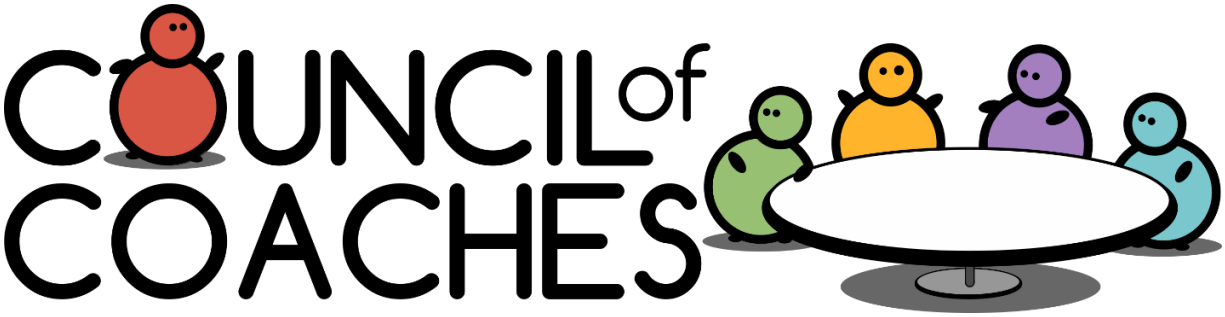

D7.4: Third functional prototype

Dissemination level: Public

Document type: Demonstrator

Version: 1.0.0

Date: June 3rd, 2019

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	Third functional prototype
Due date of deliverable	May 31st, 2019
Work package	WP7
Author(s)	Dennis Reidsma (CMC), Merijn Bruijnes (CMC), Gerwin Huizing (CMC), Randy Klaassen (CMC), Kostas Konsolakis (CMC), Marcel Weusthof (CMC), Oresti Baños (CMC), Jorien van Loon (CMC), Tessa Beinema (RRD), Silke ter Stal (RRD), Dennis Hofs (RRD), Boris van Schooten (RRD), Harm op den Akker (RRD), Mark Snaith (UDun), María Jesus Arnal (UPV), Jose Luís Bayo (UPV), Álvaro Fides (UPV), Reshmashree Bangalore Kantharaju (SU), Donatella Simonetti (SU)
Reviewer(s)	Kostas Konsolakis (CMC)
Approved by	Coordinator
Dissemination level	PU: Public
Document type	Demonstrator
Total number of pages	26

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, Grupo SABIEN (UPV)
- Innovation Sprint (iSPRINT)

Abstract

A description of the third version of the Functional Prototype in its two variants: the design-oriented Functional Prototype used to gather feedback from users and the Technical Prototype where all the technical integration progress is consolidated. This document also reports the progress in the development process, and includes a summary of the innovation beyond the state of the art for each module.

Table of Contents

1 Introduction 5
2 Objectives 6
3 Development process 7
3.1 Technical integration taskforce 7
3.1.1 Workshop 4 7
4 Functional Prototype software 8
4.1 Canvas & Splash Screen 8
4.2 Main Menu 10
4.3 Creating an account 13
4.4 The Living room (Main User Interface) 16
4.5 Widgets 18
5 Technical Prototype software 19
5.1 Pre-requisites 19
5.2 Installation 19
5.3 Execution 20
5.3.1 Android 21
5.3.2 Stopping 21
6 Innovation beyond state of the art 22
6.1 Functional Demonstrator 22
6.2 Technical Demonstrator 22
6.2.1 Innovation in Measuring User Behaviour 22
6.2.2 Innovation in Automated Personalized Multi-Party Dialogue 23
6.2.3 Innovation in Multi-Party Embodied Conversational Agent Systems 23
7 Bibliography 25

List of figures

Figure 1: Council of Coaches BETA: Splash Screen. 8
Figure 2: Council of Coaches BETA on a wide screen. 9
Figure 3: Council of Coaches BETA on a narrow screen. 9
Figure 4: Council of Coaches BETA Main Menu. 10
Figure 5: Council of Coaches BETA > Main Menu > Credits. 11
Figure 6: Council of Coaches BETA > Main Menu > Patch Notes. 11
Figure 7: Council of Coaches BETA > Main Menu > Privacy Statement. 12
Figure 8: World of Warcraft (2006) login screen, serving as inspiration for the Council of Coaches Main Menu. 12
Figure 9: Council of Coaches BETA > Creating Account > Step 1. 13
Figure 10: Opening character creation and tutorial scene in Bethesda's Morrowind, see https://youtu.be/VgR2ISC6Ets?t=58. 14
Figure 11: Creating an account, entering your email address. 15
Figure 12: Creating an account, the continue button explained. 15
Figure 13: Creating an account, the finish button. 16
Figure 14: The Council of Coaches BETA Main "Living room" interface. 16
Figure 15: The origins: Civilization 2, with 5 "coaches" in their own little box. 17
Figure 16: Second functional demonstrator user interface, with the coaches sharing a room (and table). 17
Figure 17: The first Council of Coaches Widget: the radio that provides background music. 18

Symbols, abbreviations and acronyms

ASAP	Articulated Social Agents Platform
BML	Behaviour Mark-up Language
CMC	Centre for Monitoring and Coaching
COUCH	Council of Coaches
D	Deliverable
DGEP	Dialogue Game Execution Platform
DBT	Danish Board of Technology Foundation
EC	European Commission
FML	Function Mark-up Language
HBAF	Holistic Behaviour Analysis Framework
ISPRINT	Innovation Sprint
M	Month
MS	Milestone
RRD	Roessingh Research and Development
RRI	Responsible Research and Innovation
SAIBA	Situation, Agent, Intention, Behaviour, Animation
SU	Sorbonne University
UDun	University of Dundee
UPV	Universitat Politècnica de València
UT	University of Twente
WP	Work Package

1 Introduction

This document of type "Demonstrator" is accompanying the third prototype release of the Council of Coaches project. In Council of Coaches, we distinguish between the "Functional Prototype" track, and the "Technical Prototype" track. When the development of each Prototype track reaches a milestone, it outputs a Demonstrator, which is nothing more than the Prototype build at the time of the milestone. For this reason the terms Functional Demonstrator and Technical Demonstrator are used indistinctively together with Functional Prototype and Technical Prototype in this series of deliverables. The Functional Prototype produces a working, stable, "production ready", robust demonstrator that can be evaluated with older adult end-users, while the Technical Prototype contains more state-of-the-art features, that is at the "lab" level of readiness.

This document accompanies the release of the two prototypes in the following way:

- Section 3 provides updates on the development process
- Section 4 provides a quick walkthrough of the Functional Prototype – and links to the working demonstrator that is available online.
- Section 5 provides the technical instructions necessary to run the Technical Prototype – and links to a video demonstration.
- Section 6 highlights the innovation beyond the state of the art of the modules that compose the Technical Prototype.

