Cross-media agent platform

Radosław Niewiadomski*
Telecom ParisTech, Paris, France
Elisabetta Bevacqua †
CNRS - Telecom ParisTech, Paris, France
Le Quoc Anh ‡
Telecom ParisTech, Paris, France
Mohammad Obaid §
HITLab NZ, University of Canterbury
Christchurch, New Zealand
Julian Looser ¶
HITLab NZ, University of Canterbury
Christchurch, New Zealand
Catherine Pelachaud ‖
CNRS - Telecom ParisTech
Paris, France

Abstract

We have developed a general purpose use and modular architecture of an embodied conversational agent (ECA). Our agent is able to communicate using verbal and nonverbal channels like gaze, facial expressions, and gestures. Our architecture follows the SAIBA framework that sets 3-step process and communication protocols. In our implementation of SAIBA architecture we focus on flexibility and we introduce different levels of the customization. In particular, our system is able to display the same communicative intention with different embodiments, be a virtual agent or a robot. Moreover our framework is independent of the animation player technology. Agent animations can be displayed across different medias, such as web browser, virtual or augmented reality. In this paper we present our agent architecture and its main features.

CR Categories: H.5.2.f [Information Technology and Systems]: Information Interfaces and Representation (HCI) — User Interfaces Graphical user interfaces; H.5.1.b [Information Technology and Systems]: Information Interfaces and Representation (HCI) — Multimedia Information Systems Artificial, augmented, and virtual realities

Keywords: embodied conversational agents, web, augmented reality, SAIBA, BML, HCI, HRI

1 Introduction

The recent technological progress made the creation of virtual humanoids possible. They can serve as interface to computer systems; they can be self-representation of users; they can be controlled by a human or be fully autonomous. In this paper we are interested in autonomous virtual humanoids. When these agents are endowed with expressive communicative capabilities, they are called Embodied Conversational Agents (ECA). An ECA is a computer-generated animated character that is able to carry on natural, human-like communication with users. For this purpose agent systems have been developed to simulate verbal and nonverbal communicative behaviors. ECAs can dialogue with users using synthesized speech, gestures, gaze and facial expressions. These agents are powerful as HCI metaphor [Reeves and Nass 1996]. Studies have shown that with appropriate behaviors they can enhance the experience users have when interacting with a system; users enjoyed it more, feel more engaged, learn more, etc. [Von der Pütten et al. 2010].

This paper presents the real-time architecture of our embodied conversational agent. It is an example of ECA system based on a common framework. Our agent architecture follows the design methodology proposed by Thorisson et al. [Thórisson et al. 2005] and it is compatible with the SAIBA framework [Vilhjálmsson et al. 2007] (see Section 2 for details). Our system can be used for different embodiments (e.g. robots, virtual agents), different virtual presentations (e.g. 2D Flash based or 3D Ogre based presentation) or different media of the same embodiment (virtual reality, augmented reality, web-based application).
After presenting the main features of our agent platform, we focus on different possible realizations of the same behavior with our SAIBA based system. We present a practical realization of SAIBA hierarchical architecture. By separating the behavior planning process from its realization our system is able to communicate the same intention through different embodiments or different display media. For this purpose distinctive behavior planning strategies can be easily defined to communicate the same intention. We can use the same behavior planning module for different embodiments taking into account technological constraints of robots or virtual embodiments. Since the behavior planning module is capable of producing distinctive behaviors, it can be also used to create agents with different behavioral tendencies such as "expansive" agent.
In our architecture modules communicate with each other using symbolic description encoded in XML languages. These languages are independent of the agent embodiment (e.g. virtual or robot) and the agent player technology (e.g. web, 3D, 2D). Our architecture is highly flexible: when applied to new agents, most of the platform can be reused. For each agent only their behavioral specification and player animation need to be customized. Throughout the paper we will present implementation details to highlight the proprieties of our agent architecture.

This paper is structured as follows. The next section is dedicated to an overview of different existing architectures of embodied conversational agents and to the presentation of leading standardization initiatives. Then, in Section 3 our implementation of the SAIBA architecture is provided whereas in Section 4 we present the agent behavior specification. In Section 5 we explain how the system selects the behavioral signals to communicate the agent's intentions while in Sections 6 and 7 we show how these behaviors can be realized for different embodiments or displayed on different media. Finally we conclude the paper in Section 8.

2 State of the art

Building an ECA system needs the involvement of many research disciplines. Issues like speech recognition, motion capture, dialog management, or animation rendering require different skills from their designers. Soon, it became obvious that there was the need to share expertise and to exchange components of an ECA system. SAIBA [Vilhjálmsson et al. 2007] is an international research ini-

*e-mail: niewiado@telecom-paristech.fr
†e-mail: bevacqua@telecom-paristech.fr
‡e-mail: quoc@telecom-paristech.fr
§e-mail: mohammad.obaid@hitlabnz.org
¶e-mail: julian@julianlooser.com
‖e-mail: pelachaud@telecom-paristech.fr

I3D 2011, Paris, France, June 20 – 22, 2011.
© 2011 ACM 978-1-4503-0774-1/11/0006 $10.00

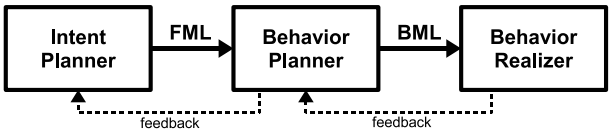

Figure 1: SAIBA framework.
[Vilhjálmsson et al. 2007]

tiative whose main aim is to define a standard framework for the generation of virtual agent behavior. It defines a number of levels of abstraction, from the computation of the agent's communicative intention, to behavior planning and realization, as shown in Figure 1. The **Intent Planner** module decides the agent's current goals, emotional state and beliefs, and encodes them into the Function Markup Language (FML) [Heylen et al. 2008]. To convey the agent's communicative intentions, the **Behavior Planner** module schedules a number of communicative signals with the Behavior Markup Language (BML). It specifies the verbal and nonverbal behaviors of ECAs [Vilhjálmsson et al. 2007]. Finally the task of the third element of the SAIBA framework, **Behavior Realizer**, is to realize the behaviors scheduled by the Behavior Planner. It receives input in the BML format and it generates the animation. A feedback system is needed in order to inform the modules of the SAIBA about the current state of the generated animation. These information is used, for example, by Intent Planner to re-plan the agent's intentions when an interruption occurs.

