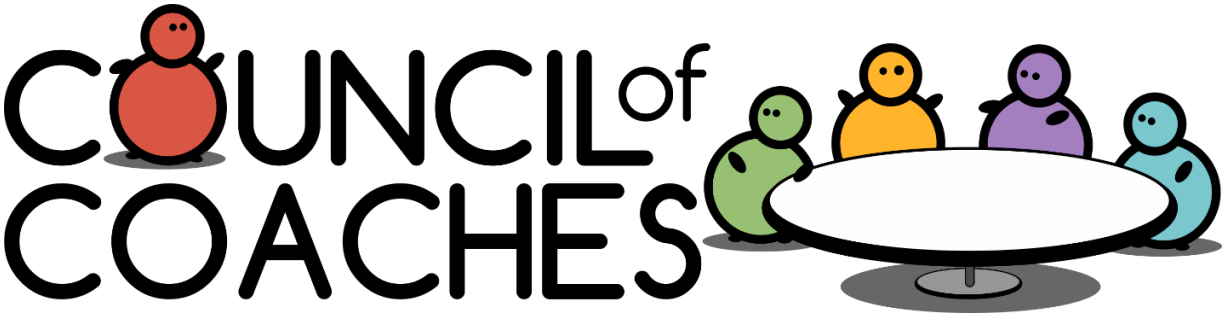


D6.3: First Prototype description and evaluations of the virtual coach platform

Dissemination level: Public

Document type: Report

Version: 1.0.1

Date: January 15, 2019 (original)

March 5, 2019 (this version)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement #769553. This result only reflects the author's view and the EU is not responsible for any use that may be made of the information it contains.

Document Details

Project Number	769553
Project title	Council of Coaches
Title of deliverable	First Prototype description and evaluations of the virtual coach platform
Due date of deliverable	November 31, 2018
Work package	WP6
Author(s)	Donatella Simonetti (SU), Reshmashree Kantharaju (SU), Randy Klaassen (CMC), Merijn Bruijnes (CMC), Harm op den Akker (RRD), Jorien van Loon (CMC)
Reviewer(s)	Kostas Konsolakis (CMC)
Approved by	Coordinator
Dissemination level	PU: Public
Document type	Report
Total number of pages	30

Partners

- University of Twente – Centre for Monitoring and Coaching (CMC)
- Roessingh Research and Development (RRD)
- Danish Board of Technology Foundation (DBT)
- Sorbonne University (SU)
- University of Dundee (UDun)
- Universitat Politècnica de València, Grupa SABIEN (UPV)
- Innovation Sprint (iSPRINT)

Abstract

In this deliverable we present the work done so far on the development of the technical prototype of the Council of Coaches system. We present the release of the first technical prototype and the work that has been done towards the second technical prototype. We present the current (October 2018) architecture of the Council of Coaches platform and discuss work on different modules of the platform. This deliverable also presents two different user evaluation studies which have been carried out, or will be carried out, to evaluate the technical demonstrator.

Corrections

v1.0.1 Correctly applied EU logo on header page.

Table of Contents

List of Figures

List of tables

Symbols, abbreviations and acronyms

ASAP	Articulated Social Agents Platform
AU	Action Unit
BAP	Body Animation Parameter
BML	Behavior Markup Language
CMC	Centre for Monitoring and Coaching
COUCH	Council of Coaches
D	Deliverable
DBT	Danish Board of Technology Foundation
DGDL	Dialogue Game Description Language
DGEP	Dialogue Game Execution Platform
EC	European Commission
ECA	Embodied Conversational Agent
FAP	Face Animation Parameter
HBAF	Holistic Behaviour Analysis Framework
iSPRINT	Innovation Sprint
M	Month
MPEG	Moving Picture Experts Group
MS	Milestone
RRD	Roessingh Research and Development
SDK	Software Development Kit
SU	Sorbonne University
TTS	Text-To-Speech
UDun	University of Dundee
UI	User Interface
UPV	Universitat Politècnica de València
UT	University of Twente
WP	Work Package

1 Introduction

In the Council of Coaches project an application is being developed that will provide tailored and personalized virtual coaching for ageing people to support them in improving their health and well-being. The focus of the coaches in Council of Coaches will be on the physical, social, mental, and cognitive domains as well as the specific cases of Diabetes Type 2, Chronic Pain, and Age-Related Impairments. Each of the six coaches that will be developed will have their own expertise, appearance, role, personality, and coaching strategies. They will interact with each other and the user to motivate and inform them, as well as discuss issues related to their health and well-being with them.

The main objective of work package 6 is to implement and evaluate the user interaction in different use case scenarios. The main user interface of the Council of Coaches application will be used in the home environment of the end user – on a screen size comparable to a laptop or tablet. The companion mobile application will facilitate interaction with the system when users are on the go.

The aim of this deliverable is to present the development of first technical prototype that was released in July 2018. This deliverable also describes the work towards the second technical prototype that is carried out after the development of the first technical prototype until the result of the third technical integration session held in October 2018 in Twente. The current architecture is presented and relevant modules are presented in more detail. This deliverable does also present user evaluation studies carried out and planned with the first technical prototype (or versions of the system that were developed towards the second technical prototype.

2 Objectives

The objective of this deliverable is to present the work that is relevant for development of the first technical prototype and the work that is carried out after the release of the first technical prototype towards the second technical prototype. This includes a description of the current platform. The first technical prototype is discussed, and the work on different modules towards the second prototype are discussed in more detail. In summary, the aims of this deliverable are:

- To provide a description of the current platform (with inclusion of work done during and until the third technical integration week)
    - Include the work up until now.
- To provide a description of improvements/progress in individual components:
    - Greta: Support of automatic gaze behaviour in group, Multi-agent support
    - ASAP: automatic gaze behaviour in group, Multi-agent supported in Unity
    - Flipper: Integration of DGEP interactions
    - The Unity scene: User interface generation and control over buttons, user input, and the integrated virtual web browser that can be controlled by Flipper
- To report on carried out and planned user evaluations. One study will investigate the influence of having a group discussion on engagement, preference, and enjoyment by the users. The other user evaluation is more focused on the use of the Council of Coaches platform. We investigate if and how easy third parties can design and implement dialogues using the Council of Coaches system.

3 Current Platform

This section presents the status of the current Council of Coaches platform. Section 3.1 discusses the first technical prototype released in July 2018. Section 3.2 presents the development done up to and including the technical integration week held in Enschede in October 2018.

3.1 First technical prototype

The first technical prototype consists of a Unity scene and the ASAP realizer. Within the Unity project a small dialogue manager is implemented that is able to steer the scripted dialogue and control the possible user input and user interface. The content of the scripted dialogue is based on the work of the scenario taskforce part of work package 7. The content from the scenarios is transformed into Behaviour Markup Language (BML) and is stored as assets in Unity. The sentences are used for the coaches' speech. The gaze behaviour of the coaches is designed as follows; coaches will look at the coach that is speaking. When they are waiting for input from the user, the coaches are looking at the user in front of the screen.

The dialogue manager is responsible for selecting the next move in the dialogue, controlling the user interface and listening to the feedback provided by the ASAP realizer. The dialogue manager is listening to the feedback of the ASAP realizer. When ASAP has finished the execution of the behaviours, the dialogue manager will display the possible reactions from the user as buttons at the top left corner of the screen (see Figure 1). The user can select their input by clicking on one of the buttons. The dialogue manager will send the corresponding BML blocks to the ASAP realizer which will execute the behaviours.

The first technical prototype can be downloaded as a zip file[1]. After unzipping the file, the prototype can be started by double clicking start_Functional_Demonstrator.bat on a Windows 10 computer with Java SDK installed.

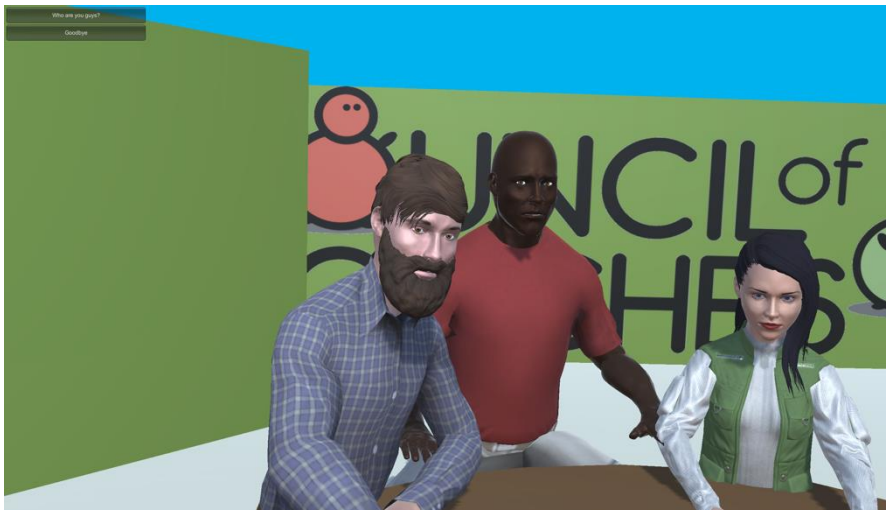


Figure 1: Screen capture from the first technical prototype.