2 Objectives

This is the third in a collection of deliverables reporting the progress of the prototype system of Council of Coaches:

- **D7.2: Initial functional prototype (M9):** An early low-fidelity prototype of the system, using Wizard-of-Oz methodologies to compensate for functionalities still under development.
- **D7.3: Second functional prototype (M15):** Focused on delivering a more realistic and smooth user experience, including a major update of the contents (dialogue possibilities).
- **D7.4: Third functional prototype (M21):** (This document) Used to test acceptance and usability of more advanced features (agent animations and behaviours, interaction concepts).
- **D7.5: Final Council of Coaches Technical Prototype (M27):** Include all the innovations connecting Knowledge Base, Holistic Behaviour Analysis Framework with autonomous Dialogue Framework, and Embodied Conversational Agents to deliver a fully working system.

From the reporting perspective, this document acts as a reference pointing to all the actual source code, binaries, executables, the software, and accompanying instructions that build up the third functional prototype (divided into Functional Prototype and Technical Prototype). Altogether, this deliverable document and the referenced material, are the delivered third functional prototype.

In order to achieve this, this document targets three objectives:

Objective 1: To report the work done in creating the third functional prototype since the release of the second functional prototype until the release date of the third (Section 3).

Objective 2: To provide links to the third prototype software and document how to execute both the Functional Prototype and Technical Prototype (Section 4 & 5).

Objective 3: To highlight the innovation and progress of the prototype beyond the current state of the art (Section 6).

3 Development process

The development process and its management procedures, including issue management, were established and reported in the previous prototype deliverable (D7.3) and have not changed since then.

3.1 Technical integration taskforce

This taskforce has continued with its development efforts and planned integration workshop meetings, of which there has been one between the previous prototype (reported in D7.3) and the current one. It was focused mainly on preparing for the third functional prototype reported here.

3.1.1 Workshop 4

The fourth workshop meeting of the Technical Integration Taskforce was held in Enschede during the week of 25th-29th of March, 2019. The aim of this workshop was to:

- Analyse the available results of the evaluation of the Second Functional Prototype and identify new requirements based on feedback
- Enhance existing scenarios with new functionalities to be demonstrated in the Third Functional Prototype
- Lay out the development tasks until the release of the Third Functional Prototype to meet the objectives identified in the above two points
- Conduct an updated RRI mini-workshop to identify new RRI issues and tackle ongoing and pending RRI issues
- Prepare a plan for how to best show off the outcome of the tasks during the intermediate review demonstration

The concrete technical outcomes of this workshop were:

- Create new sample dialog games aimed at two scenarios: 1) Use the "Goal setting" strategy and 2) Incorporating real sensor data into a dialog (also using a Yarn script)
- Greta can now receive Function Mark-up Language (FML) from Flipper, and in return send it feedback about time markers (for synchronization). It is possible to recreate gestures based on the model made by SU
- Improved user experience of the Functional Demonstrator, including resolution scaling, login mechanism (linked to R2D2 storage system), privacy section and tutorials
- Written script to process automatic behavior detection from raw sensor data. This new behavior data can be stored in the Aware system and then be pushed to the Knowledge Base.
- Implemented the Knowledge Base using R2D2 storage system. Established push-based connection to Aware system, the Holistic Behavior Analysis Framework (HBAF) (through web2web), and the interfaces of the Functional Demonstrator. The Knowledge Base stores and feeds the dialogues among the agents in the Functional Demonstrator
- Completed the connection of universAAL Bluetooth health sensors application to Aware system and HBAF through REST APIs.

4 Functional Prototype software

The third milestone of the Council of Coaches functional prototype is available online through the following URL:

https://www.council-of-coaches.eu/beta

In the coming months, we intend to invite people to test this BETA version of the Council of Coaches demonstrator, in preparation for the final version to be used in the project's final demonstration (planned to start in February 2020).

Below we will provide a quick walkthrough of the features of this version, highlighting the differences between the previous M15 release (see D7.3).

4.1 Canvas & Splash Screen

Upon navigating to the BETA URL, users are greeted with the Splash screen (see Figure 1):

Figure 1: Council of Coaches BETA: Splash Screen.

Developer Commentary:

The splash screen was created to give a "professional" feel to the Council of Coaches application, as if starting a video game. It also gives the opportunity to credit the EU funding body, and the project partners.

Furthermore, creating this splash screen set the development process for the rest of the Council of Coaches application in terms of scalability and design. The screen designs are all made in Inkscape [1]

[1] https://inkscape.org/es/

(all graphics are vector-graphics which obviously is ideal for scaling, but also results in small file sizes for quick page-load times). The design canvas is always set to 1920x1080 or 16:9 aspect ratio, which is by far the most common for PC, Laptop and Tablet screens. As the final demonstration will be held using Tablets, this is the leading requirement in terms of screen size and aspect ratio.

But, although 16:9 aspect ratios are very common, screen resolutions differ vastly from device to device. We therefore needed a way to reliably scale the user interface to various screen sizes. For this, the entire HTML canvas is scaled to maintain its aspect ratio, while attempting to fill either the horizontal or vertical space available (or both) on every page resize event. This results in a page that will always look exactly the same, just smaller or larger depending on the screen, and with black-space in case of wide screen (see Figure 2) or narrow screens (Figure 3).

Figure 2: Council of Coaches BETA on a wide screen.

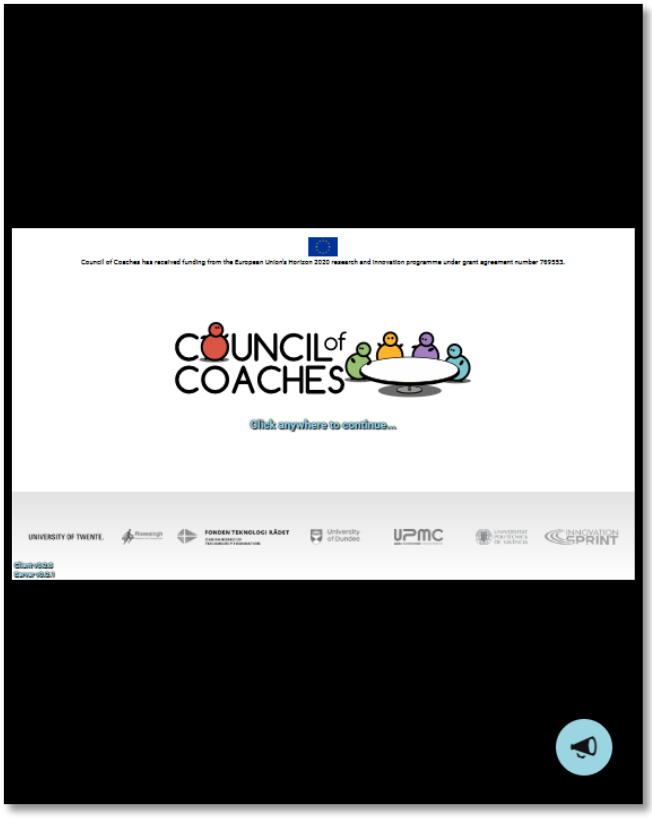

Figure 3: Council of Coaches BETA on a narrow screen.