There exist several implementations of SAIBA standard. SmartBody [Thiébaux et al. 2008] is an example of the Behavior Realizer. It takes as input BML code (including speech timing data and the world status updates); it composes multiple behaviors and generates the character animation synchronized with audio. For this purpose it uses extended version of BML allowing one to define interruptions and predefined animations. SmartBody is based on the notion of animation controllers. The controllers are organized in a hierarchical structure. Ordinary controllers manage the separate channels e.g. pose or gaze. Then the meta-controllers manipulate the behaviors of subordinate controllers allowing the synchronization of the different modalities to generate consistent output from the BML code.
SmartBody can be used with the Nonverbal Behavior Generator [Lee and Marsella 2006] that corresponds to the Behavior Planner in the SAIBA framework. It is a rule-based module that generates BML annotations for nonverbal behaviors from the communicative intent and speech text. SmartBody can be used with different characters, skeletons and different rendering engines.
Heloir and Kipp [Heloir and Kipp 2009] extend the SAIBA architecture by a new intermediate layer called animation layer. Their EMBR agent is a real-time system that offers a high degree of animation control through the EMBRScript language. This language permits control over skeletal animations, morph target animations, shader effects (e.g. blushing) and other autonomous behaviors. Any animation in EMBRScript is defined as a set of key poses. Each key pose describes the state of the character at a specific point in time. Thus the animation layer gives access to animation parameters related to the motion generation procedures. It also gives an ECAs developer the possibility to control better the process of the animation generation without constraining him to enter into the implementation details.
Elckerlyc [van Welbergen et al. 2010] is a modular and extensible Behavior Realizer following the SAIBA framework. It takes as input a specification of verbal and nonverbal behaviors encoded with extended BML and can eventually return feedback concerning the execution of a particular behavior. Elckerlyc is able to re-schedule behaviors that are already queued with behaviors coming from a new BML block in real-time while maintaining the synchronization of multimodal behaviors. It receives and processes a sequence of BML blocks continuously allowing the agent to respond to the unpredictability of the environment or of the conversational partner. Elckerlyc is also able to combine different approaches to animation generation to make agent motion more human-like. It uses both: procedural animation and physical simulation to calculate temporal and spatial information of motion. While physical simulation controller provides physical realism of motion, procedural animation allows for the precise realization of the specific gestures.
BMLRealizer [Árnason and Porsteinsson 2008] created in the CADIA lab is another implementation of the Behavior Realizer layer of the SAIBA framework. It is an open source animation toolkit for visualizing virtual characters in 3D environment that is partially based on the SmartBody framework. As input it also uses BML; the output is generated with the use of the Panda3D rendering engine.
RealActor [Cerekovic et al. 2009] is another BML Realizer developed recently. It is able to generate the animation containing verbal content that is complemented by rich set of nonverbal behaviors. It uses the algorithm based on neural networks to estimate words duration. Consequently it can generate the correct lips movement without explicit information about the phonemes (or visemes). RealActor was integrated in various open-source 3D engines (e.g. Ogre, HORDE3D).

Our architecture is SAIBA compliant. Compared to existing platforms, our solution allows for creating cross-media and multi-embodiment agents. On the one hand, we propose an innovative finely grained hierarchical organization for SAIBA Behavior Realizer; on the other hand, we introduce different levels of customization of our agent. Different instantiations of agent can share Behavior Planning and Realization; only the animation computation and rendering display may need to be tailored. That is, our system is able to display the same communicative intention with different media (AR, VR, web), representations (2D, 3D) and/or embodiments (robots, virtual and web Flash-based agents). To our knowledge it is a first agent architecture that can be used with such a diversity of different outputs technologies.

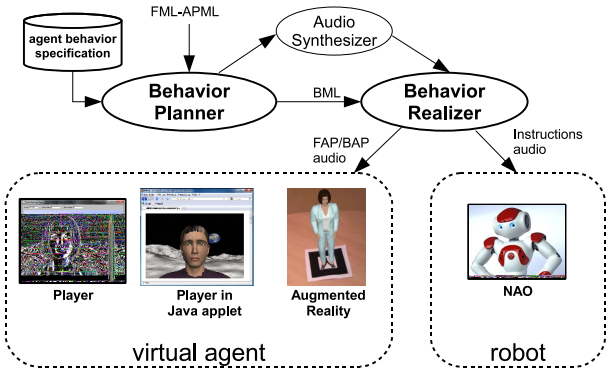

Figure 2: Our system architecture.

3 Implementation of SAIBA architecture

Following the SAIBA approach, all modules in our platform are independent of the agent's embodiment and the animation player technology. Thus given an input FML message the corresponding behavior animation is computed with the same agent platform whatever the agent's embodiment and the device on which it is displayed. In this paper we describe our implementation of the last two modules of the SAIBA framework, the Behavior Planner and the Behavior Realizer. Figure 2 shows the overall architecture of our

system. The Behavior Planner (see Section 5 for details) receives as input the communicative intentions the agent wants to transmit. For each intention it chooses the appropriate behaviors to be displayed according to the agent's behavior specification (lexicon and repositories; see Section 4). Since the FML language [Heylen et al. 2008] proposed in SAIBA is still being defined, to encode the agent intentions we use the FML-APML language, a temporary solution, proposed by Mancini and Pelachaud [Mancini and Pelachaud 2008b], based on the APML language [Carolis et al. 2004].

The multimodal signals selected by the Behavior Planner are sent to the Behavior Realizer in BML format. From the same BML input the Behavior Realizer is able to generate the animation for different embodiments (such as robots and virtual agents), for different 2D or 3D presentations and for the same embodiment displayed on different media (virtual reality, augmented reality, web-based application). For this purpose, similarly to the solution presented by Heloir and Kipp [Heloir and Kipp 2009] we introduce more fine-grained nonverbal behavior processing. In particular the Behavior Realizer is divided into two sub-layers. The first one, Keyframe Generator (KG), contains keyframe generation and scheduling; this module is common for nearly all agent's realizations. The second layer, Animation Generator (AG), is responsible for the generation of the animation from the keyframes. This layer is embodiment dependent and each embodiment (e.g. robot, virtual agent) needs specific implementation. We explain our algorithm in more details in Section 6. Finally the behavior realization process is separated from the visualization process. Specific players need to be developed allowing using the same Animation Generator (AG) layer with different media such as virtual reality or web.