3.2 Towards second technical prototype

After the release of the first technical demonstrator the implementation of the Council of Coaches platform has been extended. This section describes the work performed on the platform till the end of the technical integration meeting (Oct '18). A preliminary architecture is shown in Figure 2, it details the technical components (planned) and how they are connected. This deliverable deals with the

[1] http://council-of-coaches.eu/downloads/council-of-coaches-demonstrator-1.zip

components in the blue box in Figure 2, while the components from other work packages and the information that flows between them is also displayed.

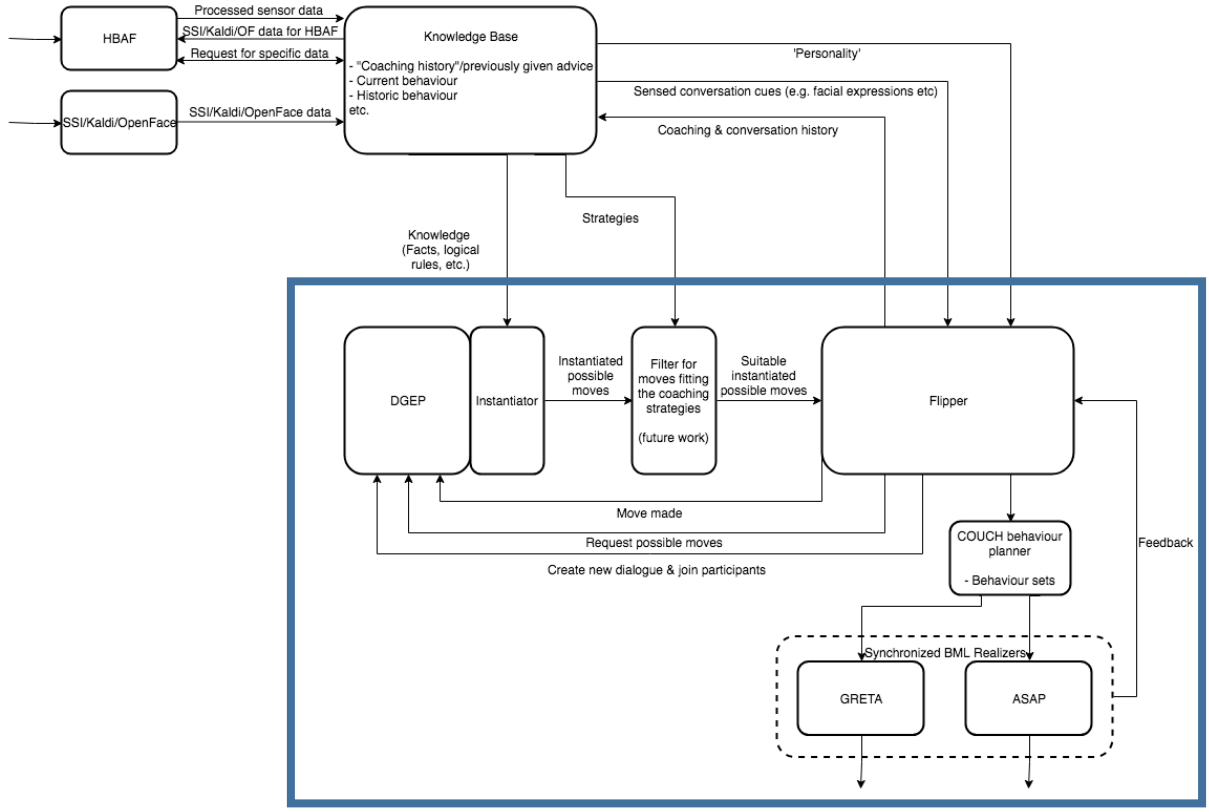


Figure 2: Architecture of the Council of Coaches platform after the 3rd technical integration week (held in Twente in October 2018).

3.2.1 ASAP

The ASAP realizer now supports multiple agents. In order to support multiple agents, changes in the BML specification have been made. To accommodate multiple agents, a realizer needs to know for which agent the behaviour is intended. For this reason, a 'characterID' attribute needs to be included in a BML-block. Additionally, to synchronize behaviours between agents, it is possible to refer to other BML-blocks and the behaviours in those blocks for the same or other agents. This only works within the same realiser (ASAP or Greta). A realiser does not have direct access to the behaviours sent to another realiser, so they cannot directly synchronise between BML-blocks. We utilise Flipper to synchronise behaviours between realisers.

3.2.2 Flipper

We describe our new dialogue engine called Flipper 2.0 (Flipper) in detail in our publication (van Waterschoot, et al., 2018)[2]. Flipper helps developers (of ECAs) to quickly and flexibly create dialogues.

Flipper 2.0 uses the feedback from a BML-realiser to synchronise behaviour between platforms. For example, agent A, controlled by ASAP, and agent G, controlled by Greta, need to synchronise their behaviour. In this example we want agent A to say "This is awesome!" and then agent G to nod in agreement. With Flipper this is relatively straight forward. Flipper sends ASAP, the realiser for agent A, the BML command to perform the speech. ASAP sends continuous feedback messages about the

[2] This publication received a best (student) paper award.

progress of the behaviour it is executing. Flipper waits for the feedback that signals the speech behaviour is complete and then sends a command to Greta to agree. Greta nods in agreement in the synchronised manner that was specified.

The way this is organised in Flipper is as follows. Flipper templates together performing a specific function, are organised in separate files. (This is not a technical requirement but good practice). Multiple template files can exist. For the example above, a small selection of template files is relevant.

Feedback listener templates subscribe Flipper to the middleware topic for each agent and handle the feedback messages on these topics. This means that when a new message arrives, the message is placed in the information state where it is available for other templates.

A rudimentary dialogue manager template file holds these templates and determines which behaviour each agent should perform at which moment. A template from the example would have, in its preconditions, the feedback message that states the behaviour it is waiting for, has been completed. The effect of this template contains behaviour for another agent to perform and places this behaviour as a variable in the information state (which functions as a cue).

An intent planner template file contains the templates for sending behaviour requests to the agents. It continuously checks some information state variable for behaviour to execute. When the example template places behaviour in this information state, the intent planner templates construct a valid behaviour request message (BML or FML) and sends it to the appropriate middleware topic.

A similar template pattern is used to create social gaze behaviour between ASAP and GRETA agents. The gaze of each agent is based on saliency of stimuli in the scene. For example, when one agent speaks it becomes more salient and the other agents will look at the speaker. For a speaker the agent or user who is being addressed becomes more salient, so the speaker will look at the addressed agent or user. Other gaze patterns are implemented, including aversion of staring at someone (a gaze target becomes less salient over time) and idle gaze patterns (random targets get a slight saliency).

3.2.3 Integration with DGEP

The Flipper module and the DGEP module communicate with each other about the next moves and steps in the dialogue. DGEP sends all possible next moves to Flipper. A Flipper template selects a move and translates the move into BML which will be executed by one of the BML realizers. When the behaviour is executed Flipper will notify DGEP about the fact that the move was executed. DGEP will send all new possible next moves to Flipper.

A Flipper template is responsible to initiate a DGEP game (the dialogue). Flipper will create a game by specifying a certain protocol. A protocol is a predefined grammar for a possible dialogue. Participants have to join the dialogue with a specific role. The grammar for the dialogue game from DGEP is called DGDL. The grammar can be written by hand or it can be generated based on a dialogue specified in the YARN editor. Flipper tracks the name and role of each participant in the conversation. One of the roles that a participant can have is "user" meaning that these are the moves that a user can make (and that the current system can understand).

The DGEP moves are currently hardcoded and contain the text that an agent should speak. A "*move instantiator*" is planned that will dynamically instantiate the move: it will create the text that an agent should speak so that it is suitable to the (coaching) strategy of the system and/or emotional state of the agent.

3.2.4 Unity scene

A Unity3D scene acts as the user interface for the Council of Coaches platform (see Figure 3). In the Unity scene, three agents controlled by ASAP and an agent controlled by Greta are rendered. Additionally, buttons for user input and a virtual web browser are present.