Finally, CSS supports scaling text sizes to the viewport width or height. But, the actual "application canvas" can differ from the viewport width or height (e.g. in Figure 2 the viewport width is 1920px, but the *application* width is only 1166px). Thus, in this case, text should scale proportionally to the 1166 value, and not to the 1920. In order to fix this, we managed to scale text proportionally to the "application width", by dynamically calculating and setting a CSS variable `--viewport-width-multiplier`, and scale text as follows:

```
.css-class {
    font-size: calc(1.0vw * var(--viewport-width-multiplier));
}
```

4.2 Main Menu

When clicking/tapping anywhere on screen, the Splash Screen animates away and the user is presented with the application's Main Menu (see Figure 4).

Figure 4: Council of Coaches BETA Main Menu.

Developer Commentary:

In the Main Menu we wanted to continue the "professional feel" of the application – including elements like the "Credits scroller" (see Figure 5), a Patch Notes section (see Figure 6), and access to the privacy statement (see Figure 7).

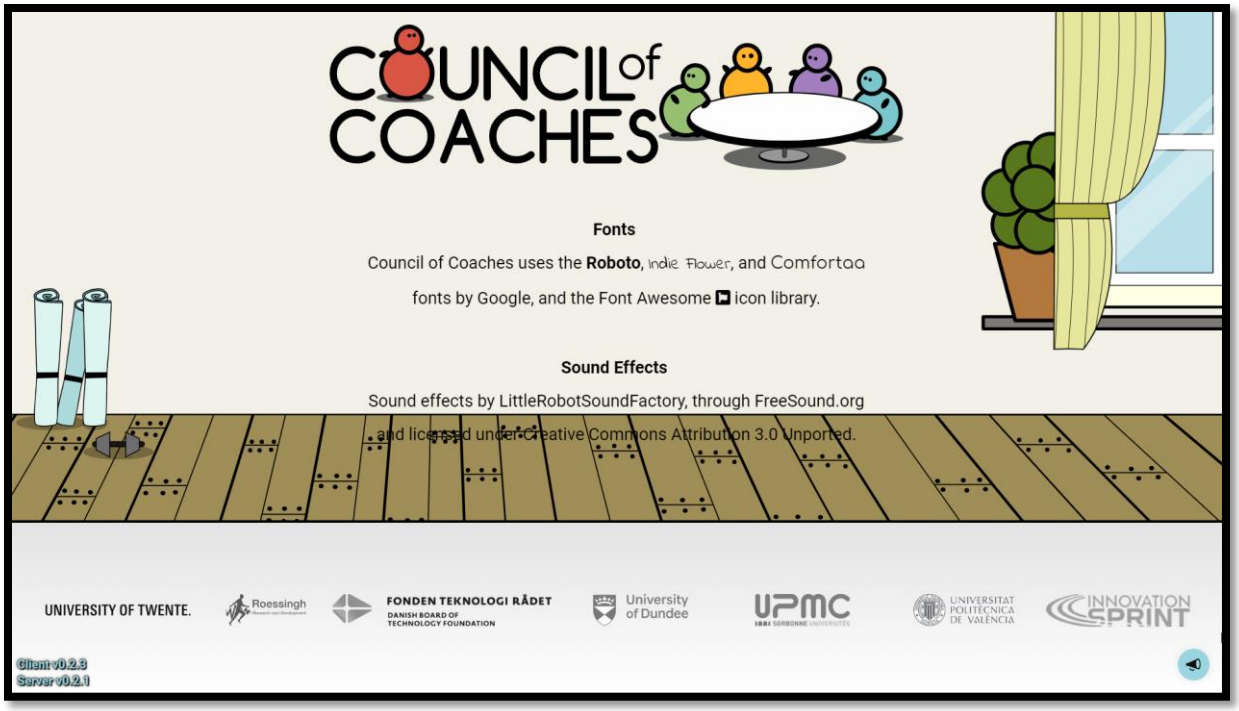

Figure 5: Council of Coaches BETA > Main Menu > Credits.

Figure 6: Council of Coaches BETA > Main Menu > Patch Notes.

Figure 7: Council of Coaches BETA > Main Menu > Privacy Statement.

Besides these features, the Main Menu sets the "mood" for the application, using background graphics from the main *living room* user interface (see Section 4.4). In this way, the Main Menu functions as a "doorway" into the Council of Coaches app, a concept with a matching placing of UI elements that was inspired by the login screen of the popular video game World of Warcraft (see Figure 8). In a future update to the main menu, we aim to further emphasize this "doorway into the Council of Coaches" metaphor by designing it to represent the outside of the coaches' house, with a literal *door* to pass through to login.