All the modules communicate using the message exchange systems (either Psyclone[1] or ActiveMQ[2]) and a set of whiteboards. The following subsections present the two languages our system uses in more details.

3.1 FML-APML language

The communicative intentions of the agent correspond to what the agent aims to communicate to the user: e.g., its emotional states, beliefs and goals. FML-APML [Mancini and Pelachaud 2008b] is based on the Affective Presentation Markup Language (APML) [Carolis et al. 2004], but has the features of the future FML [Heylen et al. 2008]. It is an XML-based markup language for representing the agent's communicative intention and the text to be uttered by the agent. FML-APML uses a similar syntax as BML one. It has a flat structure and allows defining explicit duration for each communicative intention. An example of FML-APML message is presented in Figure 8.

3.2 BML language

BML language is not yet a standard in the sense of W3C or ISO, however researchers agreed on a "common" BML syntax specification to allow one to exchange BML messages and engines between different systems, as described in [Vilhjálmsson et al. 2007]. The BML language allows us to specify the nonverbal signals that can be expressed through the agent communication modalities. Each BML top-level tag corresponds to a behavior the agent is to produce on a given modality: head, torso, face, gaze, body, legs, gesture, speech, lips. Each BML tag contains also temporal information that corresponds to the timing of appearance and ending of the signals. The BML language we have implemented in our agent contains some extensions which allow us to define labels to use a larger set of signals which can be produced by the agent and to specify the expressivity of each signal [Hartmann et al. 2002]. An example of BML message is presented in Figure 9.

[1] http://www.mindmakers.org/projects/Psyclone

[2] http://activemq.apache.org/

4 Agent behavior specification

In our system agent's behavior can be personalized. Consequently, the same architecture can be used to create agents characterized by different styles of behavior or having different communicative capabilities. For such a purpose we specify an agent by a pair: a *lexicon* of communicative behaviors and its *repositories* of nonverbal behaviors. We introduce these two concepts in the following subsection.

4.1 Lexicon and repositories

A **lexicon** is made of pairs where one entry is a communicative intention and the other one is the set of behaviors that conveys this given intention or emotional state. Each one of these associations represents an *entry* of the lexicon, called *behavior set*. The definition of a behavior set BS is a quadruple:

$$\text{BS} = (\text{Name},\text{Signals},\text{Core},\text{Rules});$$

where:

- $Name$ is the identification of the corresponding communicative intention.
- $Signals$ is the set of nonverbal signals produced on single modalities that can be used to convey the intention specified by the parameter $Name$.
- $Core$ specifies which signals are mandatory to communicate the given intention;
- $Rules$ is a set of rules that precise the relations between signals, e.g. the simultaneous presence of two (or more) signals.

We use an XML-based representation language [Mancini and Pelachaud 2008b; Niewiadomski et al. 2009] to define the lexicon. It contains a tag *behaviorset* for each communicative intention the agent must be able to transmit. An example of behavior set presented in Figure 3 corresponds to the communicate intention "greeting". Just one constraint is defined in the core: the agent must show a *greeting* iconic gesture.

```
<lexicon>
  <behaviorset name="performative-greeting">
    <signals>
      <signal id="1" name="iconic=greeting" modality="gesture"/>
      <signal id="2" name="nod" modality="head"/>
      <signal id="3" name="smile" modality="face"/>
    <signals>
    <constraints>
      <core>
        <item id="1">
      <core>
  </behaviorset>
</lexicon>
```

Figure 3: Example of a behavior set for a virtual agent lexicon.

All body and facial behaviors, which can be associated to a communicative intention in the lexicon, are defined in external *repositories*. Each different virtual or physical agent in our system can be characterized by its own repository that contains a description of specific behaviors. For example, we can define an agent that shows asymmetrical facial expressions or one that does not have gestures with

inward wrist orientation. The latter is particularly important when designing gestures for the robot because of its physical limits.

We propose a new XML notation to describe symbolically the nonverbal behaviors in repositories. For instance, the symbolic specification of gestures in our system is based on a predefined finite set of key-positions in the movement space [Hartmann et al. 2002]. The arm position is always defined by three tags $< vertical\_location >$ that corresponds to axis *Y*, $< horizontal\_location >$ that correspond to axis *X*, and $< location\_distance >$ corresponding to axis *Z*. Following the gestural space of McNeill [McNeill 1992] (see Figure 4), we have 5 horizontal values: XEP, XP, XC, XCC, XOppC, 7 vertical values: YUpperEP, YUpperP, YUpperC, YCC, YLowerC, YLowerP, YLowerEP and 3 distance values: ZNear, ZMiddle, ZFar. By combining these values, we have 105 possible hand positions. An example of such description for *greeting* gesture is presented in Figure 6.

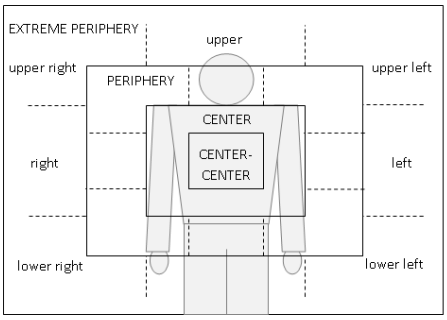

Figure 4: Symbolic gestural space.
[McNeill 1992]

4.2 Behavior distinctiveness

Through the lexicon and the repository we can specify distinctive behavior for different agents. For example, within the SEMAINE[3] project, to generate distinctive behaviors for each agent, we defined a lexicon and a repository for each of them. They have been determined partly through perceptive tests [Heylen 2007; Bevacqua et al. 2007] and partly by analyzing videos of human interactions from the SEMAINE database [McKeown et al. 2010].