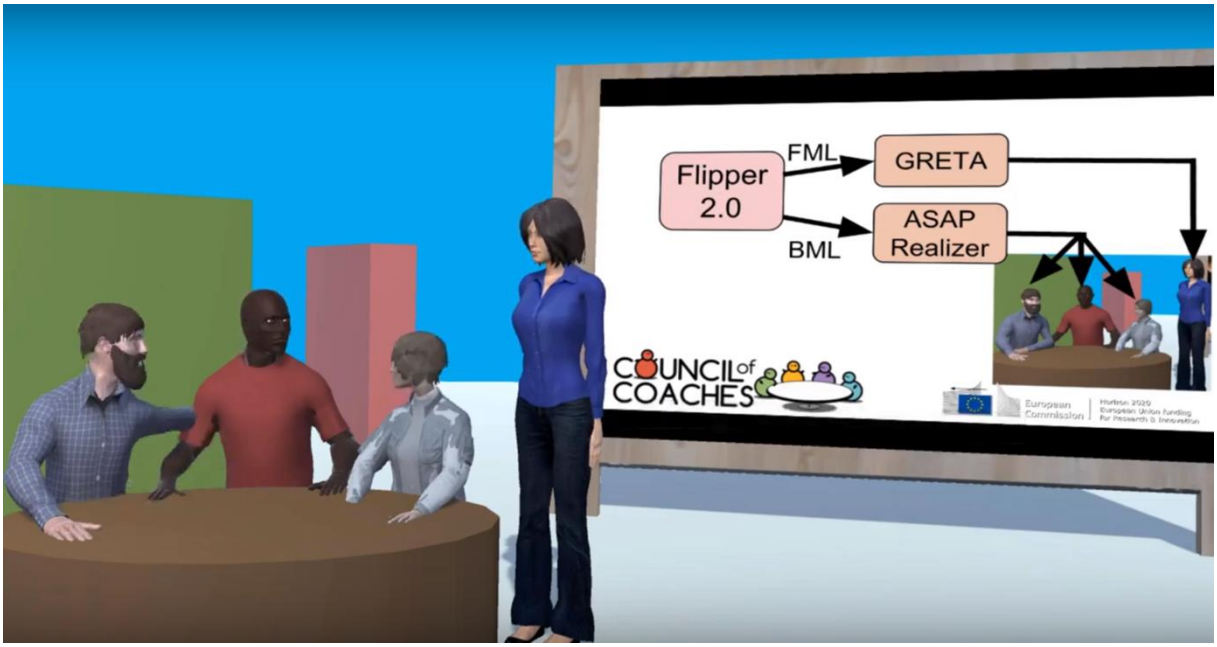


Figure 3: The current Unity scene – three ASAP agents (most left agents), one GRETA agent (right agent) and the virtual web browser).

3.2.4.1 UI generation and control

Buttons for user input are controlled by Flipper. Flipper can control the text displayed on each button and the visibility of each button. When Flipper receives moves from DGEP, all available possible dialogue moves that are intended for the participant "user" are displayed on the buttons in the Unity scene. A user can select a move by clicking on the button. Buttons are not shown when one of the agents is speaking or when there are no available moves for the user.

A virtual web browser is present in the Unity scene. The web browser can be controlled by BML and is able to show all kind of content (e.g. videos, games, websites, etc.) to support the statements of the agents. Showing content in the browser can be synchronized with the behaviours generated by the BML realizers. An example of a BML message controlling the browser where the webpage at http://demo.council-of-coaches.eu/ is loaded (at this location, the project's *Functional Demonstrator* is currently running – see D7.3):

```
<bml composition="REPLACE" xmlns="http://www.bml-initiative.org/bml/bml-1.0" xmlns:mwe="
http://hmi.ewi.utwente.nl/middlewareengine" id="bml1"  characterId="COUCH_M_1">

    <mwe:sendJsonMessage id="m1" start="0" end = "1" middlewareloaderclass=
    "nl.utwente.hmi.middleware.activemq.ActiveMQMiddlewareLoader" middlewareloaderproperties=
    "oTopic:COUCH/UI/URL">
        {"url":"http://demo.council-of-coaches.eu"}
    </mwe:sendJsonMessage>

</bml>
```

4 Greta

Greta is a real-time three-dimensional embodied conversational agent (ECA) with MPEG-4 animation standard. It is able to communicate using a rich palette of verbal and nonverbal behaviours. Greta can talk and simultaneously show facial expressions, gestures, gaze, and head movements. Two standard XML languages (FML and BML) allow the user to define her communicative intentions and behaviours based on standard SAIBA architecture (SAIBA is a common framework for the autonomous generation of multimodal communicative behaviour in Embodied conversational agents (Kopp, et al., 2006)).

The Greta platform is a fully SAIBA compliant system for the real-time generation and animation of ECA's verbal and nonverbal behaviours. The Greta platform is contained in a java program, called Modular.jar, which implements an agent without coding but just using graphic modules. The modular architecture of this platform supports the interconnection with external tools, Cereproc text-to-speech engine, enhancing an agent's detection and synthesis capabilities.

With Modular application different configurations can be built by adding the modules available in the menu (Figure 4). All menus, modules, connectors, and styles are defined in an XML file named Modular.xml.

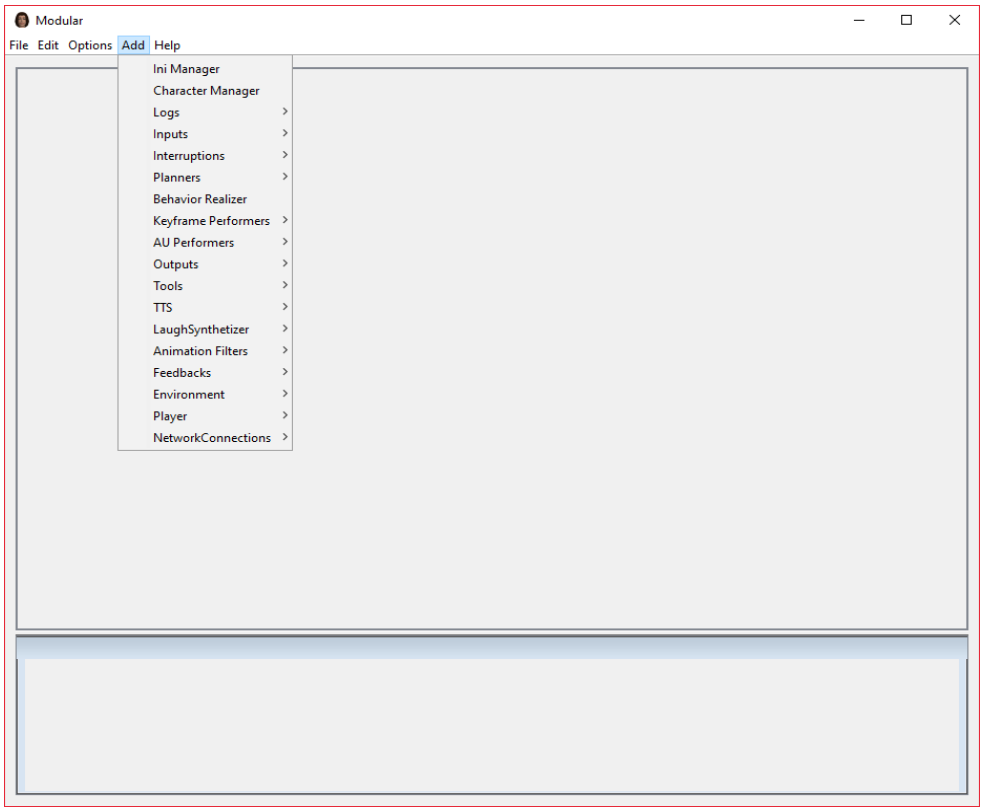


Figure 4: Modular application.

In Figure 5, a basic configuration containing the main modules is shown. This configuration allows to send FML or BML files, process them and schedule verbal and non-verbal behaviours. Once the list of signals is obtained, these are sent to their generators so that the skeleton can move and be displayed in the virtual environment. The main modules and their functionalities are introduced below.