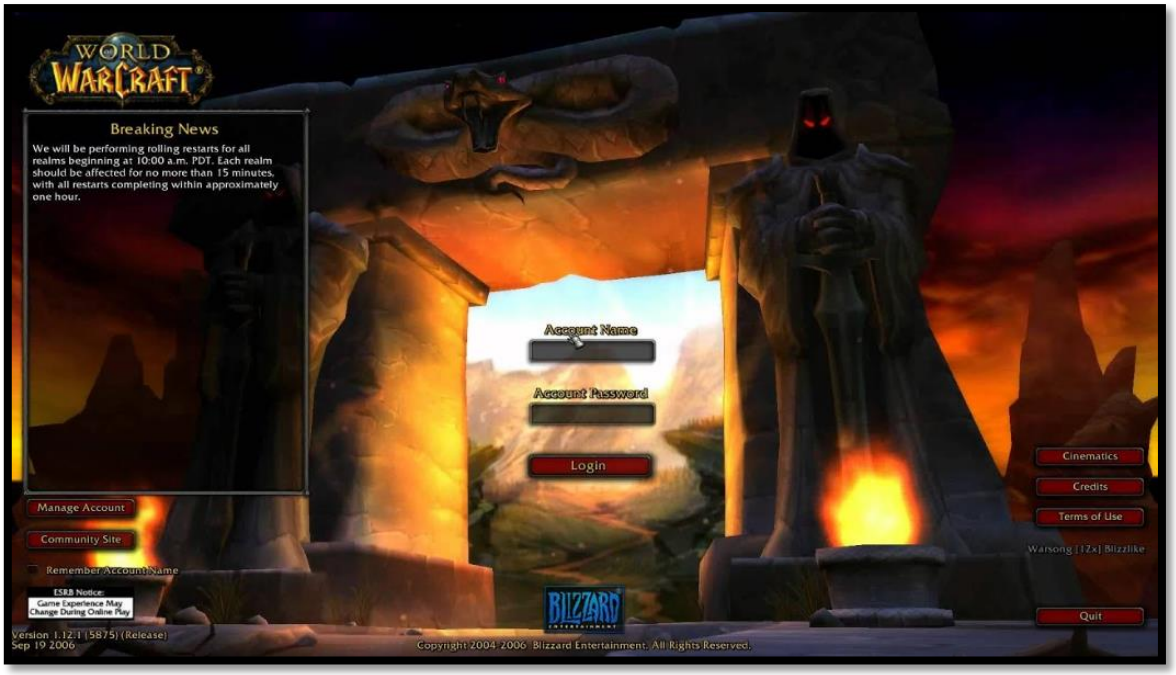

Figure 8: World of Warcraft (2006) login screen, serving as inspiration for the Council of Coaches Main Menu.

4.3 Creating an account

The first thing that any user of the Council of Coaches application has to do is to create an account, so that his preferences and information can be stored, and dialogues can be personalized. The account creation procedure in Council of Coaches serves a dual purpose, as it also explains users how to interact with the core dialogue system.

Developer commentary:

As can be seen in Figure 9 below, the procedure to create an account is not your normal "fill in some form fields, and press a button"-procedure. Since Council of Coaches is all about dialogues, we wanted to start off right away by turning the account creation procedure into a dialogue itself.

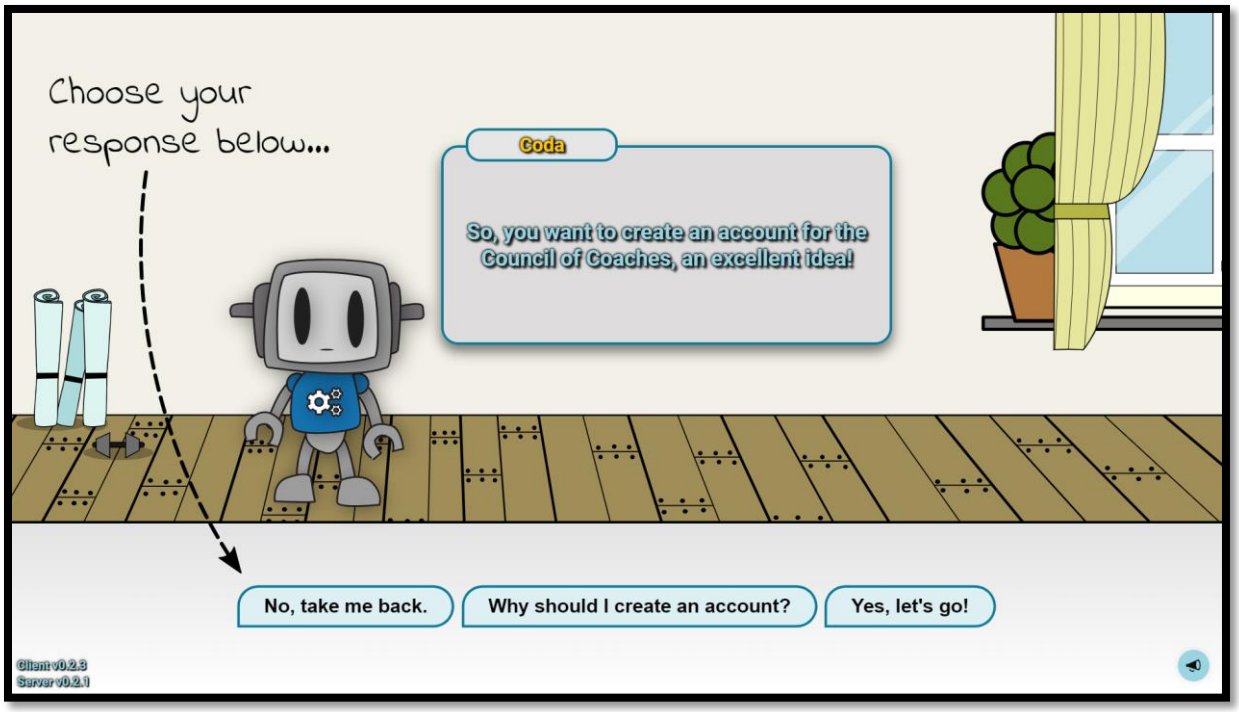

Figure 9: Council of Coaches BETA > Creating Account > Step 1.

For a dialogue, we need a virtual agent. The little robot called "Coda" was designed specifically to execute typical "system features". He (or she) is a robot, and not a "human" to make clear that this is not actually a coach. Coda is the equivalent of an in-app "menu", and is designed for functions like the account creation, logging out of the platform, changing settings.

Furthermore, a new application might need some help explaining how things work – however good or easy your user interface is. Instead of providing a manual, the account procedure integrates "tutorial steps", basically letting the user learn as he progresses through the dialogues (see the "Choose your response below..." pointer in Figure 9).

This combination of "account creation + tutorial" is again drawing inspiration from the world of video games, where in games like Bethesda's Morrowind, the player is dumped into the game world, taught how to move around and asked questions: "Stand up. There you go. You were dreaming. What's your name?" (see Figure 10).

Figure 10: Opening character creation and tutorial scene in Bethesda's Morrowind, see https://youtu.be/VgR2ISC6Ets?t=58.

In the next images, Figure 11, Figure 12, and Figure 13 the tutorial steps for the last three input types are shown: entering text, continuing the conversation, and finishing the conversation.