4.3 Different embodiments

Different embodiments need different lexicons and repositories that correspond to their communicative capabilities. In one of our applications we use the humanoid robot NAO [Gouaillier et al. 2009] that has very different body capabilities with respect to a virtual agent. For example, our virtual agent can control the thumb independently from the other fingers of the hand while NAO cannot. Hence quantity of hand configurations of the virtual agent is much higher than NAO ones. Thus, a tailored lexicon and a tailored repository have to be defined for NAO to convey a given communicative intention. To ensure that both the robot and the virtual agent convey similar information, their lexicons have entries for the same list of communicative intentions. These entries can be, however, instantiated with different sets of nonverbal signals. The elaboration of the robot lexicon encompasses the notion of "gesture family with variants" proposed by Calbris [Calbris 1984]. Gestures from the same family convey similar meanings but may differ in their shape. For example, the robot NAO has no facial expressions and only two hand configurations (i.e. open and close). To do a deictic gesture, this robot must use an arm outstretched with (or without) turning its head while the virtual agent can use its gaze or pointing index finger and head gestures for the same intention.

```
<lexicon>
  <behaviorset name="performative-greeting">
    <signals>
      <signal id="1" name="iconic=greeting" modality="gesture"/>
      <signal id="2" name="up" modality="head"/>
    <signals>
    <constraints>
      <core>
        <item id="1">
      <core>
  </behaviorset>
</lexicon>
```

Figure 5: Example of a behavior set for a robot lexicon.

Figures 3 and 5 show behavior sets of the same communicative intention *greeting* for the virtual agent and the robot, respectively. To greet somebody, the virtual agent can choose a combination of the signals: *smile*, *head nod* and the iconic gesture *greeting*, whereas the robot can raise its head and perform the similar iconic gesture. For both embodiments the iconic gesture is mandatory.

```
<gesture id="greeting" type="iconic">
  <description level="1" type="agentbml">
    <phase type="STROKE-START" twohand="ASSYMMETRIC" time="START+0.4">
      <hand side="RIGHT">
        <vertical_location>YUpperPeriphery</vertical_location>
        <horizontal_location>XCenter</horizontal_location>
        <location_distance>ZNear</location_distance>
        <hand_shape>FORM_OPENAPART_THUMB_AWAY</handshape>
        <wrist_orientation>UP</wrist_orientation>
        <palm_orientation>AWAY</palm_orientation>
      </hand>
    </phase>
    <phase type="STROKE-END" twohand="ASSYMMETRIC" time="STROKE-START+1.4">
      <hand side="RIGHT">
        <vertical_location>YUpperPeriphery</vertical_location>
        <horizontal_location>XPeriphery</horizontal_location>
        <location_distance>ZNear</location_distance>
        <hand_shape>FORM_OPEN_THUMB</handshape>
        <wrist_orientation>UP</wrist_orientation>
        <palm_orientation>AWAY</palm_orientation>
      </hand>
    </phase>
  </description>
</gesture>
```

Figure 6: Description of the iconic gesture *greeting* in the virtual agent gestural repository.

Gestures description is specified in the agent and robot repositories. In our example, both the agent and the robot may perform the iconic gesture *greeting* to greet somebody. However the description of this gesture differs with the embodiments. Figures 6 and 7 show the *greeting* gesture description in the agent and the robot gestural repository, respectively. In this example, both of them raise and wave their right hand. However, while the virtual agent can open the right hand with stretched thumb, the robot has only one simple configuration of the opened hand. Moreover, the robot cannot rotate its wrist. Consequently in Figure 7 there is no description of wrist orientation. Additionally, the robot cannot realize an arm movement in less than 1 second. For this reason the "stroke start" phase starts later for a robot gesture than for a virtual agent one (see Figures 6 and 7).

5 Behavior Planner

The Behavior Planner takes as input both the agent's communicative intentions specified by the FML-APML language and the

[3] www.semaine-project.eu

```
<gesture id="greeting" type="iconic">
  <description level="1" type="naobml">
    <phase type="STROKE-START" twohand="ASSYMMETRIC" time="START+1.1">
      <hand side="RIGHT">
        <vertical_location>YUpperPeriphery</vertical_location>
        <horizontal_location>XPeriphery</horizontal_location>
        <location_distance>ZNear</location_distance>
        <hand_shape>OPEN</handshape>
        <palm_orientation>AWAY</palm_orientation>
      </hand>
    </phase>
    <phase type="STROKE-END" twohand="ASSYMMETRIC" time="STROKE-START+1.4">
      <hand side="RIGHT">
        <vertical_location>YUpperPeriphery</vertical_location>
        <horizontal_location>XExtremePeriphery</horizontal_location>
        <location_distance>ZNear</location_distance>
        <hand_shape>OPEN</handshape>
        <palm_orientation>AWAY</palm_orientation>
      </hand>
    </phase>
  </description>
</gesture>
```

Figure 7: Description of the iconic gesture *greeting* in the robot gestural repository.

agent's characteristics described in Section 4 (see Figure 2). The FML-APML message can be given as input to this module either manually by the user or automatically by a dialog manager module. An example of input written in FML-APML is provided in Figure 8. It specifies the agent intention to greet somebody while saying "Hi there!".

```
1. <fml-apml>
2.     <bml>
3.       <speech id="s1">
4.           <text> <sync  id="tm1"/> Hi there! <sync id="tm2"/> </text>
5.           <pitchaccent id="xpa1" type="Hstar" start="s1:tm1" end ="s1:tm2"/>
6.           <boundary id="b1" type="HH" start="s1:tm1" end="1.0"/>
7.       </speech>
8.     </bml>
9.     <performative id="p1" type="greeting" start="s1:tm1" end="s1:tm2">
10. </fml-apml>
```

Figure 8: Example of FML-APML message received by the Behavior Planner as input.

The main task of Behavior Planner module is to select automatically, for each communicative intention, the adequate set of behaviors to display. The output of the Behavior Planner is described in BML language. It contains the sequence of behaviors with their timing information to be displayed by the agent. In the following we present our Behavior Planner in more details.

First, the text to be said by the agent is sent to the TTS that computes the corresponding list of phonemes and their duration. Based on this temporal information the Behavior Planner computes all timing information of the communicative intentions (their beginning and end). Then, the planning process applies the *multimodal signal selection* algorithm proposed by Mancini and Pelachaud [Mancini and Pelachaud 2008a]. Such an algorithm is parameterized by the agent's behavior specification (see Section 4).

```
<bml>
  <face id="smile" start="2" end="5">
    <description level="1" type="agentbml"/>
  </face>
  <gesture id="greeting" start="2" end="5">
    <description level="1" type="agentbml"/>
  </gesture>
</bml>
```

Figure 9: BML generated by the Behavior Planner for a virtual agent.