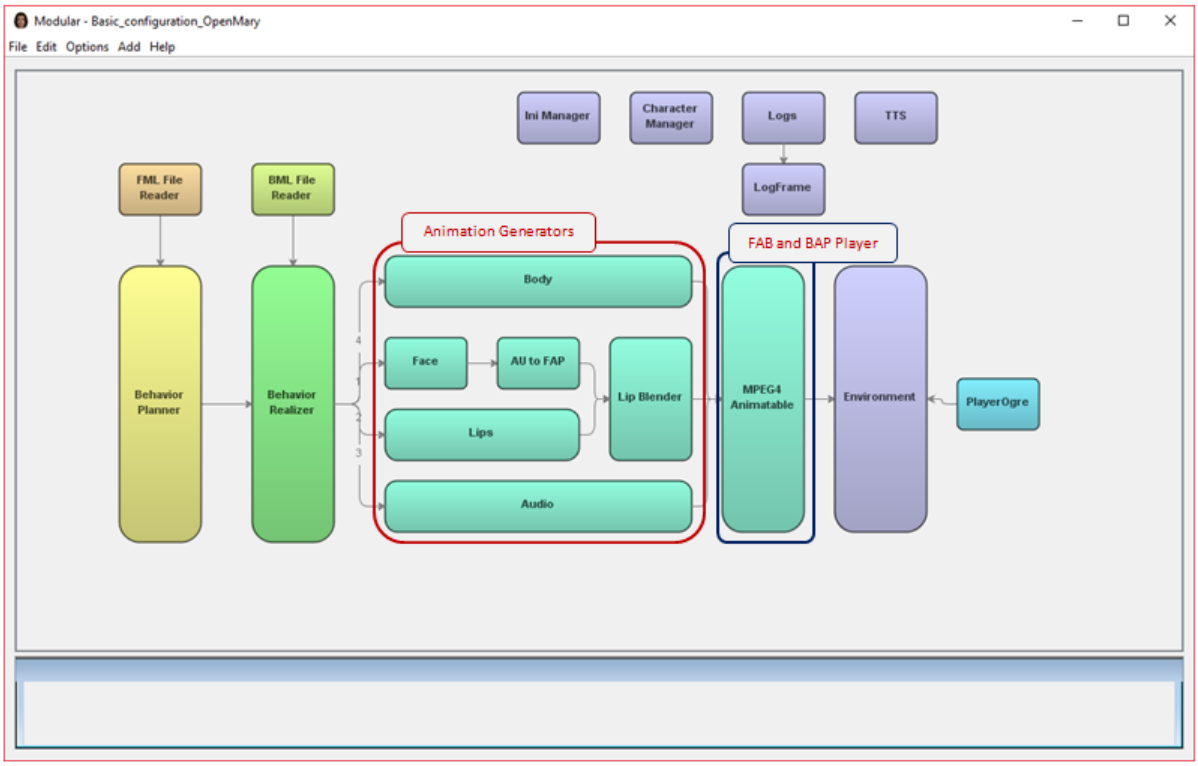


Figure 5: Modular - Basic Configuration (single agent support).

We can explain the role of each module by following the workflow that FML and BML files take to be processed and to be translated into real verbal and non-verbal behaviours. The workflow can start from a very high level, like the FML file, or from lower level, BML file. The FML represents what an agent wants to achieve: its intentions, goals, and plans. The BML language allows us to specify the signals that can be expressed through the agent communication modalities (head, torso, face, gaze, body, legs, gesture, speech, lips).

4.1.1 FML File Reader

It takes as input the XMLtree file in FML format then translate it in a list of intentions. The FML file is divided in two parts (see Figure 6): the first is the BML that can contain the speech signal, boundary signals, and pitch accents; the second part is the FML that contains the intentions (emotion, certainty, laugh, performative, deictic, etcetera). The list of intentions is sent to the following module, the BehaviorPlanner.

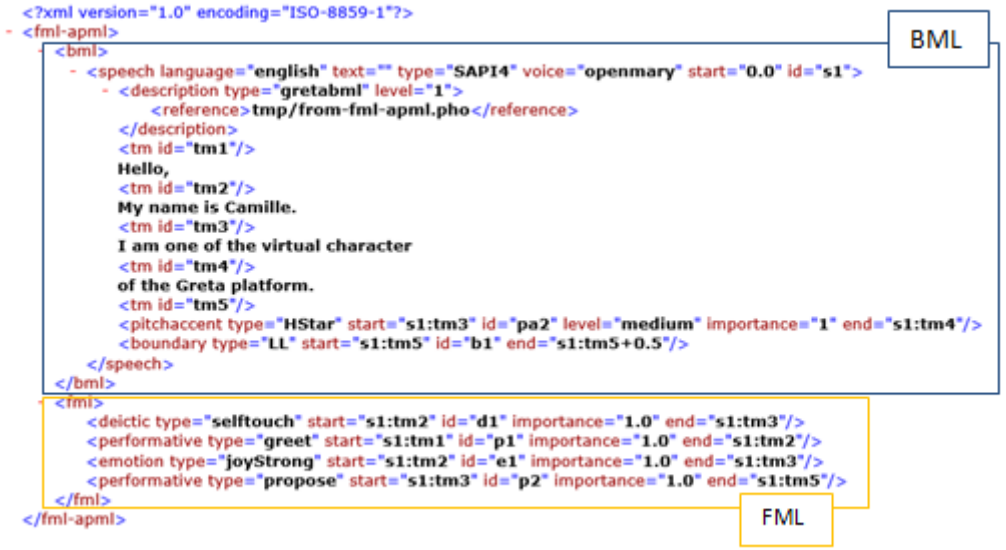


Figure 6: FML example.

4.1.2 BehaviorPlanner

Once the module receives the list of intentions, they are temporized and sorted by importance, starting time and duration. For each intention, the corresponding behaviour-set in extracted from the default or character specific lexicon. The behaviour-set can contain different signals (HeadSignal, Gesture, FaceSignal, TorsoSignal, etc.) that are extracted and added to the list of signals sent to the next module, the BehaviourRealizer.

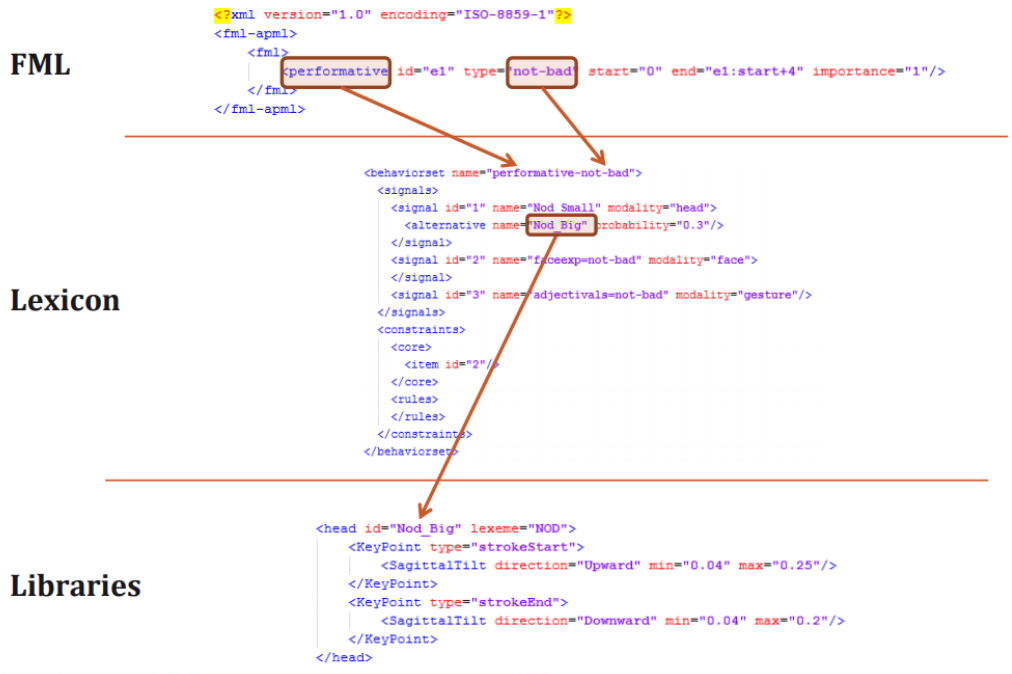


Figure 7: Example of FML, Lexicon and Libraries dependencies.

4.1.3 Behavior Realizer

This module receives the list of signals sent by the BehaviorPlanner or the "BMLFileReader" module as input.

The BMLFileReader takes as input, the BML file that contains the text to be spoken and/or a set of nonverbal signals to be displayed. All signals like facial expressions, gaze, gestures, torso movements are described symbolically in repository files (libraries).

The agent's speech, which is also part of the BML input, is synthesized by an external TTS system (TTS Module). The TTS system provides the list of phonemes and their respective duration. This information is used to compute the lip movements.

Once received the signals' list, the BehaviorRealizer follows some steps to translate the signal in keyframe:

- To calculate start and end times for each signal;
- To sort the signals;
- To identify which phase is realized in each signal and when it is realized. In this way the signals can be scheduled independently;
- The phases are translated into keyframes;
- Sort the keyframe and store them in a list;
- Send each keyframe to the correspondent KeyframePerformer (i.e. HeadKeyframe are sent to HeadkeyframePerformer, TorsoKeyFrame to TorsoKeyframePerformer, etcetera).

The BehaviorRealizer solves eventual conflicts between the signals that are scheduled to happen on the same modality at the same time. It uses repository files of predefined facial expressions, gestures, torso movements and so on.

4.1.4 Keyframe Performers

The keyframe list contains different kind of keyframes: AUKeyframe (for face expression and gaze), BodyKeyframe (for Head, Torso, Shoulder and Arms gestures), SpeechKeyframe. There are sent to the correspondent Keyframe performer (i.e. BodyAnimationPerformer, FaceKeyframePerformer, LipModel, AudioPerformer). Each keyframe is translated as FAP (Face Animation Parameter) or BAP (Body Animation Parameter) keyframe, or audio and then sent to the MPEG-4 module.

4.1.5 MPEG-4

This module implements the MPEG-4 Face and Body Animation (MPEG-4 FBA) [ISO14496] International Standard for the users of Visage Technologies software. It provides the details on the Face Animation Parameters (FAPs) and Body Animation Parameters (BAPs).

For the face, the MPEG-4 specification defines 66 low-level Face Animation Parameters (FAPs) and two high-level FAPs. The low-level FAPs are based on the study of minimal facial actions and are closely related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of most natural facial expressions. Exaggerated values permit the definition of actions that are normally not possible for humans, but could be desirable for cartoon-like characters.