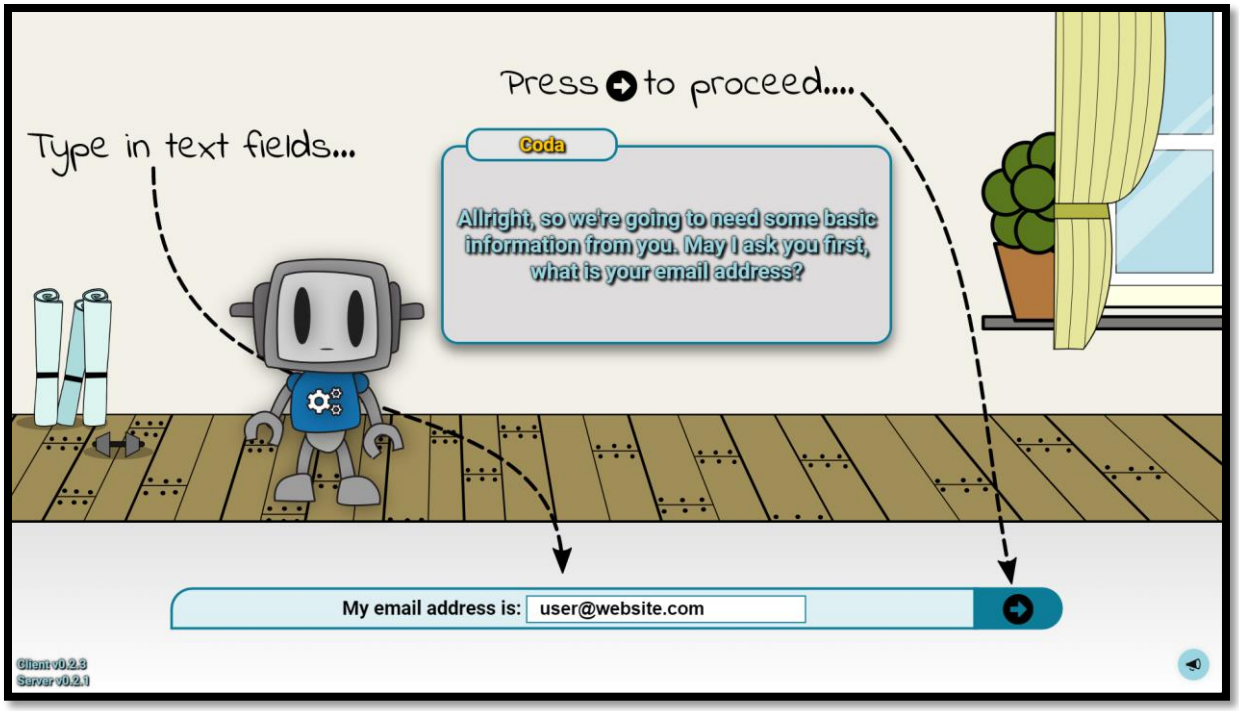

Figure 11: Creating an account, entering your email address.

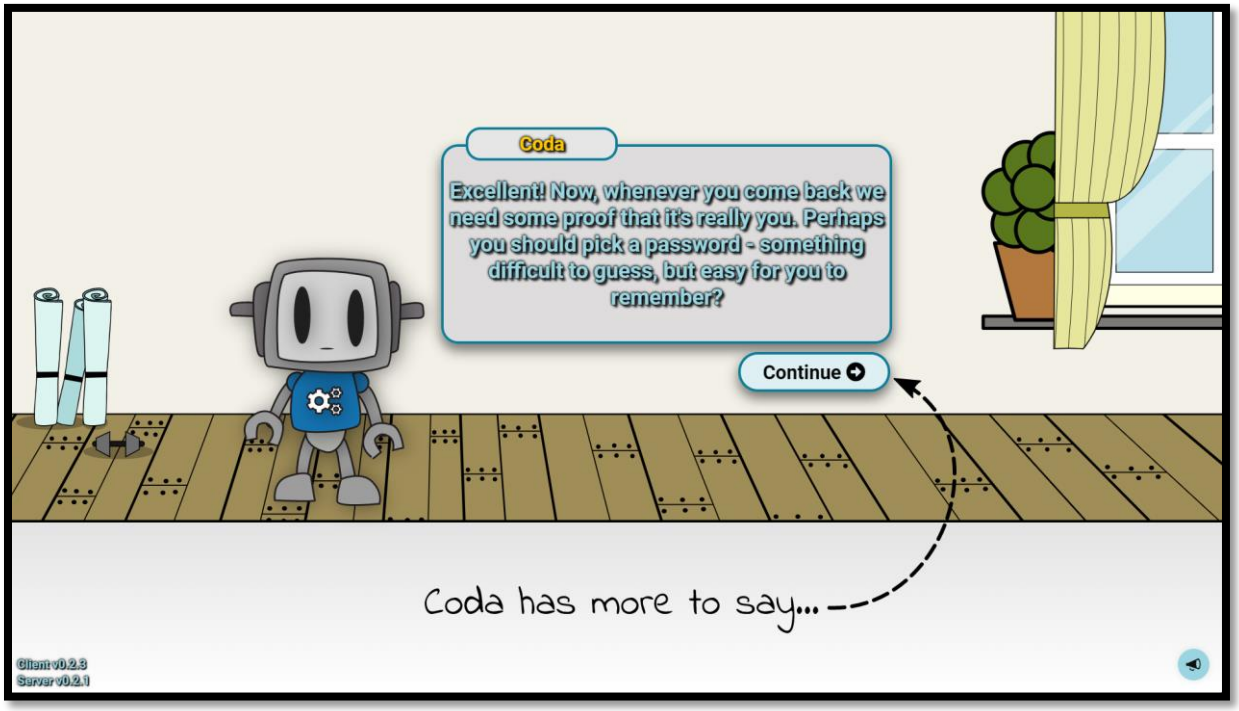

Figure 12: Creating an account, the continue button explained.