By only defining application-specific lexicon we can use the same Behavior Planner to generate the behavioral signals for different embodiments. Let us consider two examples of the lexicons and the gestural repositories for a virtual agent (Figures 3 and 6) and NAO robot (Figures 5 and 7). From the same FML-APML input shown in Figure 8 our Behavior Planner module generates the BML shown in Figure 9 for the virtual agent. Consequently our virtual agent will greet its interlocutor with a smile and *greeting* gesture. While for the robot NAO, with the same FML-APML message (see Figure 8) the Behavior Planner will generate the BML message shown in Figure 10. Consequently, the robot will use only a gesture.

```
<bml>
  <gesture id="greeting" start="2" end="5">
    <description level="1" type="naobml"/>
  </gesture>
</bml>
```

Figure 10: BML generated by the Behavior Planner for the robot.

6 Behavior Realizer

Once the Behavior Planner has chosen a set of signals, the description of these signals and their timing information is sent through a whiteboard to the Behavior Realizer module as BML messages. The main task of our Behavior Realizer is to generate animation of the agent (virtual or physical) from the received BML. This process is divided into two main stages: the first one, called Keyframes Generator (KG) can be used commonly for different embodiments, while the second one, Animation Generator (AG), is embodiment specific. Figure 11 presents the structure of our Behavior Realizer. In the following subsections we present these modules in details.

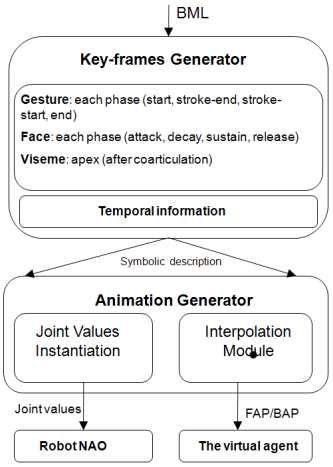

Figure 11: Two layers of Behavior Realizer.

6.1 Keyframes Generator

At this stage the behavior signals described in the BML message are initialized. They may be facial expressions, eye, head and torso movements, or hand gestures. All the signals that may occur in BML messages have their symbolical descriptions in corresponding repositories (see Section 4.1). The Keyframes Generator synchronizes the behaviors across modalities with speech. In our system, the synchronization between behavioral signals is realized by adapting the timing of the behaviors to the speech timing. It means that temporal information of nonverbal behaviors in BML tags are relative to speech; they are specified through time markers.

In the case of gestures, the temporal information of each behavior refers to gesture phases as defined by Adam Kendon [Kendon 2004]. As shown in Figure 12, they are encoded by seven synch points: start, ready, stroke-start, stroke, stroke-end, relax and end [Vilhjálmsson et al. 2007]. They divide a behavior into several phases of realization, in which the most meaningful part occurs between stroke-start and stroke-end (i.e. stroke phase). The preparation phase goes from start to ready. It serves to take the articulatory joints (e.g. hand and wrist) to the position where the stroke occurs. According to observations by McNeill [McNeill 1992], gesture always coincides or lightly precedes speech. In our system synchronization between gesture and speech is ensured by forcing the starting time of the stroke phase to coincide with the stressed syllables. The system has to pre-estimate the time required for realizing the preparation phase, t_{pre}, in order to make sure that the stroke happens on the stressed syllables. This pre-estimation is done by calculating the distance between the current hand-arm position and the next desired positions and by computing the time it takes to perform the trajectory (t_{traj}). In case the allocated time is not enough to do the preparation phase (that is $t_{pre} < t_{traj}$), the whole gesture has to be cancelled, leaving free time to prepare for the next gesture. We do similar computation for facial expressions. In this case the dynamic pattern of a behavior is defined with four phases attack, decay, sustain and release [Kalra and Magnenat-Thanmann 1994].

The result of the Keyframes Generator is a set of keyframes. Each keyframe contains the symbolic description and timing of each facial phase (attack, decay, sustain, release), each viseme apex after co-articulation algorithm [Bevacqua and Pelachaud 2004] and each gesture phase (start, stroke_start, stroke_end, end). In particular gestures are described using the XML notation introduced in Section 4.1. The symbolic representation of keyframes allows us to use the same synchronization algorithm for different realizations of planned behaviors. It also assures correct synchronization of gestures with speech independently of the agent embodiment or animation parameters.

6.2 Animation Generator

To compute the animation given the set of keyframes we need to use an Animation Generator specific to each embodiment. While all the previous computations use the common agent framework, this stage is embodiment dependent. Given an embodiment the adequate Animation Generator receives as input the keyframes and calculates the values of the animation parameters. So far we have developed two different Animation Generators. For the virtual agent we use Interpolation Module (IM), while for the robot NAO we use Joint Values Instantiation Module (JVIM).

6.2.1 Interpolation Module

For the virtual agent each keyframe is converted into MPEG-4 body and facial animation parameters (BAP and FAP) [Doenges et al. 1997]. FAP corresponds to facial point displacement while BAP to joint value. Each symbolic description of facial action and gesture phases is translated into specific BAP/FAP values [Pasquariello and Pelachaud 2001; Hartmann et al. 2002]. The animation between keyframes is obtained by interpolating these values. For this purpose we use TCB-Splines algorithm [Hartmann et al. 2002].

6.2.2 Joint Values Instantiation Module

The animation of the robot is driven by the same representation language BML that is used for the virtual agent; but as explained in Section 5, in the Behavior Planner, the BML tags are instantiated using each embodiment's own lexicon. Each keyframe contains the temporal information and the description of the behaviors to be realized. The Joint Values Instantiation Module (see Figure 2) translates these keyframes into joint values of the robot. The symbolic notation of gestures (see Section 4.1) uses predefined finite set of key-positions in the movement space. Each symbolic wrist position corresponds to a set of fixed joint values of the robot. In our example (see Figure 7) the position of the right hand (YUpperPeriphery, XPeriphery and ZNear) will be translated to concrete values of 6 NAO joints: RElbowRoll, RElbowYaw, RHand, RShoulderPitch, RShoulderRoll, RWristYaw. Due to physical limitations of NAO robots some combinations of parameters described at symbolic level cannot be realized. For example NAO is not able to place its hand in front of its body. In such cases the mapping between the symbolical description and NAO joints is realized by choosing the most similar available movement. Finally, based on the absolute time of the keyframes, the animation is obtained by interpolating between joint values with robot built-in proprietary procedures [Gouaillier et al. 2009].