For the body, there are 196 Body Animation Parameters (BAPs). BAP parameters are the angles of rotation of body joints connecting different body parts. These joints include: toe, ankle, knee, hip, spine (C1-C7, T1-T12, L1-L5), shoulder, clavicle, elbow, wrist, and the hand fingers.

The hands are capable of performing complicated motions and are included in the body hierarchy. There are totally 29 degrees of freedom on each hand, assuming that the hand has a standard structure with five fingers.

The MPEG-4 module is the skeleton of the agent. Once the BAP or FAP keyframes are received, MPEG-4 module allows to actually move the character.

4.1.6 Environment

The Environment module is used to add characters and objects into a scene. This module creates a tree of the scene, where characters and objects are leaves. Each leaf contains three nodes for direction, rotation, and scale. Because this tree is compatible with computer graphics standards, this environment can be used in different graphic engines like Ogre3D or Unity3D.

4.1.7 Editors

In the Greta System there are also modules that allow to edit gesture, libraries, or the environment, like:

- **Gesture Editor**: it allows to create the own proper gesture choosing:
  - Arm position: defines the gestures arm position by symbolic value gestures (see MecNeill sectors)
  - Hand shape: defines the hand shape key frame which is already defines in library
  - Hand orientation: defines the hand orientation by using different combination of directions
  - Trajectory: defines the trajectory for arm gesture by adding new key frames on the path
  - Openness: changes the gesture form (elbow space) in IK
  - Expressive parameters: defines the expressivity that can influence the variation of a whole gesture;
- **Hand shape editor**: it allows to choose the proper hand shape by editing the joints' orientation of the fingers;
- **Face Library Viewer**: it allows to edit the all 62 face Action Unit (AU) and so create the own facial expression;
- **AU Library editor**: it allows to edit the position of the main face AU (just 16/62) by using a graphic panel where it is drawn the face and the small point indicating the position of AU;
- **Environment editor**: it allows to select all the object or agents displayed in the scene and change their position, scale and orientation. It is also possible to add new objects.

4.1.8 Character Manager

This module contains information (see Figure 8) about the characters that can be used by the system. The data provided is about the path where the lexicons, libraries, or the voice, etc. are stored. All data contained in CharacterManager module are accessible by any of the other modules.

There are two characters (Camille and DefaultCharacter) available on the GitHub version of the Greta platform.

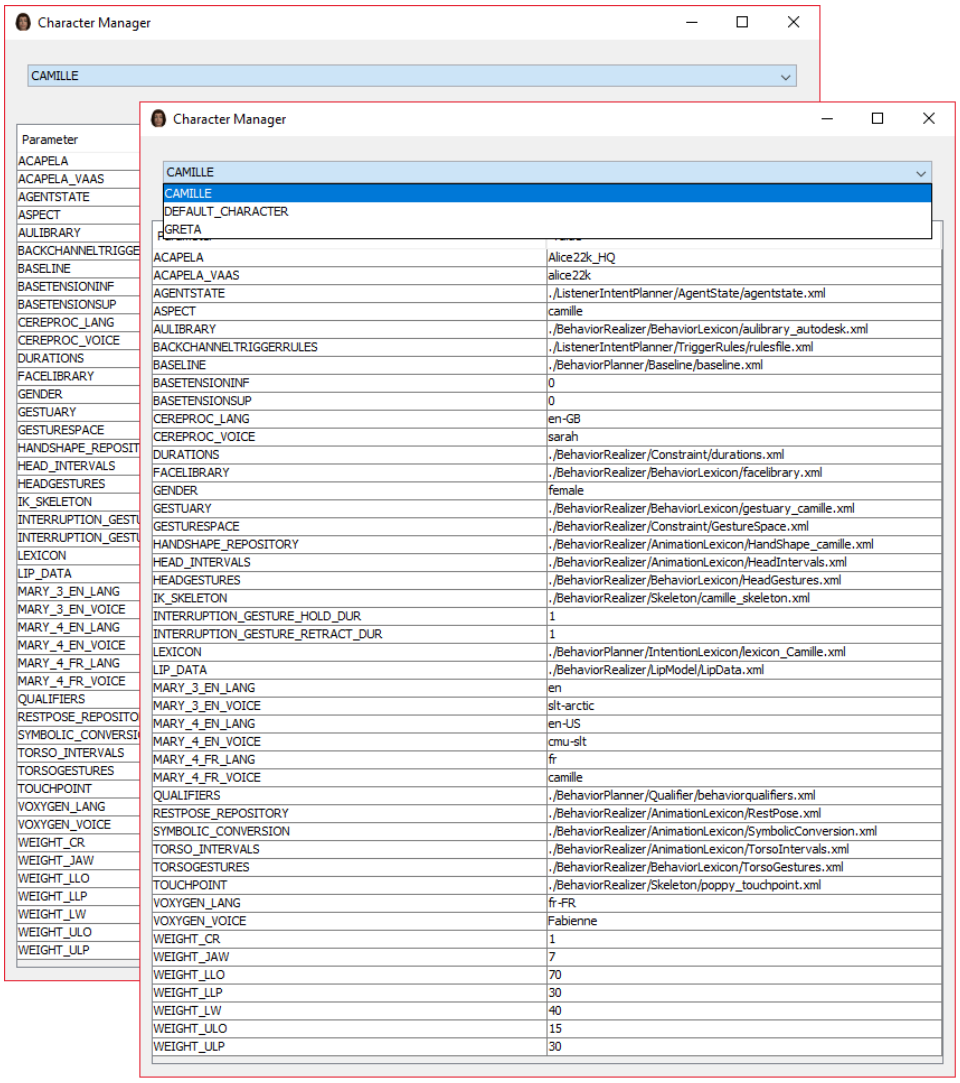
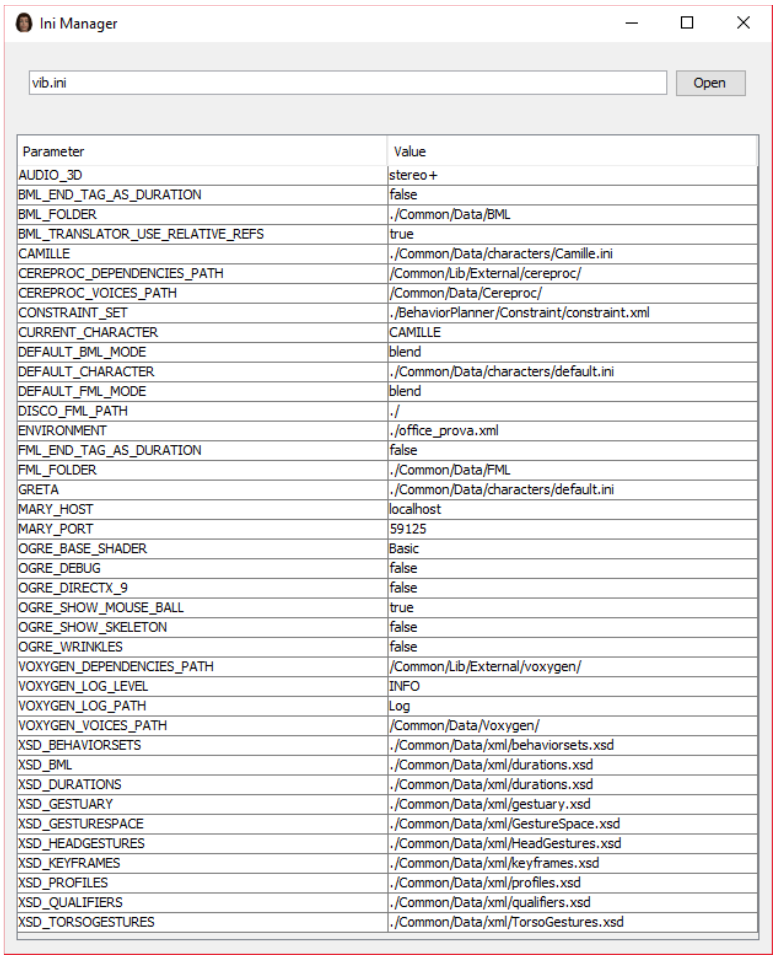


Figure 8: Character Manager module.

4.1.9 INI Manager

This module contains information stored in "vib.ini" file (see Figure 9). It displayed the info about all characters' path file, external TTS software (OpenMary and Cereproc) and Ogre3D engine, etc. All data contained in INIManager module are accessible by any of the other modules.