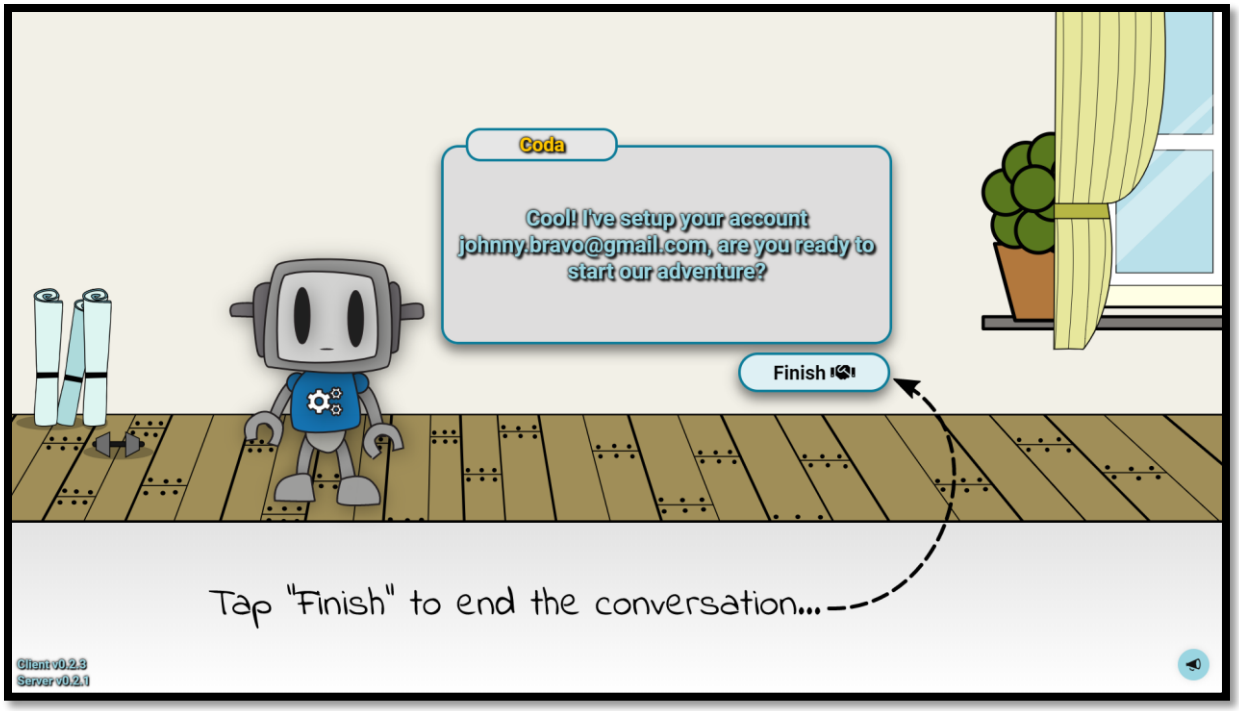

Figure 13: Creating an account, the finish button.

4.4 The Living room (Main User Interface)

Finally, the user interface part where users will spend most of their time when interacting with the Council of Coaches. In this BETA, the living room main user interface houses three coaches: Alexa, Helen and Francois (from left to right), as well as the little robot helper, Coda (see Figure 14).

Figure 14: The Council of Coaches BETA Main "Living room" interface.

Developer commentary:

At first sight it is obvious that Council of Coaches doesn't take a standard approach to its user interface design. We wanted to create something that is fun and engaging, matching the aims we have of fun and engaging dialogues with the user.

The "main scene" design basically had three iterations. The original idea, the one that inspired the project as a whole was to have the coaches represented in their own "boxes", similar to having a group Skype call (see Figure 15).

Figure 15: The origins: Civilization 2, with 5 "coaches" in their own little box.

Such a user interface representation has a major drawback, in that it limits the coaches' ability to interact with each other. Coach #2 can look to his left, to address Coach #4, but it obviously doesn't feel right that they are not situated in the same environment.

The second functional demonstrator as presented in D7.3 already addressed this, by placing the coaches behind a table in a room (similar to the project's logo, see Figure 16). In this way, the coaches share the same environment, and their interactions seem more natural. However, the table setup does give the feeling that the user is in a "job interview".

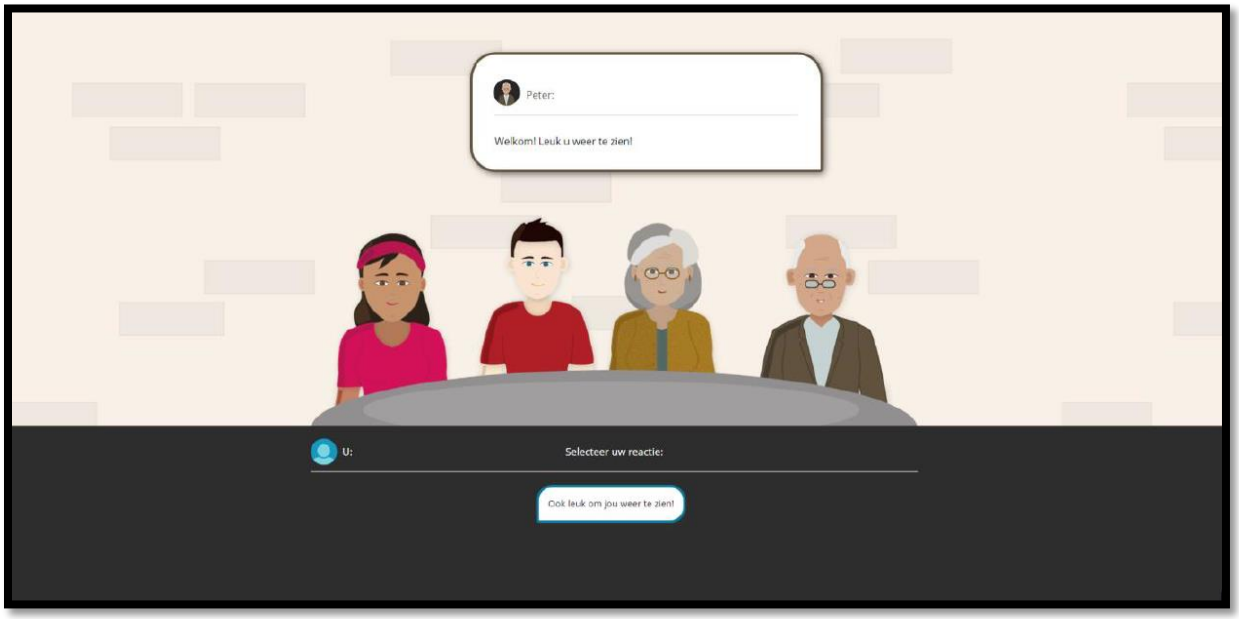

Figure 16: Second functional demonstrator user interface, with the coaches sharing a room (and table).

Thus, the current user interface design puts the coaches together in a room, without creating the "job interview" feeling, by giving them each their own little space within a cosy living room, as seen in Figure 14.

4.5 Widgets

Much of the core functionality of Council of Coaches will be offered through natural language dialogues with the coaches (content will be described in detail in D3.4: *Final coaching actions and content*, due to be released in M27 (November 2019).

However, there are many ways in which the user experience can be approved through other types of media, that we call *Widgets*. In the current Council of Coaches BETA, one of these widgets is available, and can be seen in Figure 17.

Figure 17: The first Council of Coaches Widget: the radio that provides background music.

Developer commentary:

Although many different widgets are currently being conceptualized, including a recipe book, and physical activity book that Alexa can use to show details about the user's physical activity behaviour – these features are tightly connected to the content of the coaches, and are thus not yet ready for release at this point.

So, as an example, we decided to break the silence and offer some background music to the users. The living room design lends itself perfectly for this type of additions, as a little old-fashioned radio is a fitting decoration. Only thing left to do was to make it interactive. This old radio has four channels that the user can choose from, with Jazz and Blues on channel 1, Classical Music on channel 2, different French songs on channel 3 (Francois loves it), and Rock & Roll on channel 4. The audio tracks stem from a collection of 78rpm records, digitized and made available through The Internet Archive[2].

[2] https://archive.org/details/georgeblood&tab=about

5 Technical Prototype software

In order to have an impression of the Technical Demonstrator, without going through the installation and configuration process, the following video demonstration is available, in which the coaches themselves give an overview of the system and its innovations:

https://www.youtube.com/watch?v=lWJvP3_HmXA

In order to build and run the actual software, the following instructions are also available in our project GitLab page: https://gitlab.com/CouncilOfCoaches/TechnicalDemonstrator. The instructions in this document have been clarified and include additional steps to account for software hosted elsewhere (Greta). **For troubleshooting, check the instructions in GitLab for any last-minute fix or update**.