7 Cross media outputs

In our architecture the BML generated by the Behavior Planner (see Section 5) may be realized in many different ways. In this section we explain different realizations that have been built so far. First of all different embodiments can be used, such as humanoid robot, AIBO robot[Al Moubayed et al. 2009] or different virtual agents (2D Flash based, 3D Ogre based). Our virtual agents may be visualized in virtual as well as augmented reality. It is important to notice that all these implementations use different instantiations of the same Behavior Planner that is parameterized by the agent behavior specification. In Figure 13 an example of interaction between two different embodiments using two instantiations of the same Behavior Planner is shown.

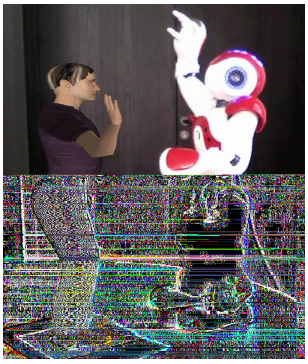

Figure 13: Virtual agent and NAO: two different agent embodiments.

7.1 VR-Player

The animation generated by the Behavior Realizer is displayed in a graphic window by a MPEG-4 compliant Player. The VR-Player uses the per-vertex animation. Facial and body configurations are described through respectively FAP and BAP frames. To create facial animation the mesh deformations for each FAP were defined for each character with 3D graphics software. Then the final animation is the generated with Ogre graphics engine that can use either DirectX9 technology or OpenGL libraries.

Within the SEMAINE project four different virtual characters can be displayed in the graphic window; the user can decide which of them she wants to interact with. Characters are loaded dynamically,

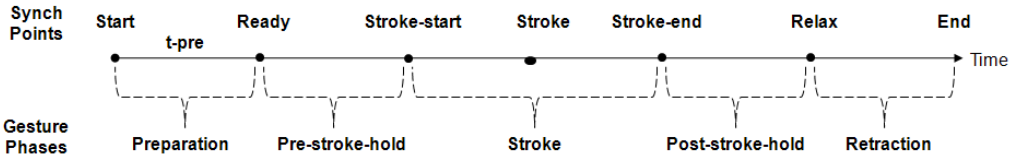

Figure 12: Standard synch points (from SAIBA website).

that allows the system to pass easily from one character to another when needed. Since each mesh is quite complex, the four characters are loaded in the memory when the system is launched. They can be visible or not in the virtual world as needed. The VR-Player sends directly the animation to the character that is actually displayed.

7.2 Web Player

Lately we implemented another version of the Player in order to be displayed in a Web browser. The Web Player is embedded in an Java applet. It is based on OGRE4J[4], that allowed us to use the Ogre libraries in the Java application. This solution has been chosen mainly to ensure interoperability within different operating systems and hardware including the next generation mobile devices. Rather than creating a new player from the scratch the Web Player reuses some parts of the original C++ code and libraries through Java Native Interface. Similarly to classic VR-Player, the Web Player receives from the Behavior Realizer a message in MPEG-4 format through the message exchange system (see Section 3) and it displays it in Java Swing Window. Consequently any client machine that runs a Java Virtual Machine can connect to the server that hosts the applet, download it and run it to display the Player. Only the Web Player has to be downloaded to the local machine while the other modules are placed remotely on the server and exchange the messages using the whiteboards. Server-side parallel processing for many clients is not available yet.

The Web Player runs in the local machine using the resources and the libraries stocked on the server. For security reasons the user's authorization is always needed to download these libraries and run the Web Player.

7.3 Flash Behavior Realizer and Player

Within the national project *MyPresentingAvatar* another web based application was created in collaboration with the companies Cantoche[5] and Lingway[6]. The aim of the project is to have the virtual agent presenting documents, such as one's CV or news. The project uses our architecture to compute the behavior the agent should display and Cantoche LivingActor technology to visualize the animation. For this purpose, for each possible BML entry, a flash animation has been previously created and stored in a lexicon. So, in this application, our Behavior Planner generates BML code which is then displayed by a 2D character created in flash. Synchronization between behaviors and speech is ensured by scaling each flash animation to fit within their BML time specification.

7.4 AR-Player

We also incorporated our virtual agent into an Augmented Reality (AR) environment to allow it to inhabit the real world with the user. Augmented Reality is a technology that allows virtual content to be superimposed on the real world such that the virtual imagery appears seamlessly blended into the real space of the user, creating a strong sense of presence.

To incorporate the agent into an AR environment, we use the ARToolKit[ARToolKit 2001] computer vision library. Using a webcam and printed markers, ARToolKit is able to compute the 3D transformations between the camera and each marker by analyzing the incoming video frames and referencing known marker sizes and camera parameters. With correct calibration, millimeter accuracy is possible, and on a standard workstation computation can easily keep pace with a typical camera capture rate of 30 frames per second.

The existing Greta software uses the Ogre3D rendering engine. Like most modern graphics libraries, Ogre uses a hierarchy of transformations to move and position objects within a virtual world. This is how the Greta character can be placed at certain locations within the environment. We integrated ARToolKit with Ogre, and used the real time tracking information to update the transformation that determines Greta's position and orientation. In addition, we added the live video feed from the user's webcam in the background behind Greta, giving the overall impression that Greta is standing on a marker within the user's real surroundings. Once each transformation matrix between camera and marker is known, it can be applied within the rendering module to position our virtual character so to appear collocated with the marker. Local transformations can then be applied to move, rotate and scale the character relative to the marker. If the marker is not detected, the character is hidden from view until the marker is successfully tracked again. Behind different visualization AR-Player-based virtual agent maintains all communicative capabilities of the virtual agent as described in Section 6.2. It can receive and realize any BML message containing verbal and nonverbal behaviors. It is also able to generate some autonomous behaviors to maintain the interaction with the user.

To create a more immersive AR experience, the webcam is often attached to a head mounted display to simulate the wearer's real world view. The video image is captured, processed by the computer, augmented with 3D objects, and then presented back to the wearer's eyes.