Parameter	Value
AUDIO_3D	stereo+
BML_END_TAG_AS_DURATION	false
BML_FOLDER	./Common/Data/BML
BML_TRANSLATOR_USE_RELATIVE_REFS	true
CAMILLE	./Common/Data/characters/Camille.ini
CEREPROC_DEPENDENCIES_PATH	/Common/Lib/External/cereproc/
CEREPROC_VOICES_PATH	/Common/Data/Cereproc/
CONSTRAINT_SET	./BehaviorPlanner/Constraint/constraint.xml
CURRENT_CHARACTER	CAMILLE
DEFAULT_BML_MODE	blend
DEFAULT_CHARACTER	./Common/Data/characters/default.ini
DEFAULT_FML_MODE	blend
DISCO_FML_PATH	./
ENVIRONMENT	./office_prova.xml
FML_END_TAG_AS_DURATION	false
FML_FOLDER	./Common/Data/FML
GRETA	./Common/Data/characters/default.ini
MARY_HOST	localhost
MARY_PORT	59125
OGRE_BASE_SHADER	Basic
OGRE_DEBUG	false
OGRE_DIRECTX_9	false
OGRE_SHOW_MOUSE_BALL	true
OGRE_SHOW_SKELETON	false
OGRE_WRINKLES	false
VOXYGEN_DEPENDENCIES_PATH	/Common/Lib/External/voxygen/
VOXYGEN_LOG_LEVEL	INFO
VOXYGEN_LOG_PATH	Log
VOXYGEN_VOICES_PATH	/Common/Data/Voxygen/
XSD_BEHAVIORSETS	./Common/Data/xml/behaviorsets.xsd
XSD_BML	./Common/Data/xml/durations.xsd
XSD_DURATIONS	./Common/Data/xml/durations.xsd
XSD_GESTUARY	./Common/Data/xml/gestuary.xsd
XSD_GESTURESPACE	./Common/Data/xml/GestureSpace.xsd
XSD_HEADGESTURES	./Common/Data/xml/HeadGestures.xsd
XSD_KEYFRAMES	./Common/Data/xml/keyframes.xsd
XSD_PROFILES	./Common/Data/xml/profiles.xsd
XSD_QUALIFIERS	./Common/Data/xml/qualifiers.xsd
XSD_TORSOGESTURES	./Common/Data/xml/TorsoGestures.xsd

Figure 9: INI Manager.

4.2 Multi-Agent Model

The Greta system supports different characters in the same environment and the agents are independent from each other.

In order to have the multi-character platform in the Greta system (see Figure 10), a hierarchy of modules is created. The UI allows us to build and arrange modules as trees. This means that the modules can have a parent, and if so, their component are instantiated, not via the default constructor, but via the constructor with the parent's reference.

Most of the modules needs to know which CharacterManager reference to use. Therefore, in order to add a new character, the CharacterManager module has to be added first and once selected, the other modules can be added as its child. In this way each agent has its own workflow: FML or BML files can be run independently for each character.

Earlier, there was an assumption that allowed only one character at a time. Hence, several modules used a static reference to a static (singleton) CharacterManager. In the updated version, this class is not a singleton anymore and it is the parent module of all modules depending on it (i.e. FMLFileReader, BehaviorPlanner, BMLFileReader, BehaviorRealizer, MPEG4Animatable, TTS, etc.). In order to do this, the CharacterDependentInferface has been updated to add the CharacterManager reference and constructors now need a CharacterManager reference and are CharacterDependent.

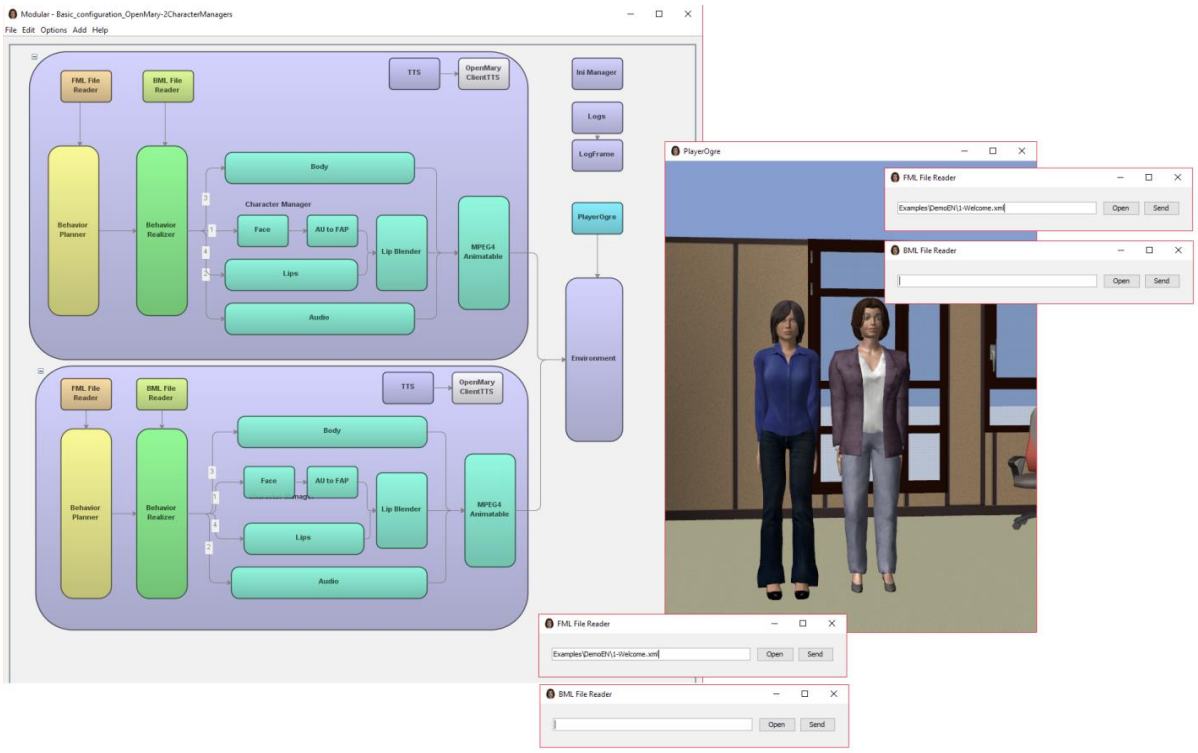


Figure 10: Configuration of Greta system with two characters.

4.3 Gaze Model

Gaze is the one of the nonverbal cues used by humans during interactions. It can be used to communicate, manage attention and trigger people in the social and cognitive processes. It can help to facilitate the natural interaction between ECAs and the humans.

In the Greta platform it is possible use the gaze behaviour. The gaze model built in the platform is based on the study of (Pejsa, Andrist, Gleicher, & Mutlu, 2015).

The gaze behaviour is based on coordinated movements of the eyes, head, shoulder and trunk toward the objects and information in the environment (like, other agents). The model takes as input the position of the target and the orientation of the body parts involved in the movements (eyes, head, shoulder and torso). Given these data and applying a set of kinematic laws derived from measurements of primate gaze reported in neurophysiology research (Guitton & Volle, 1987) (McCluskey & Culle, 2007), the model computes the shifts of the body parts necessary to look at the target. The shifts are achieved by rotating the body joints, distributed into several groups that are controlled jointly. Neck joints (cervical vertebrae) are grouped together under the head. Similar to humans, the rotation of the head involves the simultaneous rotation of the joints that constitute the head. In the same way, shoulders rotation is achieved through simultaneous rotation of the thoracic vertebrae, and so trunk rotation is referred to distributed rotation among lower spine joints (lumbar vertebrae).

The gaze behaviour is performed by the agents based on commands received by FML or BML. To process the BML or FML command, the Environment module should be connected to the BehaviorRealizer. This allows to retrieve angles to the target. If it is not connected, the agent will not be able to process gazes with a target but only with `offsetDirection` and `offsetAngles`.

4.3.1 BML Command

In the BML two different type of gaze can be specified:

1. <gaze> to temporarily direct the gaze of the character towards a target;

2. <gazeShift> to permanently change the gaze direction of the character towards a certain target.

<gaze>

Below an example of BML command for the simple gaze behaviour:

<bmlxmlns=http://www.bml-initiative.org/bml/bml-1.0 character="Alice"id="bml1">
<gazeid="gaze1"start="1"end="10"influence="HEAD"target="bluebox"/>
</bml>

The syntax of the command:

NAMESPACE:	http://www.bml-initiative.org/bml/bml-1.0
ELEMENT:	<gaze>
SYNC POINTS:	start, ready, relax, end
ATTRIBUTES:	• id: unique ID that allows referencing to a particular <bml> behavior. The id 'bml' is reserved; • Target: a reference towards a target instance that represents the target direction of the gaze; • Influence: determines what parts of the body to move in order to perform the gaze direction. It can be: EYES, HEAD, SHOULDER, TORSO; • OffsetAngle: adds an offsetangle in degrees to gaze direction relative to the target in the direction specified in the offsetDirection; • OffsetDirection: direction of the offsetAngle can be RIGHT, LEFT, UP, DOWN, UPRIGHT, UPLEFT, DOWNLEFT, DOWNRIGHT; • sync attributes: - start: gaze start to move - ready: gaze target acquired - relax: starts returning to default direction - end: gaze returned to default position
CONTENTS:	none

Table 1: Syntax of gaze BML command.