To access our project in GitLab you will have to provide us with your GitLab account in order to grant permissions to access the project, since it is currently private.

Take into account that the first time the prerequisites and installation steps are followed it will take a considerable amount of time (over 1 hour) due to installation processes and download time (depending on Internet speed).

If you find that some of the following links do not work, it may be due to the document format. Copy the link and paste it into your browser.

5.1 Pre-requisites

- Windows 10 (Instructions here are only for *Windows 10 Pro*)
- Unity3D (*version 2017.4.24f1*. Other 2017.x versions might work, but versions 2018.x or later do NOT work). In the installer, include the Android, iOS, tvOS, macOS build supports. https://unity3d.com/get-unity/download/archive.
- Docker (Should work with any current version of Docker, including *Community* for Windows) https://www.docker.com/
- Java 8 JDK (*Update 211*) https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. Make sure you add Java to your Path environment variable.
- Ant build system (*Latest version*) https://ant.apache.org/. Make sure you add Ant to your Path environment variable.
- Visual C++ Redistributable for Visual Studio *versions 2013 and/or 2015* (This is recommended for Unity, although your Windows installation may already have this) https://support.microsoft.com/en-gb/help/2977003/the-latest-supported-visual-c-downloads
- Approximately 10GB of disk space and at least 8GB RAM (recommended)

5.2 Installation

1. Download the code from Council of Coaches GitLab repository https://gitlab.com/CouncilOfCoaches/TechnicalDemonstrator/-/archive/milestone4/TechnicalDemonstrator-milestone4.zip
2. Unpack the contents in any folder you want. We will refer to this folder as *{installation}* from now on.
3. Download https://github.com/jankolkmeier/ASAPToolkitUnity/archive/master.zip and place the contents of the downloaded *ASAPToolkitUnit* folder inside *{installation}\UT_HMI\UnityProject\ASAPUnityProject\Assets\ASAPToolkitUnity*

4. Start Docker. Right click the tray icon and go to *Settings*. Go to *Shared Drives* and share the main drive. Go to *Advanced* and set Memory to 4GB (Recommended).
5. Open a command line shell, go to *{installation}\UDUN_ArgTech* and type the command `docker-compose pull`
6. Open a command line shell, go to *{installation}\UT_HMI\HmiCouch* and type command `ant resolve`
7. Start Unity. Select *Open project*, and then select the folder *{installation}\UT_HMI\UnityProject\ASAPUnityProject*. (You may get a warning dialog depending on your exact version of Unity. Ignore it and *Continue*).
8. In *Project assets* (usually bottom-left), navigate to *\Assets\Couch* and double-click the scene *CouchMain.unity*
9. Download Mary TTS from http://mary.dfki.de/download/index.html, Runtime Package, and unpack the contents in any folder you want. We will refer to this folder as *{mary-installation}* from now on.
10. Go to *{mary-installation}/bin* and run *maryttscomponent-installer.bat*. From that tool, install the following languages: *enUS/cmu-slt, en-US/cmu-bdl, fr/enst-camille, fr/enst-camille-hsmm*.
11. Download the code from Greta GitHub repository master branch: https://github.com/gretaproject/greta/archive/master.zip
12. Unpack the contents in any folder you want. We will refer to this folder as *{greta-installation}* from now on.
13. Go to *{greta-installation}\bin\Player\Lib\External\{your-platform}* and edit the *Plugins***.cfg* files in order to replace any folder path you find there to your actual folder paths.
14. Go to *{greta-installation}\bin* and edit the files *vib.ini* and *Modular.xml* to replace their occurrences of `empty.xml` with `{installation}\UT_HMI\Launchers\couch-environment-for-greta.xml`. In vib.ini, replace as well the value of `MARY_SERVER_DIRECTORY` with `{openmary-installation}\bin`. Then build Greta:
    a. Delete the content in *{greta-installation}/bin/Common/Lib/Internal* directory
    b. Delete the content in *{greta-installation}/bin/Player/Lib/Internal* directory
    c. Delete the file *{greta-installation}/bin/Modular.jar*
    d. Open a command line shell, go to *{greta-installation}/application/Modular* directory and execute the command: `run ant -buildfile build_manual.xml -Dant.build.javac.target=1.8 -Dant.build.javac.source=1.8`

5.3 Execution

1. **Run OpenMary TTS**: Open a command line shell, go to *{openmary-installation}\bin* and execute `marytts-server.bat`. Wait until it is up and running on port 59125
2. **Run Greta**: Open a command line shell, go to *{greta-installation}\bin* and type the command `java -jar Modular.jar`. The Greta user interface window will open. From its menus, select *File* > *Open* and go to *{installation}\UT_HMI\Launchers*, and select *CouchGretaUnity-Simple.xml*
3. **Run Docker** (if not started before)
4. **Run the Dialog Framework**: Open a command line shell, go to *{installation}\UDUN_ArgTech* and type the command `docker-compose up`. Wait until it is up and running.
5. **Run ASAP**: Open a command line shell, go to *{installation}\UT_HMI\Launchers* and run `ASAP_Superior_Couch_Start_NoAndroid.bat`. Wait until you see the message *"Waiting for AgentSpec..."*
6. **Run Unity**: Open the Unity project (if not started before). Press the *Play* button (usually at the top) and wait for the agents to take their place (you may be asked to allow firewall access).
7. **Run Flipper**: Open a command line shell, go to *{installation}\UT_HMI\Launchers* and run `Flipper_Superior_Couch_Start.bat`. Wait until the scenario begins.

[To restart the dialog you need to restart only Flipper]

5.3.1 Android

You can optionally try to add the Android-based coach by installing the Android applications found in *{installation}\UT_HMI\UnityProject\Binaries\Android*. Install the *AndroidDemonstrator.apk* and *CouchWizard.apk* applications in an Android device that is connected to the same network as the rest of the system. Then, when running ASAP, you have to run `ASAP_Superior_Couch_Start.bat` instead of the *NoAndroid* one.

5.3.2 Stopping

- **Unity**: Press the *Play* button again (and close the Unity editor).
- **Greta**: Select the shell window with Greta and press *Ctrl+c*. Do the same with OpenMary if its window is still open.
- **ASAP**: Select the shell window with ASAP and press *Ctrl+c*.
- **Flipper**: Select the shell window with Flipper and press *Ctrl+c*.
- **Dialog Framework**: Select the shell window where you executed `docker-compose up` and press *Ctrl+c*. Wait for Docker containers to quit. Then execute `docker-compose down`.
- **Docker**: You can then quit Docker by right-clicking its system tray icon.

