-- smil_structure - elements -->
  <xsd:element name="par" type="parType" substitutionGroup="struct"/>
  <xsd:complexType name="parType">
    <xsd:choice maxOccurs="unbounded">
      <xsd:group ref="smil_structure"/>
      <xsd:group ref="allowed-within-container"/>
    </xsd:choice>
    <xsd:attributeGroup ref="smil_timingAttributes"/>
  </xsd:complexType>

```

```

</xsd:complexType>

<xsd:element name="seq" type="seqType" substitutionGroup="struct"/>
<xsd:complexType name="seqType">
  <xsd:choice maxOccurs="unbounded">
    <xsd:group ref="smil_structure"/>
    <xsd:group ref="allowed-within-container"/>
  </xsd:choice>
  <xsd:attributeGroup ref="smil_timingAttributes"/>
</xsd:complexType>

<!-- allowed-within-container - elements -->
<!-- E.g. <gesture> , <act>, <audio> and <viseme> : these will be
defined in the rrl-schema -->

<xsd:attributeGroup name="smil_timingAttributes">
  <xsd:annotation>
    <xsd:documentation> Timing attributes for smil : at the moment
only integer are allowed, which are interpreted as [ms] Later
on
  values like "3.2s" or "25ms" could be allowed
  </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="dur"      type="xsd:nonNegativeInteger"
use="optional"/>
  <xsd:attribute name="begin"   type="xsd:integer"
default="0"      use="optional"/>
  <xsd:attribute name="end"     type="xsd:nonNegativeInteger"
use="optional"/>
</xsd:attributeGroup>

</xsd:schema>

```