The command in the example indicates the agent to gaze towards the target (bluebox). Starting from second 1 the agent moves to reach the target, using eyes and head (influence="HEAD"), and within second 9 it will come back to the starting position. This behaviour causes the character to temporarily direct its gaze to the requested target.

The influence parameter is read as follows: EYE means 'use only the eyes'; HEAD means 'use only head and eyes to change the gaze direction', etcetera.

<gazeShift>

Below an example of BML command for the simple gaze behavior:

<bmlxmlns="http://www.bml-initiative.org/bml/bml-1.0" character="Alice" id="bml1">
<gazeShift id="gaze1" start="1" end="2" influence="HEAD" target="bluebox"/>
</bml>

The syntax is the same, except for the sync attributes; they will include just "start" (gaze start to move to the new target) and "end" (gaze target acquired).

In this example, the command indicates the agent to change the default gaze direction to be towards the blue box. The shift in gaze takes 1 second to be ready and will employ the eyes and the head.

This behaviour causes the character to direct its gaze to the requested target. This changes the default state of the agent: after completing this behaviour, the new target will be the default gaze direction of the character.

4.3.2 FML Command

In the FML file the intention to control the gaze behaviour of the agent can be included. The command can tell the character to look at something or use the gaze to express an emotional state (i.e. when we are sad, we look down without a precise target).

To just express an emotional state, the intention (performative, emotion, iconic, etc.) can include a gaze expression. As shown in the Figure 11, an intention can include (in the lexicon) a gaze expression whose value is specified in the Facelibrary.

Figure 11: example of FML intention.

To look at something or someone, the gaze behaviour has to be included in the **deictic** intention. It can be added as an attribute "**target**" that gives the ID of the element to gaze at. As shown in the Figure 9, the FML file has a deictic intention with a target attribute. In this case there is no gaze expression in the lexicon and in the Facelibrary. The attribute target allows to add a new GazeSignal to the list of signals belonging to that deictic intention. Like in the example, the deictic-selftouch intention is made by just gesture signal. The target attribute in the FML file allows to add a gaze signal to look at the "Andre_chair0" in the same time slot (start and end) of gesture signal.

Figure 12: FML deictic intention containing gaze command.

4.3.3 Target attribute

The "target" attribute both in FML and BML is required to gaze at something. In Greta, we allow BML messages that do not contain a "target" attribute but contain both "offsetDirection" and "offsetAngle" attributes: with this kind of messages, it is possible to look at directions defined in the GazeDirection enumeration.

The target attribute should be a reference to leaf of a TreeNode in the Greta Environment (see Figure 13). It is possible to access the leaf of that object, have the position in the environment and compute the gaze shift to orientate the agent to the target. Therefore, everything included in the XML TreeNode Environment can be a target for the Gaze. When we add an object, the position of the object should be set to the base position of the object on the x,y,z plane and not at the center of the object itself. This is done since the position of the target is computed during gaze shift by calculating the sum of the y position of the base position of the object and the half height of the object.

Now the TreeNode environment contains just the object in the scene and not the position of the agents and their body parts. In order to gaze the other agents, the wording for the target attribute as to be "Agent:*nameAgent*" (i.e. target="Agent:DEFAULT_CHARACTER"). The name of the Agent is searched among the character in scene and once found the position of the head and eyes is take in order to compute the right rotation angle.

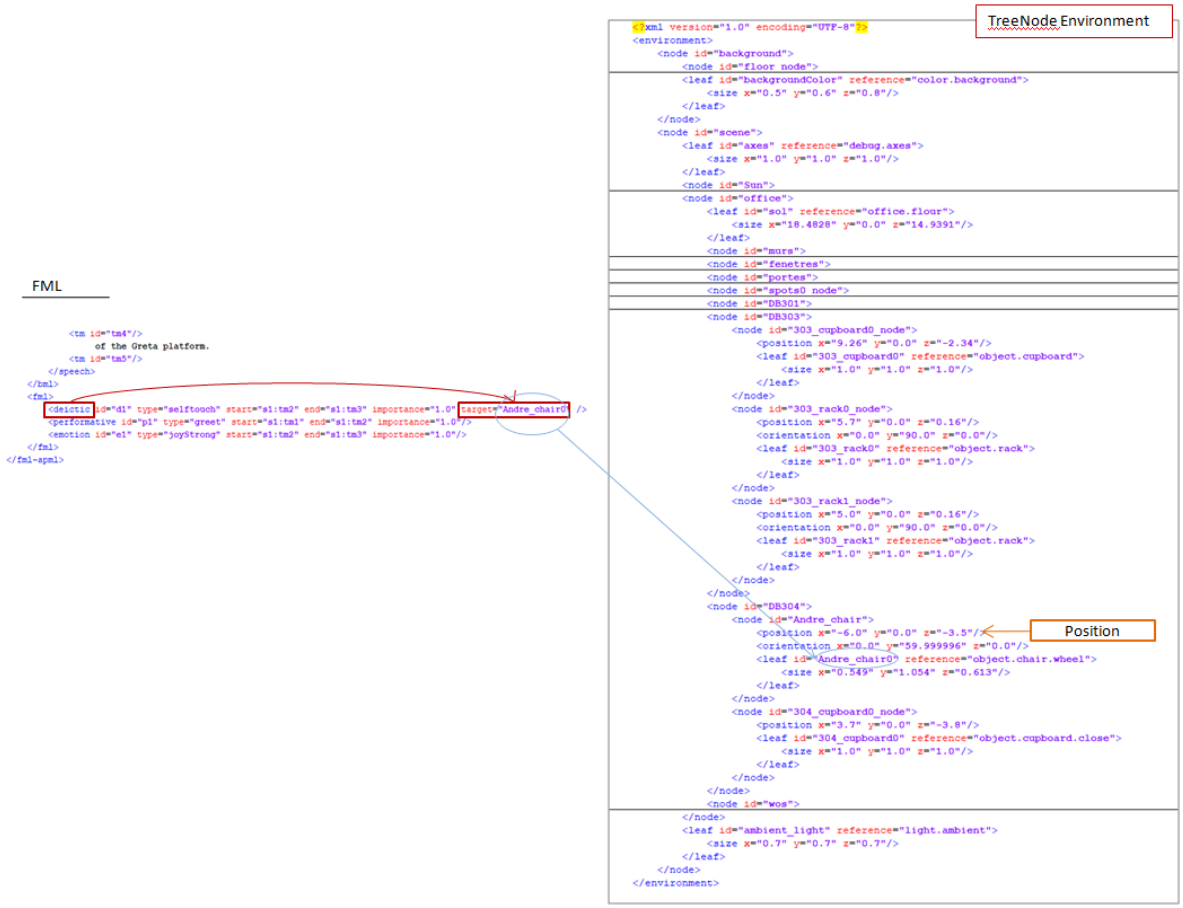


Figure 13: example of a target found in the environment TreeNode.

4.3.4 Influence attribute

The influence attribute indicates which body part should be moving in the gaze movement. This attribute should match one of the entries in the Influence enumeration: EYES, HEAD, SHOULDER, TORSO. It is possible chose the influence but can also not be specified and according to the overall rotation the system will compute the necessary influence.

For each influence, one should keep in mind that there is a physical limit to movements. For instance, if you want to look at a target that is behind Greta you should use at least SHOULDER influence.

The angle limits are not totally justified. It is used as reference the study of (Pejsa, Andrist, Gleicher, & Mutlu, 2015) and also a logic based on 'folk's wisdom'.

	Pitch	Yaw	Roll
Eyes	45°	60°	/
Head	45°	90°	/
Shoulder/Torso	/	45°	/

Table 2: Influence angle limits.

5 Evaluations

Besides the continuous end-user evaluations that are performed in the Council of Coaches project in Work Package 2 (T2.3: User Needs and Continuous Evaluation, reported in D2.4, D2.5 and D2.6), we describe here more technological and fundamental evaluations performed with the Council of Coaches virtual coaching platform. Section 5.1 describes a technical evaluation of the Council of Coaches-as-a-Platform paradigm, while Section 5.2 describes an in-progress evaluation on the fundamental value of group-based discussions.

5.1 Test with Interaction Technology students

The Council of Coaches project was one of the topics of the course Foundations of Interaction technology at the University of Twente. 42 students followed the course and were divided into nine project groups of 4 – 6 students. During one week (two lectures of 1:45h and a group challenge) the students were introduced to the topic of Behaviour Change Support Systems (BCSS) and as an example the Council of Coaches project. During the first lecture we introduced the high-level concepts of BCSS, we presented an overview of the Council of Coaches project and there were invited talks about the different types of work done in the project from partners in Twente (work package 3, work package 4, work package 6). At the end of the lecture we discussed the Council of Coaches system (based on the work of Technical Integration meeting in October 2018), showed a demo and explained the BML language.

The challenge was to design a conversation with three coaches and to think about how to make use of the concept and the possibilities of having three coaches instead of one. The students first had to think of the concept of possibility they wanted to use. After that they had to design the content and act out the dialogue with three of their project members and record the conversation. When the content was designed, the students had to try to translate the content into the Council of Coaches system. A minimal version of the system was prepared (ASAP and Unity version only) and shared with the students. Instructions on how to install (for Windows 10) and use the system, including the reference to the BML specifications, were shared and the examples shown during the lecture were included in the distributed software.