6 Innovation beyond state of the art

The following subsections highlight the innovations beyond the current state of the art that have been implemented during both the Functional Prototype and Technical Prototype development tracks, at the point of the release of this deliverable (point at which they are encapsulated in their respective Demonstrators).

6.1 Functional Demonstrator

The Functional Demonstrator is, as the name implies, built to demonstrate *function*. By definition, the technology used here is not beyond any state of the art. The demonstrator is built to be easy to deploy, easy to extend, scalable, and usable cross-device. The result is a robust HTML/JavaScript web-client that runs on any PC/laptop or tablet device with a large enough screen (in fact, it *will* run on phones, the user interface simply isn't designed for it).

The demonstrated functionality that *is* beyond the state of the art is the content of the multi-coaching sessions that form the core of the Council of Coaches project. The aim of the demonstrator is to show the impact of providing coaching on multiple domains, and from multiple viewpoints. To show the effects of vicarious persuasion by presenting discussions between the coaches. The ability to keep the coaching conversation going without direct input from the end-user. These fundamental concepts, not possible to achieve with a single-coach application, represent the innovation beyond the state of the art for the Functional Demonstrator.

6.2 Technical Demonstrator

The Technical Demonstrator integrates the different modules of the Council of Coaches system in the Technical Prototype development track. As such, it encompasses all the innovations brought up by the modules it is composed of. These are explained individually in the following subsections.

From the point of view of the integration itself, the innovation comes from the fact that all these state-of-the-art modules (and their respective innovations beyond that state of the art) have been brought together to build up a cohesive system. This is in opposition to any standalone, ad-hoc system put in place during their respective developments to demonstrate their features in isolation.

6.2.1 Innovation in Measuring User Behaviour

Eliciting effective and relevant coaching demands a good understanding on different aspects of the user's behaviour. Accordingly, one major innovation of this Technical Demonstrator revolves around the measurement and modelling of users' behaviour in a holistic fashion. Not only physical behaviours - highly explored in past projects - but also social, emotional and cognitive behaviours are measured via diverse on-body and off-body sensors. This is made possible via the so-called Holistic Behaviour Analysis Framework (HBAF), which is aimed at generating objective and continuous user's background information for improving the user-coach interaction and supporting the dialog and argumentation process.

Smartphones are used as the primary source of data as they happen to be available to most of the population and they provide a rich set of sensor data types. Despite smartphones have been already used for detecting physical behaviour (e.g., steps from accelerometer data), something that is also leveraged here, this demonstrator moves beyond prior work while exploiting raw smartphone sensor data to automatically determine social (non)interaction of the user with other people. This includes data generated during implicit and explicit interactions of the user with their phones, combining Bluetooth, phone call and SMS logs, ambient noise and location data in a model trained to scrutinise whether the user is socially active or not.

6.2.2 Innovation in Automated Personalized Multi-Party Dialogue

Underpinning the communication in coaching sessions within the Council of Coaches is the Dialogue Game Execution Platform (DGEP). Several aspects of DGEP have been updated and extended to support the types of dialogue required in a coaching session.

First, DGEP now supports fully-autonomous dialogue between multiple non-human agent participants. This allows the coaches in the council to conduct a conversation amongst themselves without any user involvement; this behaviour is advantageous in situations where, for instance, the user is not fully engaged with the system, and generally adds realism to the overall coaching session. Rather than it always being incumbent on the user to push the dialogue forward, the coaches can also do it amongst themselves.

Second, it is possible for dialogues within DGEP to trigger "sub-dialogues" to discuss or resolve a new issue that only became apparent during the "primary" dialogue. For example, the user and the coaches might be discussing ways to increase daily levels of physical activity (i.e. the "Tips & Advice" topic) when the coach realises he doesn't know about the users interests, requiring a different dialogue to take place (i.e. the "Users Interests" topic), while also keeping a bookmark in the original dialogue (that is, to pick up the discussion again later).

The selection of the topics to be discussed is an integral part of the Knowledge Base component that feeds DGEP in this way. Models of coaching topics are being developed for each of the virtual coaches, with their contents being derived from the literature of behaviour change techniques (see D3.1). In this way, while DGEP manages the dynamic low-level conversational structures, the higher level "topic" structure of coaching dialogues is also managed in a dynamic way.

Finally, a new module has been added that allows dialogues specified in the Yarn editor to be executed using DGEP. This allows for rapid prototyping of dialogues and content for feeding into the Technical Demonstrator, without requiring a fully abstract dialogue system to be specified, which in turn allows for convenient testing of DGEP's integration with other systems (both in the Council of Coaches and beyond).

6.2.3 Innovation in Multi-Party Embodied Conversational Agent Systems

The multi agent coaching approach of the Council of Coaches is supported by the "Situation, Agent, Intention, Behaviour, Animation" (SAIBA) platforms Greta (SU) and Articulated Social Agents Platform (ASAP) (University of Twente). To support multi-party embodied conversations within the Council of Coaches project these platforms need to be extended to be able to support multiple agents.

The ASAP platform now does support multiple agents. To be able to support multi-party coaching conversation the Behaviour Mark-up Language (BML) standard is extended to support synchronization of verbal and nonverbal behaviour between the agents.

The ASAP agents will be integrated in a Unity scene which makes it possible to be exported to different devices, such as PCs (Windows and MacOS) and mobile devices (Android powered phones and tablets), which makes it possible to have multi-party multi-device coaching conversations. Both ASAP and Greta do support agents in Unity, and for the first time agents driven by both platforms are present in one scene, and behaviour of the agents from both platforms can be synchronized.

The Greta platform now supports multiple agents. Both BML and FML standard can be used to make the agents perform verbal and non-verbal behaviour.

Regarding this non-verbal behaviour, the agents of Greta platform have three innovative features that make them more realistic and behave in a more human-like manner:

- Gaze behaviour: having as input the ID of the object or other agents in the environment to look at, the character is able to gaze at the target computing automatically the body parts to involve (eyes, head, shoulder and torso) in that movement. The agent can also gaze at the user thanks the user's head in real-time.
- Backchannels: the Greta platform has been integrated with the study of (Bevacqua, 2007), which allows the virtual agent to provide feedback (voice, head, face, gaze, posture and gesture) to the addressee expressing intentions of perception, attention, interest, understanding, attitude or acceptance.
- Automatic gestures: the Greta platform has been integrated with the study of (Ravenet, 2018). The virtual agent can perform representational gestures to illustrate the content of the speech. So metaphoric gestures are automatically produced, aligned with the speech of the agent in terms of timing and meaning.

7 Bibliography

Bevacqua, E. H. (2007). Facial feedback signals for ECAs. In *AISB* (pp. 328-334).

Ravenet, B. P. (2018). Automating the production of communicative gestures in embodied characters. In *Frontiers in psychology, 9*.