Figure 14 illustrates the process ARToolKit follows when incorporating our virtual agent into an AR:

- (a) Receive a video stream image from the camera.
- (b) The image is binarized and the square marker is detected.
- (c) The square marker is extracted and its position and orientation are computed.
- (d) The pattern within the square marker is extracted.
- (e) The pattern within the marker is processed using template matching and identified within the set of loaded patterns.

[4] http://ogre4j.sourceforge.net/
[5] http://www.cantoche.com
[6] http://www.lingway.com

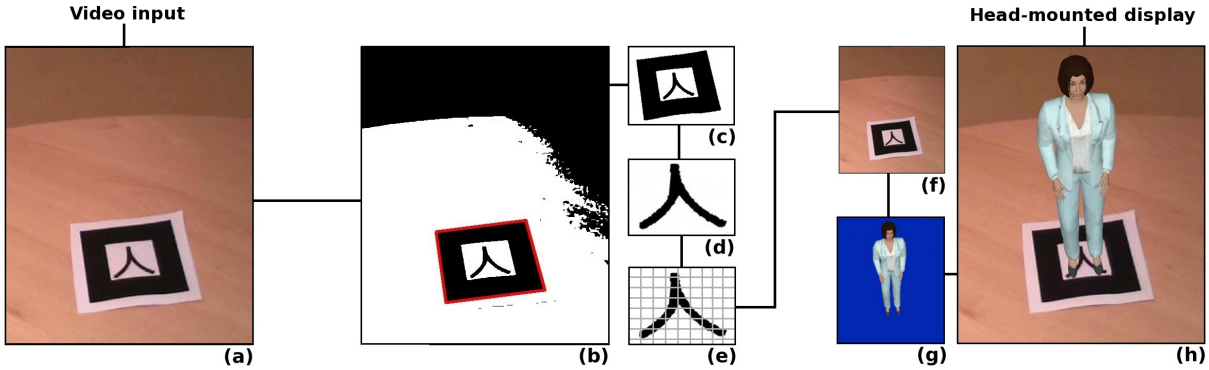

Figure 14: The process of incorporating virtual agent in an AR interface. (a) Video capture, (b) tracking the marker, (c) compute position and orientation of the marker, (d) identify the marker, (e) pattern matching, (f) texturing the video image, (g) rendering the 3D content, (c) the final output image.

- (f) The input video image becomes the background for the final output rendering.
- (g) The 3D content to be displayed is prepared.
- (h) The 3D content (g) is rendered over the video background (f) to produce the final image of the 3D content incorporated into the real scene. The final image is then displayed and observed by the user.

Our AR-Player permits to use more than one character in the scene. Each of them can become an independent agent with its own behavior specification, and its own instantiation of Behavior Planner. These different agents can be associated with different markers in the same scene.

7.5 Robot

The humanoid robot NAO developed by Aldebaran [Gouaillier et al. 2009] has 23 degrees of freedom that make it able to move expressively. From results of the Joint Values Instantiation Module, the animation is obtained by using the interpolation mechanism available in the NAO.

8 Conclusions

In this paper we presented a general purpose use and modular architecture of an embodied conversational agent. Our architecture follows the SAIBA framework that defines three of levels of abstraction, from the computation of the agent's communicative intention, to behavior planning and realization. Each module of the architecture communicates with each other using XML languages. We designed an architecture that allows fewer possible customizations when applied to different agent technologies and on a variety of media. Customization can happen at different levels. By creating specific lexicons and repositories, the same architecture can be used to drive the behaviors of various agent types. Behaviors are described independently of the agent player technology. This independency allows us to control the behavior of a virtual agent and a humanoid robot using the same language, namely BML. By changing the player technology, agent animations can be displayed across different media, PC screen, web browser or even augmented reality. Figure 15 summarizes the flexibility of our architecture for customization process.

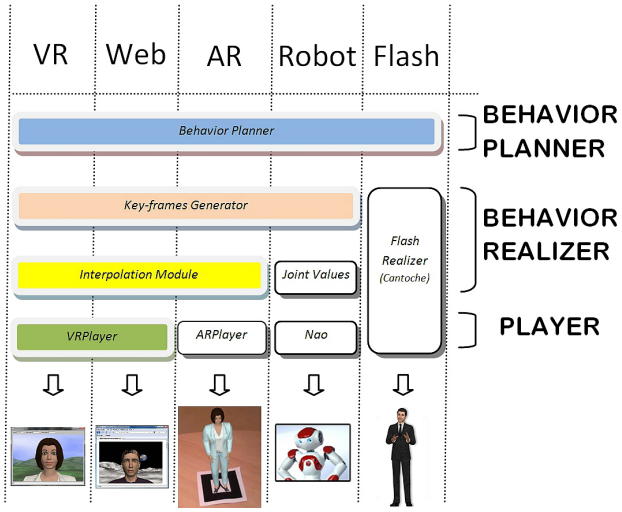

Figure 15: Different levels of customization of our agent.

9 Acknowledgment

This work has been funded by the STREP SEMAINE project IST-211486 (http://www.semaine-project.eu), and by French National Projects MyPresentingAvatar, IMMEMO and GVLEX.

References

AL MOUBAYED, S., BAKLOUTI, M., CHETOUANI, M., DUTOIT, T., MAHDHAOUI, A., MARTIN, J.-C., ONDAS, S., PELACHAUD, C., URBAIN, J., AND YILMAZ, M. 2009. Generating robot/agent backchannels during a storytelling experiment. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, IEEE Press, Piscataway, NJ, USA, ICRA'09, 2477–2482.

ÁRNASON, B., AND PORSTEINSSON, A., 2008. The CADIA BML realizer. http://cadia.ru.is/projects/bmlr/.

ARTOOLKIT, 2001. http://www.hitl.washington.edu/artoolkit/.

BEVACQUA, E., AND PELACHAUD, C. 2004. Expressive audio-visual speech. *Journal Computer Animation and Virtual Worlds (CAVW) 15*, 3-4 (July), 297–304.

BEVACQUA, E., HEYLEN, D., TELLIER, M., AND PELACHAUD, C. 2007. Facial feedback signals for ECAs. In *AISB'07 Annual convention, workshop "Mindful Environments"*, 147–153.

CALBRIS, G. 1984. *Contribution à une analyse sémiologique de la mimique faciale et gestuelle française dans ses rapports avec la communication verbale*. ANRT.