The students could install the software on the own computers or laptops or could make use of one of the PCs in our lab. In the end all groups managed to work with the Council of Couches system and translated their dialogue into the system. Students also had to record the dialogue with the virtual agents.

During the challenge students experienced problems and issues during the setup of the environment and the installation and use of the system. The instructions written for the challenge were based on Windows 10 Pro. In practice, there are differences between Windows 10 Home, Pro and Education. Updating from Home to Education solved some of the issues.

Three groups were able to install and work with the system on their own computers without asking for help. Three groups had to work on one of the PCs in our lab to install and work with the system and the other three groups we able to install and work with the system on their own PC or laptop, but had some problems with the performance of the system. Performance issues ranged from dropping behaviours by asap (e.g. not showing nonverbal behaviour) to not showing any behaviour at all. This was due to the fact that the Council of Coaches system requires powerful and up to date hardware. The PCs and laptops of the students ranged from low end to high end. Other performance issues were due to the fact that students tried to send all BML blocks at once, which results dropping behaviours by ASAP to frozen systems. Other problems and issues deal with closing the different process of the system. Closing a

process in a wrong way will result in a process that is still running in the background, consuming resources and causing unexpected behaviour.

At the end of the challenge, all project groups designed a dialogue where they tried to make use of the possibilities of having multiple coaches instead of one coach. The concepts ranged from having different perspectives and views to a problem to having a peer or buddy in the dialogue that is asking questions for the end user of the system. All groups were able to work with this (minimal) version of Council of Coaches system, but the installation and setting up of the environments before using the system should be improved.

5.2 Influence of group discussion

When a group of people make a decision together, this often happens after some discussion. This ideally helps them use more of the available knowledge in the group, look at the issue from multiple angles, find common ground to base their decision on, arrive to the best solution, and create commitment to a decision. These kinds of discussions usually have participants engaged, as they all care about the outcome. Furthermore, these discussions also impact the group members opinions of each other. On the flipside, a group discussion can be potentially confusing or irritating to some group members, as a lot of people might be talking and shifting the topic of discussion rather rapidly, the decision by the group might not match that of some of the group members, and people might not feel heard.

In the previous paragraph we discussed all sorts of factors that might come in to play in discussions between several people. We wonder how this would translate to a discussion with a group of virtual characters talking to a user. In this case, in the setting where the virtual characters are all coaches, and the user is their client coming to them for advice on topics to do with health and wellbeing. One could imagine the discussion being engaging, virtual coaches coming across as more animate and intelligent and more capable of coaching, the discussion leading to more reflection on the decision and reasoning for it by the user, and the user showing more commitment to the decision. On the other hand, irritation and confusion might also come from this approach. To find out how to best approach group coaching with virtual coaches, we will investigate these potential effects of an interaction format containing discussion versus one that does not contain discussion and merely showcases the different viewpoints of the coaches.

5.2.1 Purpose and questions

The objective that we are trying to achieve with this study, is to find out whether a discussion with an embodied conversational coaching team is beneficial to users.

The main questions we try to answer in this study are:

- Are users more engaged, enjoying and preferring interactions with coaches that discuss their viewpoints?
- Do coaches that discuss their viewpoints seem more animate and intelligent (“real and competent”)?
- Does the discussion with the coaches make the user appreciate their coaching ability more?
- Does the discussion with the coaches lead to more commitment to follow the chosen solution?
- Does the discussion with the coaches make the user reflect more on their options and reasons for choosing so?

5.2.2 Methods

The study will be a within-subject comparison of two different group interactions with three embodied conversational coaches on a computer: a group discussion versus a simple sharing of opinions. The content of these interactions will be similar, but in the discussion scenario characters will more clearly show that there is a conflict between their opinions and in the no discussion scenario they will simply present the conflicting opinions without this discussion. The dialogue will be written in Yarn and presented by the ASAP agents using the current Council of Coaches technical prototype (October 2018).

We plan to use the following questionnaires to answer our questions:

- Quality of coaching questionnaire (adapted version Coaching Behaviour Scale for Sport (Cote, Yardley, Hay, Sedgwick, & Baker, 1999)) to assess: opinions on coaching ability of coaches
- Godspeed questionnaire (Bartneck, Kulic, Croft, & Zoghbi, 2009) about the characters to assess: Feelings on anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety
- GEQ (IJsselsteijn, de Kort, & Poels, 2013) questionnaire to assess: perceived engagement
- Questionnaire/my own questions on commitment to the behaviour to assess: reported commitment to change behaviour
- Questionnaire/my own questions on reflection on choice and reasoning to assess: reported thinking on possible options and reflection on reasons to pick each option

We plan to use a 2x2 Repeated Measures ANOVA (gender x experience with virtual environments and characters) for statistical analysis with $\alpha = 0.05$, $\beta = 0.95$, and to try to find any difference between our two scenarios with at least a medium effect size (0.25). We will check whether this differs between sexes and people less or more experienced with virtual environments and characters. This would require a sample size of about 60 participants.

5.2.3 Participants

Participants will be a convenience sample of university students. Preferably these will be first year students, as they might vary on their experience with virtual environments and characters. We will invite students in our program to participate. We will ask participants for their sex and level of previous experience with virtual environments and characters.

General inclusion criteria for the user study are:

- Fluency in Dutch
- Proficiency in English
- Willingness to provide informed consent
- No cognitive impairments
- No visual or hearing impairments
- Ability to work with a mouse and/or touchpad

There are no potential risks and no anticipated benefits to individual participants.

5.2.4 Procedure

1. Participant comes in and researchers introduce themselves.
2. Reading of the information letter and informed consent by participant.
3. Questions and remarks by participant and necessary further explanation by the researchers.

4. Signing of informed consent and filling out general information by participant (sex, previous experience virtual environments and characters).
5. Participant is seated behind computer/laptop.
6. An interaction with either the **discussion OR no discussion scenario**: randomly assigned order
    a. Standard introduction of characters and topic to discuss
    b. Presentation of opinions either with OR without discussion
    c. Choice of participant for a certain solution
    d. Closing of conversation with coaches saying goodbye
7. Filling out the questionnaires for this scenario.
8. Second interaction with either the **discussion OR no discussion scenario**: whichever scenario was not done yet
    a. Standard introduction of characters and topic to discuss
    b. Presentation of opinions either with OR without discussion (the option that was left)
    c. Choice of participant for a certain solution
    d. Closing of conversation with coaches saying goodbye
9. Filling out the same questionnaires again for this scenario.
10. Debriefing by the researchers and possibility for participants to ask questions, make remarks, and ask to be excluded from the study.

6 Bibliography

Bartneck, C., Kulic, D., Croft, E., & Zoghbi, S. (2009). Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots. *International Journal of Social Robotics*, 71-81.

Cote, J., Yardley, J., Hay, J., Sedgwick, W., & Baker, J. (1999). An exploratory examination of the coaching behaviour scale for sport. *AVANTE*, 82-92.

Guitton, D., & Volle, M. (1987). Gaze control in humans: Eye-head coordination during orienting movements to targets within and beyond the oculomotor range. *Journal of Neurophysiology*, 427–459.

IJsselsteijn, W. A., de Kort, Y. A., & Poels, K. (2013). *The Game Experience Questionnaire*. Eindhoven: Technische Universiteit Eindhoven.

Kopp, S., Krenn, B., Marsella, S., Marshall, A. N., Pelachaud, C., Pirker, H., . . . and Vilhjalmsson, H. H. (2006). Towards a common framework for multimodal generation: the behavior markup language. *Proceedings of the 6th international conference on Intelligent Virtual Agents* (pp. 205–217). Berlin, Heidelberg: Springer-Verlag.

McCluskey, M. K., & Culle, K. E. (2007). . Eye, head, and body coordination during large gaze shifts in rhesus monkeys: Movement kinematics and the influence of posture. *Journal of Neurophysiology*, 2976–2991.

Pecune, F., Cafaro, A., Chollet, M., Philippe, P., & Pelachaud, C. (2014). Suggestions for Extending SAIBA with the VIB Platform. *Workshop on Architectures and Standards for IVAs, held at the '14th International Conference on Intelligent Virtual Agents (IVA 2014)'* (pp. 16–20). Boston: Bielefeld eCollections.

Pejsa, T., Andrist, S., Gleicher, M., & Mutlu, B. (2015, March). Gaze and Attention Management for Embodied Conversational Agents. *ACM Transactions on Interactive Intelligent Systems,*.

van Waterschoot, J., Bruijnes, M., Flokstra, J., Reidsma, D., Davison, D., Theune, M., & Heylen, D. (2018). Flipper 2.0: A Pragmatic Dialogue Engine for Embodied Conversational Agents. *Proceedings of the 18th International Conference on Intelligent Virtual Agents* (pp. 43-50). ACM.