CAROLIS, B. D., PELACHAUD, C., POGGI, I., AND STEEDMAN, M. 2004. APML, a mark-up language for believable behavior generation. In *Lifelike Characters. Tools, Affective Functions and Applications*, H. Prendinger and M. Ishizuka, Eds. Springer.

CEREKOVIC, A., PEJSA, T., AND PANDZIC, I. 2009. RealActor: Character animation and multimodal behaviour realization system. In *Proceedings of 9th International Conference on Intelligent Virtual Agents, IVA 2009*, 486–487.

DOENGES, P., CAPIN, T., LAVAGETTO, F., OSTERMANN, J., PANDZIC, I., AND PETAJAN, E. 1997. MPEG-4: Audio/video and synthetic graphics/audio for real-time, interactive media delivery, signal processing. *Image Communications Journal 9*, 4, 433–463.

EKMAN, P., FRIESEN, W. V., AND HAGER, J. C. 2002. *Facial Action Coding System. The Manual*. A Human Face, Salt Lake City, USA.

GOUAILLIER, D., HUGEL, V., BLAZEVIC, P., KILNER, C., MONCEAUX, J., LAFOURCADE, P., MARNIER, B., SERRE, J., AND MAISONNIER, B. 2009. Mechatronic design of NAO humanoid. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, 769–774.

HARTMANN, B., MANCINI, M., AND PELACHAUD, C. 2002. Formational parameters and adaptive prototype instantiation for MPEG-4 compliant gesture synthesis. In *Proceedings of the Computer Animation*, 111–120.

HELOIR, A., AND KIPP, M. 2009. EMBR - a realtime animation engine for interactive embodied agents. In *Proceedings of 9th International Conference on Intelligent Virtual Agents, IVA 2009*, 393–404.

HEYLEN, D., KOPP, S., MARSELLA, S., PELACHAUD, C., AND VILHJALMSSON, H., 2008. Why conversational agents do what they do? Functional representations for generating conversational agent behavior. The First Functional Markup Language workshop.

HEYLEN, D. 2007. Multimodal Backchannel Generation for Conversational Agents. In *MOG 2007 Workshop on Multimodal Output Generation*.

KALRA, P., AND MAGNENAT-THANMANN, N. 1994. Modeling of vascular expressions in facial animation. *Computer Animation*, 50–58.

KENDON, A. 2004. *Gesture: Visible action as utterance*. Cambridge University Press.

KOPP, S., KRENN, B., MARSELLA, S., MARSHALL, A., PELACHAUD, C., PIRKER, H., THÓRISSON, K., AND VILHJÁLMSSON, H. 2006. Towards a common framework for multimodal generation: The behavior markup language. In *Intelligent Virtual Agents*, Springer, 205–217.

LEE, J., AND MARSELLA, S. 2006. Nonverbal behavior generator for embodied conversational agents. In *Proceedings of 6th International Conference on Intelligent Virtual Agents*, Springer, Marina Del Rey, CA, USA, vol. 4133 of *Lecture Notes in Computer Science*, 243–255.

MANCINI, M., AND PELACHAUD, C. 2008. Distinctiveness in multimodal behaviors. In *Proceedings of 7th Conference on Autonomous Agents and Multi-Agent Systems*.

MANCINI, M., AND PELACHAUD, C. 2008. The FML-APML language. In *Proceedings of The First Functional Markup Language Workshop*, 43–47.

MCKEOWN, G., VALSTAR, M., COWIE, R., AND PANTIC, M. 2010. The SEMAINE corpus of emotionally coloured character interactions. In *IEEE Int'l Conf. Multimedia & Expo*, 1079–1084.

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

NIEWIADOMSKI, R., HYNIEWSKA, S., AND PELACHAUD, C. 2009. Modeling emotional expressions as sequences of behaviors. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, Springer, Amsterdam, Holland, 316–322.

OSTERMANN, J. 2002. Face animation in MPEG-4. In *MPEG-4 Facial Animation - The Standard Implementation and Applications*, I. Pandzic and R. Forchheimer, Eds. Wiley, England, 17–55.

PASQUARIELLO, S., AND PELACHAUD, C. 2001. Greta: A simple facial animation engine. In *6th Online World Conference on Soft Computing in Industrial Applications*.

REEVES, B., AND NASS, C. 1996. *The media equation: How people treat computers, television and new media like real people and places*. CSLI Publications, Stanford, CA.

RICKEL, J., MARSELLA, S., GRATCH, J., HILL, R., TRAUM, D., AND SWARTOUT, B. 2002. Towards a new generation of virtual humans for interactive experiences. *IEEE Intelligent Systems*, 32–38.

THIÉBAUX, M., MARSELLA, S., MARSHALL, A., AND KALLMANN, M. 2008. SmartBody: behavior realization for embodied conversational agents. In *Proceedings of 7th Conference on Autonomous Agents and Multi-Agent Systems*, 151–158.

THÓRISSON, K. R., LIST, T., PENNOCK, C., AND DIPIRRO, J. 2005. Whiteboards: Scheduling blackboards for semantic routing of messages & streams. In *AAAI-05 Workshop on Modular Construction of Human-Like Intelligence*, 8–15.

VAN WELBERGEN, H., REIDSMA, D., RUTTKAY, Z., AND ZWIERS, J. 2010. Elckerlyc - a BML realizer for continuous, multimodal interaction with a virtual human. *Journal on Multimodal User Interfaces*, 4, 271–284.

VILHJÁLMSSON, H., CANTELMO, N., CASSELL, J., CHAFAI, N. E., KIPP, M., KOPP, S., MANCINI, M., MARSELLA, S., MARSHALL, A., PELACHAUD, C., RUTTKAY, Z., THÓRISSON, K., VAN WELBERGEN, H., AND VAN DER WERF, R. 2007. The Behavior Markup Language: Recent developments and challenges. In *Proceedings of 7th International Conference on Intelligent Virtual Agents*, Springer, Paris, France, vol. 4722 of *Lecture Notes in Computer Science*, 99–111.

VON DER PÜTTEN, A., KRÄMER, N. C., GRATCH, J., AND KANG, S. 2010. "It doesn't matter what you are!" Explaining social effects of agents and avatars. *Computers in Human Behavior, in press*.